საქართველოს ტექნიკური უნივერსიტეტი

თეიმურაზ სუხიაშვილი

პროგრამული სისტემის დამუშავება ინსტრუმენტალური საშუალება **IBM Rational Rose** გამოყენებით

დამხმარე სახელმძღვანელო პრაქტიკული და ლაბორატორიული სამუშაოების შესასრულებლად

“ტექნიკური უნივერსიტეტი”

თბილისი 2018

უაკ 681.3

დამხმარე სახელმძღვანელო შეიცავს დავალებებს და სავარჯიშოებს პროგრამული უზრუნველყოფის მოდელების ასაგებათ მეთოდიკის გამოყენებით, რომელიც გამოიყენება პროგრამული უზრუნველყოფის შესაქმნელად უნიფიცირებულ პროცესში (ყველაზე სრულად რეალიზებულ ტექნოლოგიაში IBM Rational Unified Process). მოდელირების სტანდარტული ენის UML მოდელებისა და დიაგრამების აგება სრულდება CASE ინსტრუმენტალური საშუალება IBM Rational Rose გამოყენებით.

მოცემული სახელმძღვანელოს მიზანია – მომხმარებლებში დამოუკიდებელი პრაქტიკული უნარ-ჩვევების ფორმირება რთული საინფორმაციო სისტემების პროგრამული უზრუნველყოფის დასაპროექტებლად თანამედროვე მეთოდებითა და საშუალებებით, დაფუძნებული ვიზუალური მოდელების გამოყენებაზე. განკუთვნილია ინფორმატიკის სპეციალობის სტუდენტების, მაგისტრანტებისა და სპეციალისტებისათვის.

შინაარსი

ლაბორატორიული სამუშაო №1
CASE საშუალებების გამოყენება პროგრამული სისტემის დამუშავებისათვის. ინსტრუმენტალური საშუალება IBM Rational Rose

1. ლაბორატორიული სამუშაოს დანიშნულება

სამუშაოს დანიშნულებაა შევისწავლოთ:

- პროგრამული სისტემის დამუშავების, ანალიზისა და დაპროექტების პროცესის ავტომატიზაცია (CASE) რაციონალური უნიფიცირებული პროცესის(RUP) საფუძველზე;
- ინსტრუმენტალური საშუალება IBM Rational Rose-ს დანიშნულება და შემადგენლობა(რეპოზიტორიუმი, მომხმარებლის გრაფიკული ინტერფეისი, პროექტის გადახედვის საშუალება (ბრაუზერი), პროექტის კონტროლის, სტატისტიკის აკრეფისა და დოკუმენტების გენერაციის საშუალებები);
- ინსტრუმენტალური საშუალება IBM Rational Rose-ს ინტერფეისი (ბრაუზერი, დოკუმენტაციის ფანჯარა, ინსტრუმენტების პანელი, დიაგრამის ფანჯარა და ჟურნალი).

2. მეთოდური მითითებები ლაბორატორიული სამუშაოს შესასრულებლად

ამჟამად სტანდარტულ ინსტრუმენტს პროგრამული უზრუნველყოფის „მონახაზის” შესაქმნელად წარმოადგენს მოდელირების უნიფიცირებული ენა - Unified Modeling Language (**UML**). მისი მეშვეობით შესაძლებელია პროგრამული სისტემების არტეფაქტების ვიზუალირება, სპეციფიცირება, კონსტრუირება და დოკუმენტირება.

UML ენა გახდა საფუძველი ობიექტ-ორიენტირებულ მეთოდოლოგიაზე დაფუძნებული პროგრამული სისტემის დამუშავების, ანალიზისა და

დაპროექტების პროცესის ავტომატიზაციისა (Computer Aided Software Engineering, CASE).

პროგრამული სისტემის დამუშავება მოითხოვს მოწესრიგებულ მიდგომას იმასთან, თუ როგორ უნდა განაწილდნენ სამუშაოები და პასუხისმგებლობები ორგანიზაციაში, რომელიც დაკავებულია წარმოების პროცესით. დამუშავების არსებული მეთოდიკებიდან საუკეთესოდ მიჩნეულია რაციონალური უნიფიცირებული პროცესი, რომელიც ყველაზე კარგად არის მისადაგებული **UML** ტექნოლოგიასთან და შესაძლებელია ადაპტირებულ იქნას სხვადასხვა დანიშნულების პროექტების ორგანიზებისათვის.

სამუშაოს არსი რაციონალური უნიფიცირებული პროცესის ფარგლებში – ეს მოდელის შექმნა და თანხლებაა. მოდელები, რომლებიც გამოხატულია **UML** ენაზე, გვაძლევენ სემანტიკურად გაჯერებულ წარმოდგენას დასამუშავებელ პროგრამულ კომპლექსზე.

რაციონალური უნიფიცირებული პროცესი ბაზირდება უბრალო და გასაგებ არქიტექტურაზე, რომელიც უზრუნველყოფს კონცეპტუალურ ერთიანობას დამუშავების მთელ პროცესში, ამასთან ადაპტირდება სხვადასხვა სიტუაციებთან. სწორედ რაციონალური უნიფიცირებული პროცესი უდევს საფუძვლად ობიექტ-ორიენტირებული მიდგომით პროგრამული სისტემების დამუშავების CASE ინსტრუმენტალურ საშუალებებს. პირველი ასეთი ინსტრუმენტალური CASE საშუალებაა– Rational Rose 98.

ინსტრუმენტალური საშუალება IBM Rational Rose განკუთვნილია პროგრამული უზრუნველყოფისათვის მოდელების ასაგებათ მისი ანალიზისა და პროექტირების პროცესში, ასევე კოდის გენერაციისათვის დაპროგრამების სხვადასხვა ენებზე და საპროექტო დოკუმენტაციის მოსამზადებლად. Rose გამოიყენება პროგრამული უზრუნველყოფის ობიექტ-ორიენტირებული ანალიზისა და დაპროექტების პროცესში, რომელიც

აღწერილია ტექნოლოგიაში Rational Unified Process (RUP). Rose-ს მუშაობას საფუძვლად უდევს მოდელთა ელემენტების შექმნა თავისი სპეციფიკაციებით და მოდელირების UML ენის დიაგრამების აგება, რომლებიც განსაზღვრავენ სისტემის არქიტექტურას, მის სტატიკურ და დინამიკურ ასპექტებს. Rose-ს შემადგენლობაში შესაძლებელია ექვსი ძირითადი კომპონენტის გამოყოფა: რეპოზიტორიუმი, მომხმარებლის გრაფიკული ინტერფეისი, პროექტის გადახედვის საშუალება (ბრაუზერი), პროექტის კონტროლის საშუალებები, სტატისტიკის აკრეფისა და დოკუმენტების გენერაციის.

რეპოზიტორი წარმოადგენს პროექტის მონაცემთა ბაზას. ბრაუზერი უზრუნველყოფს ნავიგაციას პროექტში, მათ შორის გადაადგილებას კლასისა და ქვესისტემების იერარქიაში, გადართვას ერთი სახის დიაგრამიდან მეორეზე და ა.შ. პროექტის კონტროლისა და სტატისტიკის აკრეფის საშუალებები გვაძლევენ საშუალებას ვიპოვოთ და შევასწოროთ შეცდომები პროექტის განვითარების პროცესში, და არა მისი აღწერის დამთავრების შემდეგ. ანგარიშების გენერატორი ახდენს გამოსასვლელი დოკუმენტების ტექსტების ფორმირებას რეპოზიტორიუმში არსებული ინფორმაციის საფუძველზე.

კოდის ავტომატიზებული გენერაციის საშუალებები, კლასების და კომპონენტების დიაგრამაზე არსებული ინფორმაციის საფუძველზე, ახდენს კლასების აღწერის ფაილის ფორმირებას. ასეთი გზით ფორმირებული პროგრამის ჩონჩხი შემდეგ შესაძლებელია დაზუსტდეს პირდაპირი დაპროგრამების გზით შესაბამის დაპროგრამების ენაზე(ძირითადი ენები, რომლებიც გამოიყენება არის Java, C++).

Rose-ს საშუალებით დამუშავებული პროექტი ახდენს შემდეგი დოკუმენტების ფორმირებას:

- UML დიაგრამები, რომლებიც ერთობლივად ქმნიან დასამუშავებელი პროგრამული სისტემის მოდელს;

- კლასების, ობიექტების, ატრიბუტებისა და ოპერაციების სპეციფიკაციებს;
- პროგრამული ტექსტების ნამზადებს.

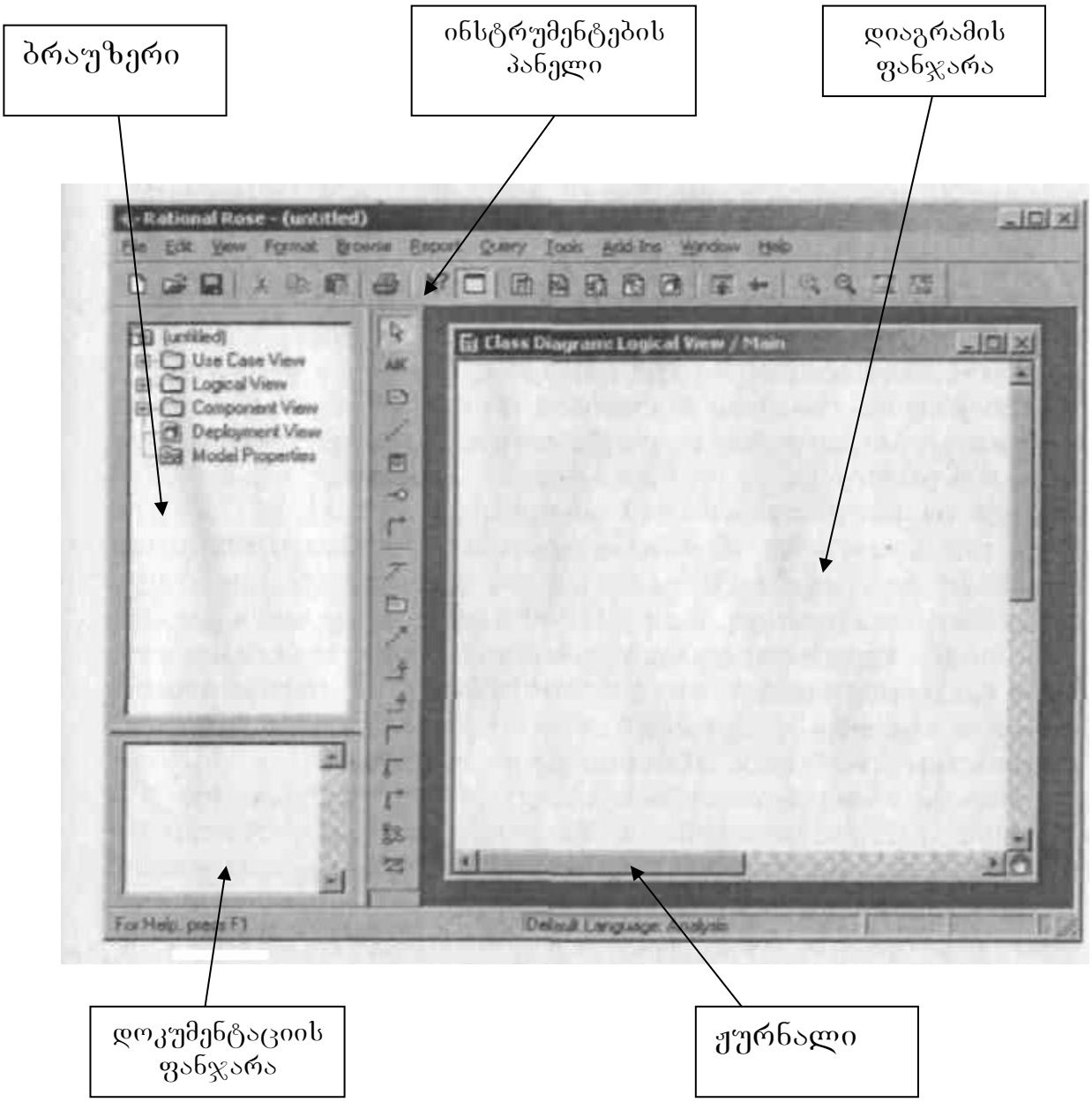
პროგრამული ტექსტები წარმოადგენენ ნამზადებს პროგრამისტების შემდგომი მუშაობისათვის. ინფორმაციის შემადგენლობა, რომელიც ჩაირთვება პროგრამულ ფაილში, განისაზღვრება, ან დუმილით, ან მომხმარებლის გათვალისწინებით. შემდეგში ეს პროგრამული ტექსტები პროგრამისტების მიერ გარდაიქმნებიან სრულყოფილ პროგრამებში.

Rose-ს ინტერფეისი შედგება ხუთი ძირითადი ელემენტისაგან – ბრაუზერი, დოკუმენტაციის ფანჯარა, ინსტრუმენტების პანელი, დიაგრამის ფანჯარა და ჟურნალი(იხ.ნახ.1.1.).

ბრაუზერი – ეს იერარქიული სტრუქტურაა, რომელიც საშუალებას იძლევა მოვახდინოთ სწრაფი ნავიგაცია მოდელზე. ყველაფერი, რაც ემატება მას – მოქმედი პირები, გამოყენებითი შემთხვევები, კლასები, კომპონენტები - ნაჩვენები იქნება ბრაუზერის ფანჯარაში.

ბრაუზერის მეშვეობით შესაძლებელია:

- დაუმატოთ მოდელს ელემენტები(მოქმედი პირები, გამოყენებითი შემთხვევები, კლასები, კომპონენტები, დიაგრამები და ა.შ.);
- ვნახოთ მოდელის არსებული ელემენტები და ამ ელემენტებს შორის კავშირები;
- გადავაადგილოთ მოდელის ელემენტები;
- დავამატოთ მოდელის ელემენტები დიაგრამას;
- დაუკავშიროთ ელემენტი ფაილს ან URL-ს (Uniform Resource Locator);
- დავაჯგუფოთ ელემენტები პაკეტებში;
- ვიმუშაოთ ელემენტის დეტალიზირებულ სპეციფიკაციასთან;
- გავხსნათ დიაგრამა.



ნახ.1.1. ინტერფეისი

ბრაუზერი ეყრდნობა მოდელის ოთხ წარმოდგენას(view); გამოყენებით შემთხვევათა, კომპონენტების, განლაგების და ლოგიკური წარმოდგენა. ყველა ისინი და მათში არსებული მოდელის ელემენტები აღწერილია 1.1.2. განყოყილებაში.

ბრაუზერი ორგანიზებულია ხის მაგვარ სტილში. მოდელის ყოველი ელემენტი შესაძლებელია შეიცავდეს სხვა ელემენტებს, რომლებიც იმყოფებიან იერარქიაში მის ქვემოთ. ნიშანი “–” ელემენტთან ნიშნავს, რომ მისი კვანძი მთლიანად გახსნილია, ნიშანი “+” – კვანძი შეკრულია.

დოკუმენტაციის ფანჯრის მეშვეობით შესაძლებელია მოდელის ელემენტების დოკუმენტირება, მაგალითად, გაკეთდეს ყოველი მოქმედი პირის მოკლე დახასიათება. კლასის დოკუმენტირებისას, ყველაფერი რაც დაიწერება დოკუმენტაციის ფანჯარაში, შემდეგ გამოჩნდება როგორც კომენტარი გენერირებულ კოდში, რაც აგვაცილებს მისი ხელით შეტანის აუცილებლობისაგან. დოკუმენტაცია გამოიტანება აგრეთვე ანგარიშებში, რომლებიც იქმნება Rose-ში.

ინსტრუმენტების პანელი უზრუნველყოფს სწრაფ მიმართვას ყველაზე გავრცელებულ ბრძანებებთან. ამ გარემოში არსებობენ ინსტრუმენტების ორი პანელი: სტანდარტული და დიაგრამების პანელი. სტანდარტული პანელი ჩანს ყოველთვის, მისი ნიშნულები შეესაბამებიან ბრძანებებს, რომლებიც შესაძლებელია გამოყენებულ იქნას ნებისმიერ დიაგრამასთან მუშაობისათვის.

დიაგრამის პანელი თითეული UML ტიპის დიაგრამას აქვს თავისი. ინსტრუმენტების ყველა პანელი შესაძლებელია შეიცვალოს და დაყენებული იქნას მომხმარებლის მიერ. ამისათვის ავირჩიოთ მენიუდან Tools>Options, ხოლო შემდეგ ჩანართი Toolbars.

იმისათვის, რომ უჩვენოთ ან დავმალოთ ინსტრუმენტების სტანდარტული პანელი(ან დიაგრამის ინსტრუმენტალური პანელი):

- ავირჩიოთ პუნქტი Tools>Options,
- ამოვირჩიოთ ჩანართი Toolbars,
- მოვნიშნოთ (მოვხსნათ მონიშვნა) საკონტროლო გადამრთველზე Show Standard ToolBar (Show diagram ToolBar), იმისათვის, რომ გავხადოთ ხილვადი ან უხილავი ინსტრუმენტების სტანდარტული პანელი.

დიაგრამის ფანჯარაში ჩანს, თუ როგორი სახე აქვს UML მოდელის დიაგრამას. დიაგრამის ელემენტებში ცვილებების შეტანისას ავტომატურათ განახლდება ბრაუზერი. ასევე ელემენტის შეცვლისას ავტომატურათ

განახლდება შესაბამისი დიაგრამა ბრაუზერის მეშვეობით, რაც უზრუნველყოფს მოდელის შესაბამისობის შენარჩუნებას.

მოდელზე მუშაობის პროცესში გარკვეული ინფორმაცია გამოიტანება **ჟურნალის** ფანჯარაში, მაგალითად, შეტყობინება შეცდომების შესახებ, რომლებიც აღიძვრება კოდის გენერაციის დროს.

3.საკონტროლო კითხვები

- რა არის რაციონალური უნიფიცირებული პროცესი;
- რა არის CASE საშუალება;
- რა დანიშნულება აქვს ინსტრუმენტალურ საშუალება IBM Rational Rose-ს;
- როგორია ინსტრუმენტალურ საშუალება IBM Rational Rose-ს შემადგენლობა;
- რა დოკუმენტების ფორმირება შეუძლია IBM Rational Rose-ს;
- რისგან შედგება IBM Rational Rose-ს ინტერფეისი.

4.დავალება

- მოვახდინოთ ინსტრუმენტალური საშუალება IBM Rational Rose-ს პროგრამული პაკეტის ინსტალაცია.
- ინსტრუმენტალური საშუალება IBM Rational Rose-ს ინტერფეისის გაცნობა.

ლაბორატორიული სამუშაო №2

მოდელის წარმოდგენა და მუშაობა ROSE გარემოში

1.სამუშაოს მიზანი

სამუშაოს მიზანია შევისწავლოთ:

- მოდელის წარმოდგენა ROSE გარემოში;
- მოდელების შექმნა;
- მოდელების შენახვა;
- Rose-ს მოდელის ელემენტებში ფაილებზე მიმართვის და URL მისამართების დამატება;
- გამოსახულების პარამეტრების დაყენება.

2.მეთოდური მითითებები ლაბორატორიული სამუშაოს შესასრულებლად

მოდელი Rose-ში წარმოიდგინება ოთხი წარმოდგენით – გამოყენებით შემთხვევათა, ლოგიკური, კომპონენტების და განლაგების.

გამოყენებით შემთხვევათა წარმოდგენა შეიცავს ბიზნეს-პროცესებისა და გამოყენებით შემთხვევათა მოდელებს. ნახ.1.2.-ზე ნაჩვენებია თუ როგორ გამოიყურება გამოყენებით შემთხვევათა წარმოდგენა Rose-ს ბრაუზერში(მოდელის სტრუქტურის შაბლონის გამოყენებისას, რომელიც მიღებულია ტექნოლოგიაში Rational Unified Process).

გამოყენებით შემთხვევათა წარმოდგენა შეიცავს:

- ბიზნეს-პროცესების მოქმედ პირებს;
- გამოყენებით შემთხვევებს ბიზნეს-პროცესების თვალთახედვით;
- სისტემის მოქმედ პირებს და გამოყენებით შემთხვევებს;
- დოკუმენტაციას გამოყენებითი შემთხვევების მიხედვით, რომლებიც ახდენენ პროცესების დეტალიზირებას (სცენარები და მოვლენათა

ნაკადები). ყოველი დოკუმენტი შეესაბამება გარეშე ფაილს, რომელიც მიბმულია მოდელის ელემენტთან. ამ დოკუმენტის ნიშნული დამოკიდებულია დანართისაგან, რომელიც გამოიყენება მოვლენათა ნაკადის დოკუმენტირებისათვის;

- გამოყენებითი შემთხვევათა დიაგრამებს, რომლებიც აღწერენ მოქმედ პირებს, გამოყენებით შემთხვევებს და მათ შორის ურთიერთქმედებას;
- მოღვაწეობის დიაგრამებს, რომლებიც გამოიყენება ბიზნეს-პროცესებისა და სისტემის გამოყენებითი შემთხვევების სცენარების აღწერისათვის;
- პაკეტებს, რომლებიც შეიცავენ გამოყენებითი შემთხვევების და/ან მოქმედი პირების ჯგუფებს.

ნახ.1.2. გამოყენებით შემთხვევათა წარმოდგენა

ლოგიკური წარმოდგენა (ნახ.1.3.) განსაზღვრავს იმას, თუ როგორ მოხდება გამოყენებით შემთხვევებში მოყვანილი ქცევის რეალიზება სისტემის მიერ. იგი მოიცავს ძირითადათ კლასებს და კლასების დიაგრამებს. ლოგიკური წარმოდგენა შეიცავს:

- კლასებს, რომლებიც წარმოადგენენ სისტემის არქიტექტურის ძირითად ელემენტებს;

ნახ.1.3. ლოგიკური წარმოდგენა

- კლასების დიაგრამებს, რომლებიც გამოიყენებიან კლასების, მათი ატრიბუტების, ოპერაციების და კავშირების აღწერისათვის;
- ურთიერთქმედების დიაგრამები, რომლებიც გამოიყენებიან გამოყენებით შემთხვევების სცენარებში ან რომელიმე სისტემური ოპერაციის რეალიზაციაში მონაწილე ობიექტების აღწერისათვის;
- მდგომარეობის დიაგრამებს, რომლებიც აღწერენ კლასების ობიექტების ქცევის დინამიკას;
- პაკეტებს, რომლებიც შეიცავენ ურთიერთდამოკიდებული კლასების ჯგუფებს. ტიპიური სისტემა შეიძლება შეიცავდეს ასზე მეტ კლასს და მათი გაერთიანება პაკეტებში ამცირებს მოდელის სირთულეს. სისტემის საერთო სურათის გაგებისათვის საკმარისია პაკეტების ნახვა.

ლოგიკური წარმოდგენის შევსება ხდება სამ ეტაპად.

პირველ ეტაპზე (ბიზნეს-პროცესების მოდელირებისას) აიგება ბიზნეს-ანალიზის მოდელი, რომელიც შეიცავს კლასებს სტერეოტიპით <<busines worker>> (შემსრულებელი) და <<busines entity>> (არსება). ბიზნეს-ანალიზის მოდელი შესაძლებელია შედგებოდეს სხვადასხვა ტიპის მოდელებისაგან. მოდელის შემადგენლობაში აუცილებლად უნდა შედიოდეს კლასების დიაგრამა, რომელიც შეიცავს შემსრულებლებს და არსებს. კლასების დიაგრამის გარდა ბიზნეს-ანალიზის მოდელი შეიძლება შეიცავდეს:

- მიმდევრობის დიაგრამას, რომლებიც აღწერენ ბიზნეს use case-ს სცენარებს შეტყობინებათა გაცვლის თანმიმდევრობის სახით ობიექტები-მოქმედ პირებსა და ობიექტები-შემსრულებლებს შორის;
- მოღვაწეობის დიაგრამებს ობიექტების ნაკადებითა და ბილიკებით, რომლებიც აღწერენ კავშირებს ერთი ან რამოდენიმე ბიზნეს use case-ს სცენარებს შორის;

- მდგომარეობის დიაგრამებს, რომლებიც აღწერენ ცალკეული არსება-კლასების ობიექტების ქცევებს.

მეორე ეტაპზე ანალიზის პროცესში განისაზღვრება ანალიზის კლასები: მოსაზღვრე (boundary) კლასები, მმართველი (control) კლასები და არსები(entity). კლასების დიაგრამები, რომლებიც ახდენენ გამოყენებით შემთხვევათა რეალიზებას, და ურთიერთქმედების დიაგრამები, რომლებიც აღწერენ ობიექტების ურთიერთქმედებას გამოყენებითი შემთხვევათა სცენარების რეალიზაციას, თავსდება კოოპერაციაში შესაბამისი გამოყენებითი შემთხვევის სახელწოდებით და სტერეოტიპით <<use case realization>>. ყველა კოოპერაციები თავსდებიან პაკეტში დასახელებით Use Case Realizations, რომელიც შედის პაკეტში Design Model.

მესამე ეტაპზე პროექტირების პროცესში ანალიზის კლასები გარდაიქმნებიან დაპროექტების კლასებში (design klass) და მოთავსდებიან პაკეტში Design Model. მოცემული პაკეტი შეიცავს ასევე არქიტექტურული დონეების, ქვესისტემების და მართვის ნაკადების სტრუქტურების აღწერას.

კომპონენტების თვალთახედვით წარმოდგენა შეიცავს:

- კომპონენტებს, რომლებიც წარმოადგენენ კოდის ფიზიკურ მოდულებს;
- კომპონენტების დიაგრამას, რომლებიც აღწერენ კომპონენტებს და მათ შორის კავშირებს. კავშირები სისტემის კომპონენტებს შორის აღწერენ დამოკიდებულებას, რომელიც არიძვრება კომპილაციის დროს.
- პაკეტებს, რომლებიც შეიცავენ დაკავშირებული კომპონენტების ჯგუფებს.

წარმოდგენა განლაგების სახით განსაზღვრავს სისტემის ფიზიკურ არქიტექტურას, რომელიც შეიძლება განსხვავდებოდეს მისი ლოგიკური არქიტექტურისაგან.

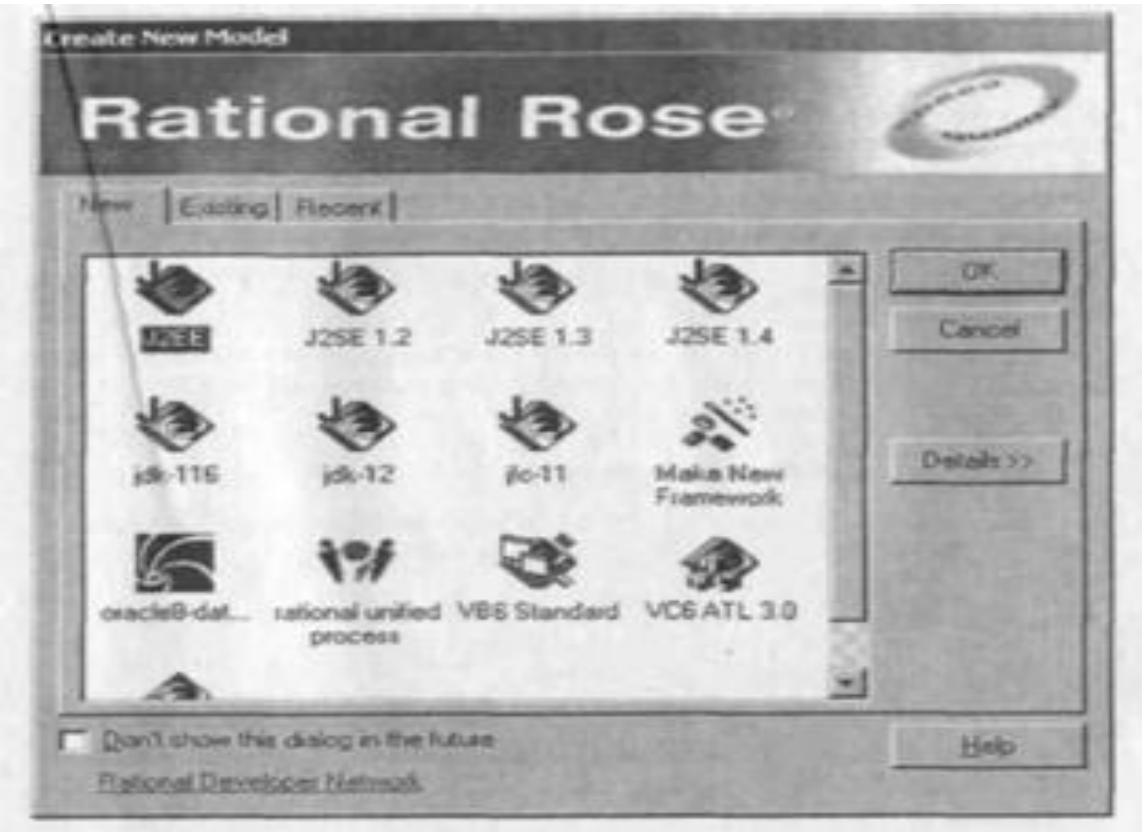
განლაგების წარმოდგენაში შედის:

- პროცესორები, მათ შორის ნებისმიერი კომპიუტერები, რომლებსაც შეუძლიათ მონაცემების დამუშავება;
- მოწყობილობები – ნებისმიერი აპარატურა, რომელსაც არ შეუძლია მონაცემების დამუშავება, მაგალითად, შეტან-გამოტანის ტერმინალები და პრინტერები;
- განლაგების დიაგრამა, რომელზედაც ნაჩვენებია პროცესორები და ქსელის მოწყობილობები, ასევე მათ შორის ფიზიკური შეერთება.

მოდელების შექმნა. პირველ ნაბიჯს Rose-თან მუშაობისას წარმოადგენს მოდელების შექმნა. ისინი შესაძლებელია ავაგოთ ან ნოლიდან, ან ავიღოთ საფუძვლათ არსებული შაბლონი. Rose-ს შექმნილი მოდელი მთელი თავისი ელემენტებით შესაძლებელია შევინახოთ ერთ ფაილში, რომელსაც აქვს გაფართოება .mdl.

მოდელის შესაქმნელად:

1. ავირჩიოთ მენიუდან პუნქტი File>New.



ნახ.1.4. შაბლონების ოსტატის ფანჯარა

2.თუ დაყენებულია შაბლონების ოსტატი (Framework Wizard), ეკრანზე გამოჩნდება ხელმისაწვდომი შაბლონების ცხრილი (ნახ.1.4.). ავირჩიოთ შაბლონი და დავაწკაპუნოთ ღილაკზე OK. თუ გათვალისწინებული არ არის შაბლონით მუშაობა, დავაწკაპუნოთ ღილაკზე Cancel.

შაბლონის არჩევის შემდეგ Rose ავტომატურათ ჩატვირთავს მოცემული შაბლონისათვის დადგენილ პაკეტებს, კლასებს და კომპონენტებს.

მოდელების შენახვა. ისევე როგორც სხვა დანართების შემთხვევაში, საჭირო ხდება პერიოდულათ შევინახოთ ფაილები მათთან მუშაობის პროცესში. მთელი მოდელი შეინახება ერთ ფაილში, ასევე ცალკე ფაილში შეიძლება შევინახოთ ჟურნალი.

მოდელის შესანახათ ავირჩიოთ მენიუში პუნქტი File>Save ან დავაწკაპუნოთ ღილაკზე Save ინსტრუმენტების სტანდარტულ პანელზე.
ჟურნალის შენახვისათვის:

1. გამოვყოთ ჟურნალის ფანჯარა;
2. გამოვყოთ მენიუში პუნქტი File> Save Log As ან დავაწკაპუნოთ ღილაკზე Save ინსტრუმენტების სტანდარტულ პანელზე.
3. მიუთითეთ ჟურნალის დასახელება.

Rose-ს მოდელის ელემენტებში ფაილებზე მიმართვის და URL მისამართების დამატება. სხვადასხვა მონაცემები, მოდელის გარეთ, მაგალითად, დოკუმენტი, სისტემისადმი მოთხოვნები ან გამოყენებით შემთხვევათა სპეციფიკაციები, შესაძლებელია დაუკავშიროთ Rose-ს მოდელის განსაზღვრულ ელემენტებს.

მოდელის ელემენტებთან ფაილების ან URL მისამართის მიერთებისათვის:

1. დავაწკაპუნოთ მარჯვენა ღილაკით ბრაუზერში საჭირო ელემენტზე;
2. შევასრულოთ New>File ან New>URL;
3. მიუთითოდ საჭირო ფაილი ან URL მისამართი ფაილების ფანჯარაში.

4. დავაწკაპუნოთ ორჯერ მისამართის ან ფაილის ნიშნულზე მისი გახსნისათვის;
5. მას შემდეგ რაც მოხდება ფაილის ან მისამართის დაკავშირება, დავაწკაპუნოთ მარჯვენა ღილაკით ფაილის ან URL მისამართის ნიშნულზე ბრაუზერში და ავირჩიოთ Delete, იმისათვის, რომ მოვსპოთ ფაილი ან URL მისამართი. ამ ოპერაციის შედეგათ ისპობა კავშირი მოდელსა და ფაილს(მისამართს) შორის, მაგრამ თვითონ ფაილი რჩება სისტემაში.

მოდელის ელემენტების ამოგდება. არსებობს ორი საშუალება ამოვაგდოთ მოდელის ელემენტი – ერთი დიაგრამიდან ან მთელი მოდელიდან.

იმისათვის, რომ ამოვაგდოთ მოდელის ელემენტი დიაგრამიდან:

1. გამოვყოთ ელემენტი დიაგრამაზე.
2. დავაჭიროთ კლავიშზე Delete.
3. მოდელის ელემენტი დიაგრამიდან ამოვარდება, მაგრამ რჩება ბრაუზერში და სისტემის სხვა დიაგრამებზე.

ელემენტის მოდელიდან გაძევებისათვის:

1. გამოვყოთ ელემენტი დიაგრამაზე.
2. ამოვირჩიოთ მენიუში პუნქტი Edit > Delete from model ან დავაჭიროთ CTRL-D.

გარდა ამისა საშუალება გვეძლევა მოვახდინოთ მოდელის და მისი ელემენტების ექსპორტირება ან მოვახდინოთ ამა თუ იმ მოდელის ელემენტის იმპორტირება მისი შემდგომი ხელმეორედ გამოყენების მიზნით.

გამოსახულების პარამეტრების დაყენება. გლობალური პარამეტრები, შრიფტი და ფერი გამოიყენება მოდელის ყველა ელემენტებისათვის - კლასებისათვის, გამოყენებითი შემთხვევებისათვის, ინტერფეისების, პაკეტებისა და ა.შ. მნიშვნელობა ამ პარამეტრებისათვის დგინდება დუმილით მენიუში Tools>Options.

Rose-ს გარემოში დასაშვებია შრიფტის შეცვლა, დაუნიშნოთ ელემენტს ახალი შრიფტი, ხაზის ფერი. ამისათვის მენიუში უნდა ავკრიფოთ Font ან Format და ამოვირჩიოთ შესაბამისი პუნქტი.
ასევე არის შესაძლებლობა დავაყენოთ კლასების დიაგრამა ისე, რომ:

- უჩვენოთ ყველა ოპერაციები და ატრიბუტები;
- დავმალოთ ოპერაციები;
- დავმალოთ ატრიბუტები;
- უჩვენოთ მხოლოდ ზოგიერთი ატრიბუტი ან ოპერაცია;
- უჩვენოთ ოპერაციები მთელი მათი სიგნატურით ან მხოლოდ მათი სახელები;
- უჩვენოთ ან არა ატრიბუტებისა და ოპერაციების ხედვა;
- უჩვენოთ ან არა ატრიბუტებისა და ოპერაციების სტერეოტიპები.

ამისათვის მენიუში უნდა ავირჩიოთ Tools>Options და შემდეგ შესაბამისი პუნქტი (Show All Atributes, Show All Operations და ა. შ).

3.საკონტროლო კითხვები

- როგორი წარმოდგენებით გამოისახება მოდელი Rose-ში;
- რისგან შესდგება გამოყენებით შემთხვევათა წარმოდგენა;
- რისგან შესდგება ლოგიკური წარმოდგენა;
- როგორ ხდება ლოგიკური წარმოდგენის შევსება;
- რისგან შესდგება წარმოდგენა კომპონენტების თვალთახედვით;
- რისგან შესდგება წარმოდგენა განლაგების თვალთახედვით;
- როგორ მოვახდინოთ მოდელების შექმნა;
- როგორ მოვახდინოთ მოდელების შენახვა;
- რისთვის არის საჭირო Rose-ს მოდელის ელემენტებში ფაილებზე მიმართვის და URL მისამართების დამატება;

- გამოსახულების რა პარამეტრების დაყენება შეგვიძლია;
- როგორ მოვახდინოთ მოდელის ელემენტის ამოგდება.

4.დავალება

- შევქმნათ მოდელი შაბლონების ოსტატის გამოყენებით, ავირჩიოთ Rational Unified Process-ის შესაბამისი მოდელის სტრუქტურა.
- Rational Unified Process-ის შესაბამისი მოდელის სტრუქტურის გაცნობა, მოდელის სხვადასხვა წარმოდგენების გაცნობა.

ლაბორატორიული სამუშაო №3

ბიზნეს-პროცესების მოდელირება. გამოყენებით შემთხვევათა მოდელის შექმნა ბიზნეს – პროცესებისათვის

1.სამუშაოს მიზანი

სამუშაოს მიზანია შევისწავლოთ:

- ბიზნეს-პროცესების მოქმედი პირების შექმნა Rose გარემოში;
- გამოყენებითი შემთხვევების შექმნა ბიზნეს-პროცესებისათვის;
- გამოყენებითი შემთხვევების დიაგრამის შექმნა ბიზნეს-პროცესებისათვის;
- გამოყენებითი შემთხვევების სპეციფიკაცია;
- ფაილის მიბმა გამოყენებით შემთხვევასთან;

2.მეთოდური მითითებები ლაბორატორიული სამუშაოს ჩასატარებლად

სისტემაში მოდელების წარმოდგენის თვალსაჩინოებისათვის განვიხილავთ პრაქტიკულ ამოცანას, რომელიც დაკავშირებულია უმაღლეს

სასწავლებლებში სტუდენტების დამატებით ფასიან კურსებზე რეგისტრაციის სისტემის დამუშავებასთან[იხ.დანართი 2].

ბიზნეს-პროცესების მოდელირება ითვალისწინებს ორი მოდელის აგებას:

- ბიზნეს-პროცესისათვის გამოყენებითი შემთხვევების მოდელი (Busines Use Case Model);
- ბიზნეს-ანალიზის მოდელი (Busines Analysis Model).

Busines Use Case მოდელი – ეს მოდელია, რომელიც აღწერს ორგანიზაციის ბიზნეს პროცესებს როლებისა და მათი მოთხოვნების ტერმინებში. იგი წარმოადგენს UML-ის გამოყენებითი შემთხვევების მოდელის გაფართოებას სტერეოტიპების შემოტანის ხარჯზე – Busines actor (მოქმედი პირის სტერეოტიპი) და Busines Use Case (გამოყენებითი შემთხვევის სტერეოტიპი).

Busines actor (ბიზნეს-პროცესის მოქმედი პირი) - ეს გარკვეული გარეშე როლია, ორგანიზაციის ბიზნეს-პროცესებთან დაკავშირებით. ბიზნეს-პროცესების მოქმედი პირების პოტენციალური კანდიდატები არიან:

- აქციონერები, შემკვეთები, მიმწოდებლები, პარტნიორები, პოტენციური კლიენტები;
- მართვის ადგილობრივი ორგანოები;
- ორგანიზაციის ქვეგანყოფილებების თანამშრომლები, რომელთა მოღვაწეობა არ არის მოდელით მოცული;
- გარე სისტემები.

მოქმედი პირების ცხრილი დგება შემდეგ კითხვებზე პასუხის გაცემის შედეგათ:

- ვინ იღებს სარგებელს ორგანიზაციის არსებობიდან.
- ვინ ეხმარება ორგანიზაციას განახორციელოს თავისი მოღვაწეობა.
- ვის გადასცემს ორგანიზაციას ინფორმაციას და ვისგან ღებულობს.

მოქმედი პირების სწორი გამოვლენისათვის საჭიროა პირველ რიგში განისაზღვროს განსახილველი ორგანიზაცია ან მოღვაწეობის სფერო. მოცემულ შემთხვევაში ასეთი ორგანიზაციის როლს ასრულებს დეკანატი, რომელიც პაცუხიმგებელია სტუდენტების კურსებზე რეგისტრაციაზე, შესაბამისად, ბიზნეს-პროცესების მოქმედი პირები არიან:
სტუდენტი – ეწერება კურსებზე;
პროფესორი – ირჩევს კურსებს სასწავლებლად.

Rose-ს გაშვებისას ფანჯარაში “Create New Model” ავირჩიოთ შაბლონი “Rational Unified Process”. იმისათვის, რომ მოვათავსოთ მოქმედი პირი ბრაუზერში:

1. ბრაუზერში წარმოდგენისათვის გავხსნათ Use Case View დავაწკაპუნოთ მარჯვენა ღილაკით პაკეტზე Busines Use Case Model.
2. გაღებულ მენიუში ავირჩიოთ პუნქტი New>Actor. ბრაუზერში გამოჩნდება ახალი მოქმედი პირი დასახელებით NewClass.
3. გამოვყოთ ახალი მოქმედი პირი და შევიტანოთ მისი დასახელება.
4. დავაწკაპუნოთ მარჯვენა ღილაკით მოქმედ პირზე.
5. მენიუში ავირჩიოთ პუნქტი Open Specification.
6. ავირჩიოთ სტერეოტიპი Busines Actor და დავაწკაპუნოთ ღილაკზე OK.
7. შევინახოთ მოდელი დასახელებით Coursereg მენიუს პუნქტით File>Save მოქმედი პირების შექმნის შემდეგ.

Busines Use Case (გამოყენებითი შემთხვევა ბიზნეს-პროცესების თვალთახედვით) განისაზღვრება როგორც მიმდევრობითი მოქმედებების თანმიმდევრობა რომელიღაც ბიზნეს-პროცესის ფარგლებში, რომლებსაც მოაქვთ მნიშვნელოვანი შედეგი კონკრეტული მოქმედი პირისათვის.

მოცემული მეთოდიკით ყურადღება მახვილდება ელემენტარულ ბიზნეს-პროცესებზე. ელემენტარული ბიზნეს-პროცესი შესაძლებელია განვსაზღვროთ როგორც ამოცანა, რომელიც სრულდება ერთი ადამიანით ერთ ადგილზე ერთდაიმავე დროს გარკვეული მოვლენის პასუხად,

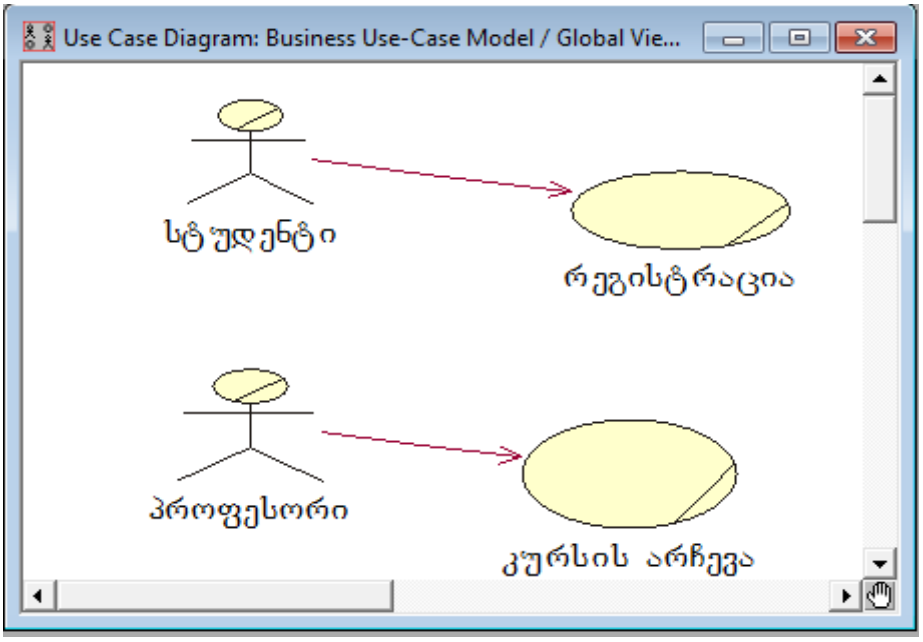
რომელსაც მოაქვს კონკრეტული შედეგი და გადაყავს მონაცემები გარკვეულ მდგრად მდგომარეობაში (მაგალითად, საკრედიტო ბარათით გადახდის დადასტურება). ასეთი ამოცანის გადაწყვეტა მოიცავს 5 დან 10 ბიჯამდე და შესაძლებელია დაიკავოს რამოდენიმე წუთიდან რამოდენიმე დღემდე, მაგრამ განიხილება როგორც ერთი სეანსი მოქმედი პირისა შემსრულებლებთან.

ყოველი Busines Use Case ასახავს მიზანს ან მოთხოვნას გარკვეული მოქმედი პირის. გამომდინარე მოქმედი პირების (სტუდენტები და პროფესორები) მოთხოვნებიდან, შესაძლებელია გამოვყოთ შემდეგი გამოყენებითი შემთხვევები: დავრეგისტრირდეთ კურსებზე და ამოვირჩიოთ კურსები სწავლებისათვის.

იმისათვის, რომ მოვათავსოთ გამოყენებითი შემთხვევა ბრაუზერში ვახდენთ იგივე მოქმედებებს რასაც მოქმედი პირების შექმნისას, მხოლოდ იმ განსხვავებით, რომ Actor-ის ნაცვლად ცხადია უნდა ავირჩიოთ Use Case.

მიღებული ელემენტების საფუძველზე საშუალება გვეძლევა შევქმნათ გამოყენებითი შემთხვევების დიაგრამა ბიზნეს მოდელისათვის დეკანატი. გამოყენებითი შემთხვევების დიაგრამის აგება ბიზნეს მოდელისათვის:

1. გავხსნათ პაკეტი Busines Case Model გამოყენებითი შემთხვევების წარმოდგენაში, დავაწკაპუნოთ რა მარცხენა ღილაკით ნიშნულზე “+” მისგან მარცხნივ.
2. დავაწკაპუნოთ ორჯერ დიაგრამის დასახელებაზე Global View of Busines Actions and bisines Use Case ბრაუზერში, რათა გავხსნათ იგი.
3. გადმოვიტანოთ ბრაუზერიდან მოქმედი პირი ან გამოყენებითი შემთხვევა დიაგრამაზე.



ნახ.3.1. ბიზნეს-მოდელის გამოყენებით შემთხვევათა მოდელი

4. მიუთითოთ ასოციაცია მოქმედ პირსა და გამოყენებით შემთხვევას შორის ღილაკით Unidirectional association (ერთმიმართულებიანი ასსოციაცია) ინსტრუმენტების პანელიდან.

Busines Use Case-ს აღწერა წარმოადგენს სპეციფიკაციას, რომელიც შედგება შემდეგი პუნქტებისაგან:

- დასახელება;
- მოკლე დახასიათება;
- მიზნები და შედეგები (მოქმედი პირის თვალთახედვით);
- სცენარების აღწერა (ძირითადის და ალტერნატიულის);
- სპეციალური მოთხოვნები (შეზღუდვები დროში ან სხვა რესურსებში);
- გაფართოება (კერძო შემთხვევები);
- კავშირები სხვა Busines Use Case –თან;
- მოღვაწეობის დიაგრამები (სცენარების თვალსაჩინო აღწერისათვის – აუცილებლობის შემთხვევაში).

გამოყენებით შემთხვევებზე აღწერების დამატება:

1. გამოვყოთ ბრაუზერში გამოყენებითი შემთხვევა “კურსებზე დარეგისტრირება”.
2. შევიტანოთ დოკუმენტაციის ფანჯარაში შემდეგი აღწერა ამ გამოყენებითი შემთხვევისათვის “მოცემული Busines Use Case საშუალებას იძლევა დავრეგისტრირდეთ კონკრეტულ კურსებზე მიმდინარე სემესტრში”.
3. შევქმნათ MS WORD – ის მეშვეობით ტექსტური ფაილი გამოყენებითი შემთხვევის ქვემოთ მოყვანილი აღწერით.

გამოყენებითი შემთხვევის “კურსებზე დარეგისტრირება” სპეციფიკაცია

დასახელება:

კურსებზე დარეგისტრირება.

მოკლე აღწერა:

მოცემული გამოყენებითი შემთხვევა საშუალებას აძლევს სტუდენტს დარეგისტრირდეს შემოთავაზებულ კურსებზე მიმდინარე სემესტრში.

ძირითადი სცენარი:

1. სტუდენტი მიდის დეკანატის მოსამსახურესთან და გადასცემს მას შევსებულ ფორმას კურსებზე რეგისტრაციისათვის.
2. დეკანატის მოსამსახურე ადასტურებს ფორმის შევსების სისწორეს.
3. დეკანატის მოსამსახურე ადასტურებს, რომ სტუდენტმა შეასრულა წინასწარი მოთხოვნები ყოველი არჩეული კურსისათვის (გარკვეული კურსების გავლა), ასევე თავისუფალი ადგილების არსებობას.
4. დეკანატის მოსამსახურეს შეყავს სტუდენტი, მის მიერ არჩეულ, ყოველი კურსის სიაში.
5. დეკანატის მოსამსახურე ავსებს სტუდენტის გრაფიკს კურსებზე მიმდინარე სემესტრში და გადასცემს მას სტუდენტს.

ალტერნატიული სცენარი:

2ა. არასწორად არის შევსებული რეგისტრაციის ფორმა.

დეკანატის მოსამსახურე უბრუნებს სტუდენტს ფორმას შეცდომების გასწორებისათვის.

3ა. არ არის შესრულებული წინასწარი მოთხოვნები ან კურსი შევსებულია. თუ დეკანატის მოსამსახურე აღმოაჩენს, რომ სტუდენტს არ შეუსრულებია აუცილებელი წინასწარი მოთხოვნები ან მის მიერ არჩეული კურსი შევსებულია (უკვე ჩაწერილია 10 სტუდენტი), მაშინ იგი სთავაზობს სტუდენტს შეიცვალოს თავისი არჩევანი ან წაიღოს ფორმა და დაუბრუნდეს მას მოგვიანებით.

ფაილის მიბმა გამოყენებით შემთხვევასთან.

1. დავაწკაპუნოთ მარჯვენა ღილაკით გამოყენებით შემთხვევაზე.
2. გახსნილ მენიუში ავირჩიოთ პუნქტი Open Specification.
3. გადავიდეთ ჩანართზე File.
4. დავაწკაპუნოთ მარჯვენა ღილაკით თეთრ ველზე და ამოვირჩიოთ გახსნილ მენიუში პუნქტი Insert File.
5. მიუთითოთ მანამდე შექმნილი ფაილი და დავაწკაპუნოთ ღილაკზე Open, იმისათვის, რომ დაუკავშიროთ ფაილი გამოყენებით შემთხვევას.

3.საკონტროლო კითხვები

- რა არის **Busines Use Case** მოდელი;
- რა არის ბიზნეს-პროცესის მოქმედი პირი;
- რა არის გამოყენებითი შემთხვევა ბიზნეს-პროცესების თვალთახედვით;
- როგორ შევქმნათ გამოყენებითი შემთხვევების დიაგრამა ბიზნეს მოდელისათვის;
- რისგან შესდგება ბისნეს-პროცესის გამოყენებითი შემთხვევების სპეციფიკაცია;

- როგორ მოვახდინოთ გამოყენებით შემთხვევებზე აღწერების დამატება.

4.დავალება

- ავაგოთ გამოყენებითი შემთხვევების დიაგრამა ბიზნეს მოდელისათვის;
- შევქმნათ გამოყენებითი შემთხვევების სპეციფიკაციები, შევინახოთ Word ფაილში და დავაკავშიროთ გამოყენებით შემთხვევასთან.

ლაბორატორიული სამუშაო №4

ბიზნეს-პროცესების მოდელირება. ბიზნეს ანალიზის მოდელის შექმნა

1.სამუშაოს მიზანი

სამუშაოს მიზანია შევისწავლოთ:

- ბიზნეს-პროცესების კლასი-აბსტრაქცია ***Busines Worker*** (შემსრულებელი);
- ბიზნეს-პროცესების კლასი-აბსტრაქცია ***Busines Entity*** (არსი);
- კლასების დიაგრამა ბიზნეს-ანალიზის მოდელისათვის;
- კოოპერაციები.

2.მეთოდური მითითებები ლაბორატორიული სამუშაოს ჩასატარებლად

ყოველი Busines Use Case-თვის იგება ბიზნეს ანალიზის მოდელი – ობიექტური მოდელი, რომელიც აღწერს ბიზნეს – პროცესების

რეალიზაციას ურთიერთმოქმედი ობიექტებით (ბიზნეს-ობიექტები – Busines Object), რომლებიც ეკუთვნიან ორ კლასს – Busines Worker და Busines Entity.

Busines Worker (შემსრულებელი) – აქტიური კლასი, რომელიც წარმოადგენს შემსრულებლის აბსტრაქციას, რომელიც ასრულებს გარკვეულ მოქმედებებს ბიზნეს-პროცესის ფარგლებში. შემსრულებლები ურთიერთქმედებენ ერთმანეთს შორის და მანიპულირებენ სხვადასხვა არსებებით, მონაწილეობენ რა Business Use Case-ს რეალიზაციაში. UML კლასების დიაგრამაზე შემსრულებელი წარმოიდგინება კლასის სახით სტერეოტიპით <<business worker>>.

Business Entity (არსი) - პასიური კლასი, რომელიც არავითარი ურთიერთქმედების ინიცირებას არ ახდენს. ასეთი კლასის ობიექტი შესაძლებელია მონაწილეობდეს სხვადასხვა Business Use Case-ს რეალიზაციაში. არსი წარმოადგენს სხვადასხვა მოქმედებების ობიექტს შემსრულებლების მხრიდან. UML კლასების დიაგრამაზე შემსრულებელი წარმოიდგინება კლასის სახით სტერეოტიპით <<business entity>>.

ბიზნეს ანალიზის მოდელი შესაძლებელია შედგებოდეს სხვადასხვა ტიპის დიაგრამებისაგან. მოდელის შემადგენლობაში აუცილებლად უნდა შედიოდეს კლასების დიაგრამა, რომელიც შეიცავს შემსრულებლებს და არსებს.

პრობლემის ფორმულირებით, შემსრულებლის როლს ასრულებს დეკანატის თანამშრომელი (უწოდოთ “რეგისტრატორი”), რომელიც ახდენს სასწავლო გეგმის და კურსების კატალოგის ფორმირებას, წერს სტუდენტებს კურსებზე, აწარმოებს ყველა მონაცემებს კურსების, პროფესორების და სტუდენტების შესახებ. არსები, რომლებითაც ის მანიპულირებს, არის:

- სტუდენტი;
- პროფესორი;
- სტუდენტის გრაფიკი (კურსების ცხრილი);

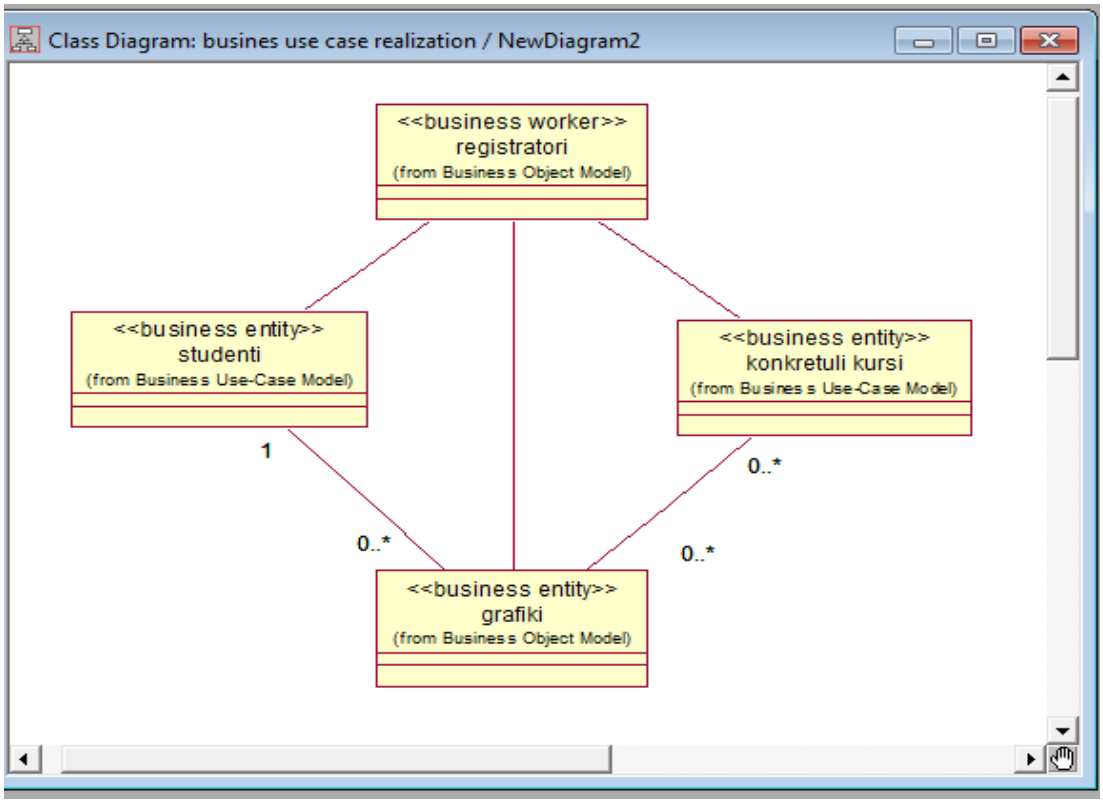
- კურსი (სწავლების პროგრამაში);
- შეთავაზებული კურსი (კურსი განრიგში).

Busines Use Case-თვის “კურსებზე დარეგისტრირება” არსების ცხრილი იქნება შემდეგი:

- სტუდენტი;
- სტუდენტის გრაფიკი;
- შეთავაზებული გრაფიკი.

კლასების დიაგრამა ბიზნეს-ანალიზის მოდელისათვის, რომელიც აღწერს Busines Use Case-ს “კურსებზე დარეგისტრირება”, მოყვანილია ნახ.4.1. (მოცემული კლასებისათვის გამოყენებულია სტერეოტიპის გამოსახულება ნიშნულის Label სახის). სტერეოტიპის გამოსახულების დაყენება შესაძლებელია შესრულდეს შემდეგნაირად:

- მთელი მოდელისათვის – მენიუში Tools >Options >Diagram > Stereotype Display.
- მოდელის ცალკეული ელემენტისათვის – მის კონტექსტურ მენიუში Options > Stereotype Display.
- მოდელის რამოდენიმე დაჯგუფებული ელემენტებისათვის – მენიუში Format > Stereotype display.



ნახ.4.1. ბიზნეს-ანალიზის მოდელის კლასების დიაგრამა

კლასების შექმნა, რომლებიც მონაწილეობენ ბიზნეს-პროცსის რეალიზაციაში "კურსებზე დარეგისტრირება", და კოოპერაციების, რომლებიც აღწერენ ბიზნეს-პროცესის რეალიზაციას

კლასების, კოოპერაციების და კლასების დიაგრამების შექმნისათვის:

1. დავაწკაპუნოთ მარჯვენა ღილაკზე ბრაუზერში წარმოდგენაში Logical View პაკეტზე Busines Object Model.
2. გახსნილ მენიუში ამოვირჩიოთ პუნქტი New > Class. ახალი კლასი სახელწოდებით NewClass გამოჩნდება ბრაუზერში.
3. გამოვყოთ იგი და შევიტანოთ სახელი "რეგისტრატორი".
4. დავაწკაპუნოთ მარჯვენა ღილაკით მოცემულ კლასზე.
5. ავირჩიოთ გახსნილ მენიუში პუნქტი Open Specification.
6. ავირჩიოთ Busines Worker სტერეოტიპების ველში და დავაწკაპუნოთ OK.

7. ანალოგიურად შევქმნათ არსება-კლასები სტერეოტიპით <<Busines Entity>>.
8. დავაწკაპუნოთ მარჯვენა ღილაკზე ბრაუზერში წარმოდგენაში Logical View პაკეტზე Busines Object Model.
9. ავირჩიოთ გახსნილ მენიუში პუნქტი New > Package.
10. დავარქვათ ახალ პაკეტს Busines Use Case Realizations.
11. შევქმნათ კოოპერაცია "კურსებზე დარეგისტრირება" (კოოპერაცია წარმოადგენს გამოყენებით შემთხვევას სტერეოტიპით "business use-case realization", რომელიც მიეთითება გამოყენებითი შემთხვევის სპეციფიკაციაში) პაკეტში Busines Use Case Realizations.
12. დავაწკაპუნოთ მარჯვენა ღილაკით შექმნილ კოოპერაციაზე.
13. გახსნილ მენიუში ავირჩიოთ პუნქტი New > Class Diagram.
14. დავარქვათ კლასების ახალ დიაგრამას Participating Classes.
15. გავხსნათ კლასების დიაგრამა Participating Classes და გადავიტანოთ კლასები გახსნილ დიაგრამაზე ნახ.4.1. –ის შესაბამისად.

იმისათვის, რომ კლასების დიაგრამაზე შევქმნათ ასოციაცია:

1. დავაწკაპუნოთ ინსტრუმენტების პანელიზე ღილაკზე Assotiation (Rose-ს დაყენების შემდეგ აღნიშნული ღილაკი პანელიზე არ არის ნაჩვენები. გამოყვანისათვის საჭიროა პანელის დაყენება).
2. გავატაროთ თაგვით ხაზი ერთი კლასიდან მეორეზე.

კავშირის სიმძლავრის მითითებისათვის:

1. დავაწკაპუნოთ მარჯვენა ღილაკით კავშირის ერთ ბოლოზე.
2. გახსნილ კონტექსტურ მენიუში ავირჩიოთ Multiplicity.
3. მიუთითოთ საჭირო სიმძლავრე.
4. გავიმეოროთ პ. 1-3 კავშირის მეორე ბოლოსათვის.
5. გავხსნათ კავშირის სპეციფიკაცია(დავაწკაპუნოთ რა მასზე ორჯერ მარცხენა ღილაკით). და დავაყენოთ სიმძლავრის ზუსტი მნიშვნელობა Multiplicity ველში ჩანართი "Role A detail" ან "Role B detail", თუ

სიმძლავრის მოთხოვნილი მნიშვნელობა არ ემთხვევა სტანდარტულ მნიშვნელობებს კავშირის კონტექსურ მენიუში.

კლასების დიაგრამის გარდა, ბიზნეს ანალიზის მოდელი შეიძლება შეიცავდეს:

- მოღვაწეობის დიაგრამებს ობიექტების ნაკადებით და ბილიკებით, რომლებიც აღწერენ ურთიერკავშირს ერთ ან სხვადასხვა Busines Use Case-ს სცენარებს შორის.
- მიმდევრობის დიაგრამებს, რომლებიც აღწერენ Busines Use Case-ს სცენარებს შეტყობინებების გაცვლის თანმიმდევრობით ობიექტ-მოქმედ პირებსა და ობიექტ- შემსრულებლებს შორის. ასეთი დიაგრამები გვეხმარებიან ნათლად განვსაზღვროთ მოდელში თითოეული შემსრულებლის მოვალეობა მისი ოპერაციების ნაკრებით.
- მდგომარეობის დიაგრამებს, რომლითაც აღიწერება ცალკეული ბიზნეს-ობიექტების ქცევა.

3.საკონტროლო კითხვები

- რა არის კლასი-აბსტრაქცია ***Busines Worker*** (შემსრულებელი);
- რა არის კლასი-აბსტრაქცია ***Busines Entity*** (არსი);
- როგორ ავაგოთ კლასების დიაგრამა ბიზნეს-ანალიზის მოდელისათვის;
- რა არის კოოპერაცია და როგორ ავაგოთ ბიზნეს-ანალიზის მოდელისათვის.

4.დავალება

- ავაგოთ კლასების დიაგრამა ბიზნეს-ანალიზის მოდელისათვის.
- შევქმნათ კოოპერაციები ბიზნეს-ანალიზის მოდელისათვის

ლაბორატორიული სამუშაო №5

მოღვაწეობის დიაგრამების შექმნა

1.სამუშაოს მიზანი

სამუშაოს მიზანია შევისწავლოთ:

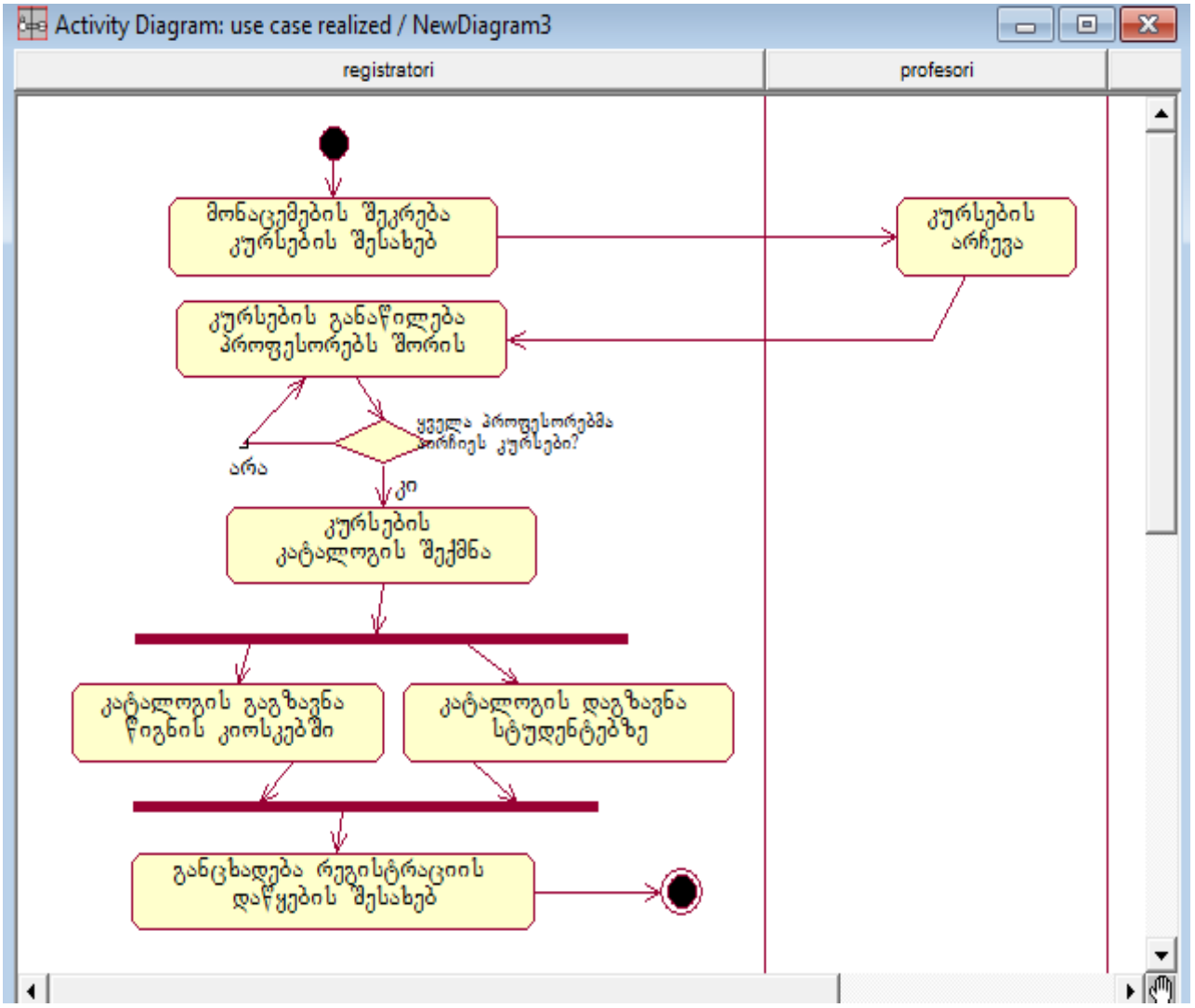
- მოღვაწეობის დიაგრამის ასაგები ფანჯრის გამოყვანა;
- ბილიკების დამატება;
- საწყისი(ბოლო) მდგომარეობების დამატება;
- გადასვლებისშექმნა მოღვაწეობებს შორის;
- გადაწყვეტილების მიღების წერტილის დამატება;
- გადასვლის პირობების დამატება;
- სინქრონიზაციის ხაზების დამატება.

2.მეთოდური მითითებები ლაბორატორიული სამუშაოს ჩასატარებლად

კლასების დიაგრამის გარდა, ბიზნეს ანალიზის მოდელი შეიძლება შეიცავდეს მოღვაწეობის დიაგრამებს. კურსების კატალოგის და მათი სტუდენტებზე განაწილებისათვის შესაძლებელია ავაგოთ მოღვაწეობის შემდეგი დიაგრამა(ნახ.5.1.). მოღვაწეობის დიაგრამის დამატებისათვის:

1. დავაწკაპუნოთ მარჯვენა ღილაკით პაკეტზე Busines Use Case Realizations ბრაუზერში.
2. მენიუში ავირჩიოთ New > Activity Diagram.
3. Rose შექმნის ბრაუზერში ელემენტს State/Activity Model, რომელიც იმყოფება პაკეტის ქვევით.
4. მივანიჭოთ სახელი მოღვაწეობის ახალ დიაგრამას.
5. დავაწკაპუნოთ ორჯერ ამ დიაგრამაზე, იმისათვის რომ გავხსნათ იგი.

მოღვაწეობის შექმნილ დიაგრამაზე ინსტრუმენტების პანელის მეშვეობით დავამატოთ ბილიკები, მოღვაწეობები და სხვა ობიექტები.



ნახ.5.1.

დიაგრამაზე ბილიკების დამატებისათვის:

1. ინსტრუმენტების პანელიზე ავირჩიოთ ღილაკი Swimlane.
2. დავაწკაპუნოთ დიაგრამის შიგნით. გამოჩნდება ახალი ბილიკი სახელწოდებით NewSwimlane.
3. გავხსნათ ბილიკის სპეციფიკაცია, დავაწკაპუნოთ რა მარჯვენა ღილაკით დასახელებაზე და ავირჩიოთ პუნქტი Open Specification.
4. მივანიჭოთ ბილიკს ახალი დასახელება, ამისათვის ველი Name გავასუფთაოთ და ველის ცხრილში Class ავირჩიოთ “რეგისტრატორი” მოდელიდან Busines Use Case(ნახ.5.2.).

5. შევქმნათ კიდევ ერთი ბილიკი დასახელებით “პროფესორი” მოდელიდან Busines Use Case.

ნახ.5.2.

იმისათვის, რომ დავამატოთ დიაგრამაში საწყისი(ბოლო) მდგომარეობა:

1. ინსტრუმენტების პანელზე გამოვყოთ ღილაკი Start (End).
2. დავაწკაპუნოთ დიაგრამის შიგნით და შესაბამისი ბილიკის შიგნით.

იმისათვის, რომ დავამატოთ დიაგრამაში ახალი მოღვაწეობა:

1. ავირჩიოთ ინსტრუმენტების პანელზე ღილაკი Activity.
2. დავაწკაპუნოთ დიაგრამის შიგნით და საჭირო ბილიკის შიგნით, რომელიც განსაზღვრავს მოქმედ პირს ან შემსრულებელს, რომელიც პასუხს აგებს მოღვაწეობის შესრულებაზე.
3. მივანიჭოთ სახელი ახალ მოღვაწეობას.

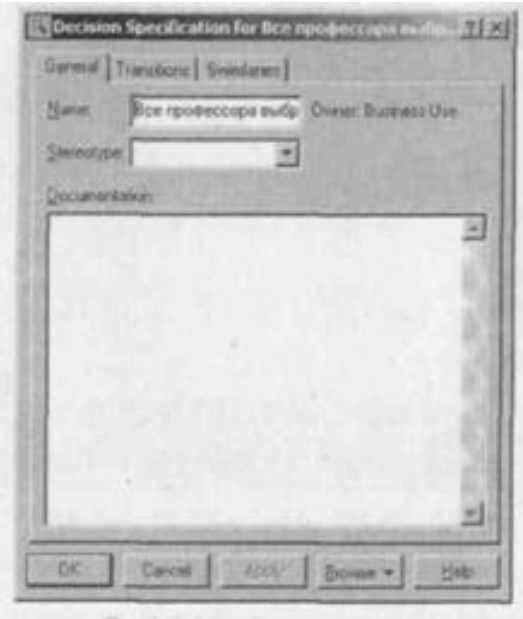
იმისათვის, რომ შევქმნათ გადასვლა მოღვაწეობებს შორის:

1. ავირჩიოთ ინსტრუმენტების პანელზე ღილაკი State Transition.
2. გადავიტანოთ თაგვის მაჩვენებელი ერთი მოღვაწეობიდან მეორეზე.

იმისათვის, რომ დავამატოთ გადაწყვეტილების მიღების წერტილი:

1. ავირჩიოთ ინსტრუმენტების პანელზე ღილაკი Decision.

2. დავაწკაპუნოთ დიაგრამის შიგნით გადაწყვეტილების ჩასმისათვის.
3. გავხსნათ სპეციფიკაციის ფანჯარა გადაწყვეტილებისათვის და შევიტანოთ მისი დასახელება (ნახ.5.3.).

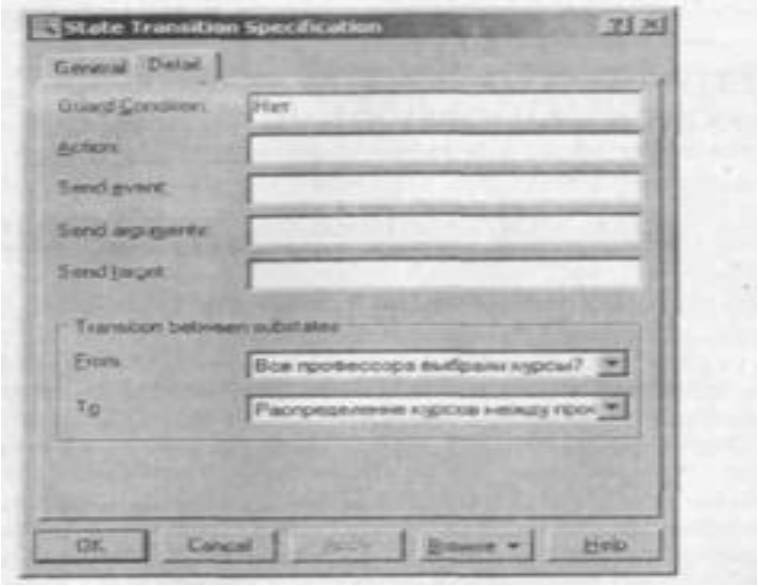


ნახ.5.3. გადაწყვეტილების სპეციფიკაცია

4. დავხაზოთ გადასვლა მოღვაწეობიდან გადაწყვეტილებისაკენ ან გადაწყვეტილებიდან ერთ ან რამოდენიმე მოღვაწეობისაკენ.

იმისათვის, რომ დავამატოთ გადასვლის პირობები:

1. დავაწკაპუნოთ მარჯვენა ღილაკით გადასვლაზე.
2. ავირჩიოთ Open Specification – გაიხსნება სპეციფიკაციის ფანჯარა გადასვლისათვის.
3. გადავიდეთ ჩანართზე “Details” (ნახ.5.4.).
4. შევიტანოთ პირობა ველში Guard Condition (შესაძლებელია შევიტანოთ პირობა უშუალოდ გადასვლის ისარზე, მოვათავსოთ რა ეს პირობა კვადრატულ ფრჩხილებში).



ნახ.5.4. გადასვლების სპეციფიკაცია

იმისათვის, რომ დავამატოთ სინქრონიზაციის ხაზები:

1. ავირჩიოთ პანელზე Vertical Synchronization ან Horizontal Syncronization.
2. დავაწკაპუნოთ დიაგრამის შიგნით მათი ჩართვისათვის.
3. დავხაზოთ გადასვლები სინქრონიზირებულ მოღვაწეობებს შორის.

3.საკონტროლო კითხვები

- როგორ მვახდინოთ მოღვაწეობის დიაგრამის ასაგები ფანჯრის გამოყვანა ROSE გარემოში;
- როგორ ხდება ბილიკების საწყისი(ბოლო) მდგომარეობების დამატება;
- როგორ ხდება გადასვლებისშექმნა მოღვაწეობებს შორის;
- როგორ ხდება გადაწყვეტილების მიღების წერტილის დამატება;
- როგორ ხდება გადასვლის პირობების დამატება;
- როგორ ხდება სინქრონიზაციის ხაზების დამატება.

4.დავალება

- ავაგოთ მოღვაწეობის დიაგრამა ბიზნეს-ანალიზის მოდელისათვის.

ლაბორატორიული სამუშაო №6

პროგრამული უზრუნველყოფისადმი მოთხოვნების სპეციფიკაცია

1.სამუშაოს მიზანი

სამუშაოს მიზანია შევისწავლოთ:

- საპრობლემო სფეროს ლექსიკონი;
- დამატებითი სპეციფიკაციები;
- გამოყენებით შემთხვევათა საწყისი ვერსიის მოდელის შექმნა;
- მოქმედი პირების შექმნა ROSE გარემოში;
- გამოყენებითი შემთხვევების შექმნა ROSE გარემოში;
- გამოყენებით შემთხვევათა დიაგრამის საწყისი ვერსიის აგება.
- აღწერების დამატება გამოყენებით შემთხვევებზე;
- პრიორიტეტის დანიშვნა გამოყენებით შემთხვევაზე.

2.მეთოდური მითითებები ლაბორატორიული სამუშაოს ჩასატარებლად

მოთხოვნები პროგრამული უზრუნველყოფისადმი დოკუმენტირდება რიგი დოკუმენტებით და მოდელებით:

- კონცეფცია, რომლითაც განისაზღვრება პროექტის გლობალური მიზნები და დასამუშავებელი სისტემის ძირითადი თავისებურებები;
- საპრობლემო სფეროს ლექსიკონი, რომლითაც დგინდება ტერმინოლოგია ყველა მოდელისათვის;
- დამატებითი სპეციფიკაციები, რომლითაც კონცეფციისაგან განსხვავებით დგინდება არაფუნქციონალური (ტექნიკური) მოთხოვნები სისტემისადმი, როგორიც არის საიმედობა, წარმადობა, გამოყენების მოხერხებულობა, უშიშროება და პროექტული შეზღუდვები.

ლექსიკონი განკუთვნილია საპრობლემო სფეროს ტერმინოლოგიის აღწერისათვის. იგი შესაძლებელია გმოყენებულ იქნას როგორც სისტემის არაფორმალური *მონაცემთა ლექსიკონი.*

მაგალითისათვის, **დანართში** 3 მოყვანილია ტერმინები და მათი მნიშვნელობა განხილული მაგალითისათვის.

დამატებითი სპეციფიკაციების აღწერა. დამატებითი სპეციფიკაციების დანიშნულებაა განისაზღვროს მოთხოვნები სისტემისადმი, რომლებსაც არ მოიცავს გამოყენებითი შემთხვევების მოდელი. ერთად ისინი ქმნიან სისტემისადმი მოთხოვნების სრულ ნაკრებს.

დამატებითი სპეციფიკაციები განსაზღვრავენ სისტემისადმი არაფუნქციონალურ მოთხოვნებს ისეთის, როგორიც არის გამოყენების მოხერხებულობა, საიმედობა, წარმადობა, ასევე რიგი ფუნქციონალური მოთხოვნებისა, რომლებიც საერთოა რამოდენიმე გამოყენებითი შემთხვევისათვის: უსაფრთხოება, საპროექტო შეზღუდვები.

განხილული მაგალითისათვი დამატებითი სპეციფიკაციები ფორმირდება შემდეგი დებულებების საფუძველზე(იხ.დანართი 2).

სისტემისადმი ფუნქციონალური მოთხოვნები მოდელირდება და დოკუმენტირდება გამოყენებითი შემთხვევების (Use Case) მეშვეობით, რომლებიც ითვალისწინებენ შემდეგს:

- გამოყენებითი შემთხვევა აფიქსირებს შეთანხმებას პროექტის მანაწილეებისა სისტემის ქცევასთან;
- გამოყენებითი შემთხვევა აღწერს სისტემის ქცევას სხვადასხვა პირობებისას, როდესაც სისტემა პასუხობს ერთერთი მონაწილის დაკვეთას, რომელიც იწოდება ძირითად მოქმედ პირად.
- ძირითადი მოქმედი პირი ახდენს სისტემასთან ურთიერთქმედების ინიცირებას, იმისათვის რომ მიღწეულ იქნას გარკვეული მიზანი. სისტემა პასუხობს, იცავს რა ყველა მონაწილის ინტერესებს.

გამოყენებითი შემთხვევების აღწერისას არსებობს სიზუსტის ოთხი დონე:

- მოქმედი პირები და მიზნები (დგინდება ყველა მოქმედი პირები და მათი მიზნები, რომელსაც უზრუნველყოფს სისტემა);
- გამოყენებითი შემთხვევის მოკლე დახასიათება (ერთ სტრიქონში) ან მოვლენათა ძირითადი ნაკადი (მოსალოდნელი შეცდომების ანალიზის გარეშე);
- უარყოფის (შეფერხების) პირობები (მოვლენათა ძირითად ნაკადში მოსალადნელი შეცდომების აღძვრის ადგილების ანალიზი);
- შეფერხებების დამუშავება (მოვლენათა ალტერნატიული ნაკადების აღწერა).

მოთხოვნების სპეციფიკაცია Rational Unified Process – ის ტექნოლოგიით არ ითვალისწინებს ორგანიზაციის ბიზნეს-პროცესების აუცილებელ მოდელირებას, რომლისთვისაც იქმნება პროგრამული უზრუნველყოფა, მაგრამ ბიზნეს მოდელის არსებობა საგრძნობლად ამარტივებს გამოყენებითი შემთხვევის სისტემური მოდელის აგებას. ბიზნეს – მოდელიდან გამოყენებით შემთხვევათა მოდელის საწყის ვერსიაზე გადასვლისას სრულდება შემდეგი წესები:

- ყოველი შემსრულებლისათვის ბიზნეს – ანალიზის მოდელში, რომელიც პერსპექტივაში გახდება ახალი სისტემის მომხმარებელი, გამოყენებით შემთხვევათა მოდელში იქმნება მოქმედი პირი ასეთივე დასახელებით. მოქმედი პირების შემადგენლობაში ჩაირთვება ასევე გარე სისტემები, რომლებიც ბიზნეს – პროცესებში ინფორმაციის წყაროს პასიურ როლს თამაშობენ;
- გამოყენებითი შემთხვევები მოცემული მოქმედი პირისათვის იქმნებიან შესაბამისი შემსრულებლის მოთხოვნათა ანალიზის საფუძველზე (უმარტივეს შემთხვევაში შემსრულებლის ყოველი ოპერაციისათვის იქმნება გამოყენებითი შემთხვევა, რომელიც მოახდენს მოცემული ოპერაციის რეალიზებას სისტემაში).

მოდელის ასეთი საწყისი ვერსია აღწერს სისტემის მინიმალურ ვარიანტს, რომლის მომხმარებლები არიან მხოლოდ ბიზნეს – პროცესების შემსრულებლები. თუ მომავალში სისტემის განვითარებისას მისი უშუალო მომხმარებლები იქნებიან ბიზნეს – პროცესების მოქმედი პირები, გამოყენებით შემთხვევათა მოდელი დაიწყებს მოდიფიცირებას.

მოყვანილი წესების გამოყენებას რეგისტრაციის სისტემისათვის მივყავართ ახალი მოქმედი პირების გამოჩენასთან სისტემის საწყისი ვერსიისათვის:

- რეგისტრატორი – ახდენს სასწავლო გეგმის და კურსების კატალოგის ფორმირებას, წერს სტუდენტებს კურსებზე, მიყავს ყველა მონაცემები კურსების, პროფესორების, მოსწრებისა და სტუდენტების შესახებ.
- საანგარიშსწორებო სისტემა – ღებულობს ინფორმაციას მოცემული სისტემიდან კურსებზე დასწრების ანგარიშსწორების შესახებ;
- კურსების კატალოგი – მონაცემთა ბაზა, რომელიც შეიცავს ინფორმაციას კურსების შესახებ.

მოქმედი პირების შექმნა Rose გარემოში.

1. ბრაუზერში Use Case View წარმოდგენაში დავაწკაპუნოთ მარჯვენა ღილაკით პაკეტზე Actors, რომელიც შედის პაკეტში Use Case Model.
2. გაღებულ მენიუში ავირჩიოთ პუნქტი New>Actor.
3. ბრაუზერში გამოჩნდება ახალი მოქმედი პირი დასახელებით NewClass.
4. გამოვყოთ ახალი მოქმედი პირი და შევიტანოთ მისი დასახელება.

მოქმედი პირების მოთხოვნებიდან გამომდინარე, გამოიყოფა შემდეგი გამოყენებითი შემთხვევები:

- სისტემაში შესვლა;
- კურსებზე დარეგისტრირება;
- მოსწრების ტაბელის ნახვა;
- ავირჩიოთ კურსები სწავლისათვის;

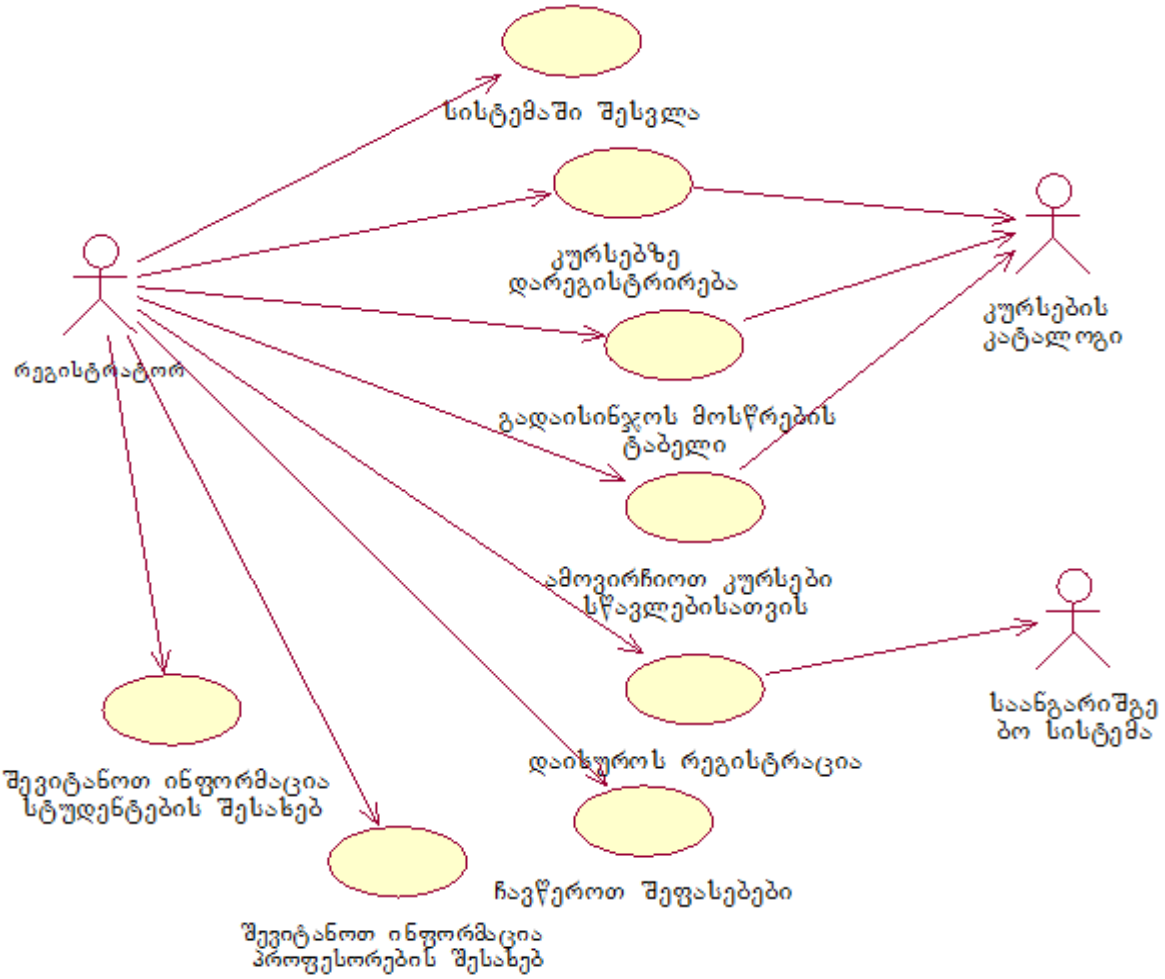
- შეფასებების ნახვა;
- შევიტანოთ ინფორმაცია პროფესორების შესახებ;
- შევიტანოთ ინფორმაცია სტუდენტების შესახებ;
- დავხუროთ რეგისტრაცია.

გამოყენებითი შემთხვევების დიაგრამის საწყისი ვერსია მოყვანილია ნახ.6.1. – ზე.

გამოყენებითი შემთხვევების შექმნა Rose გარემოში.

იმისათვის, რომ მოვათავსოთ გამოყენებითი შემთხვევა ბრაუზერში:

1. ბრაუზერში Use Case View წარმოდგენაში დავაწკაპუნოთ მარჯვენა ღილაკით პაკეტზე Use Case, რომელიც შედის პაკეტში Use Case Model.
2. გაღებულ მენიუში ავირჩიოთ პუნქტი New>Use Case.
3. ბრაუზერში გამოჩნდება ახალი გამოყენებითი შემთხვევა დასახელებით NewUseCase.
4. გამოვყოთ ახალი გამოყენებითი შემთხვევა და შევიტანოთ მისი დასახელება.



ნახ.6.1.

Rose გარემოში გამოყენებითი შემთხვევების დიაგრამები იქმნებიან გამოყენებით შემთხვევათა წარმოდგენაში. მთავარი დიაგრამა(Global View of Busines Actions and bisines Use Case) წარმოდგება დუმილით. შემდეგ სისტემის მოდელირებისათვის შესაძლებელია იმდენი დამატებითი დიაგრამის დამუშავება, რამდენიც საჭიროა.

გამოყენებით შემთხვევათა ახალი დიაგრამის შექმნისათვის:

1. ბრაუზერში წარმოდგენაში Use Case View დავაწკაპუნოთ მარჯვენა ღილაკით პაკეტზე Use Case Model.
2. გაღებულ მენიუში ავირჩიოთ პუნქტი New>Use Case Diagram.
3. გამოვყოთ ახალი დიაგრამა, შევიტანოთ მისი დასახელება.

ბრაუზერში დავაწკაპუნოთ ორჯერ ახალი დიაგრამის დასახელებაზე, იმისათვის, რომ გავხსნათ იგი.

გამოყენებით შემთხვევათა დიაგრამის საწყისი ვერსიის აგება.

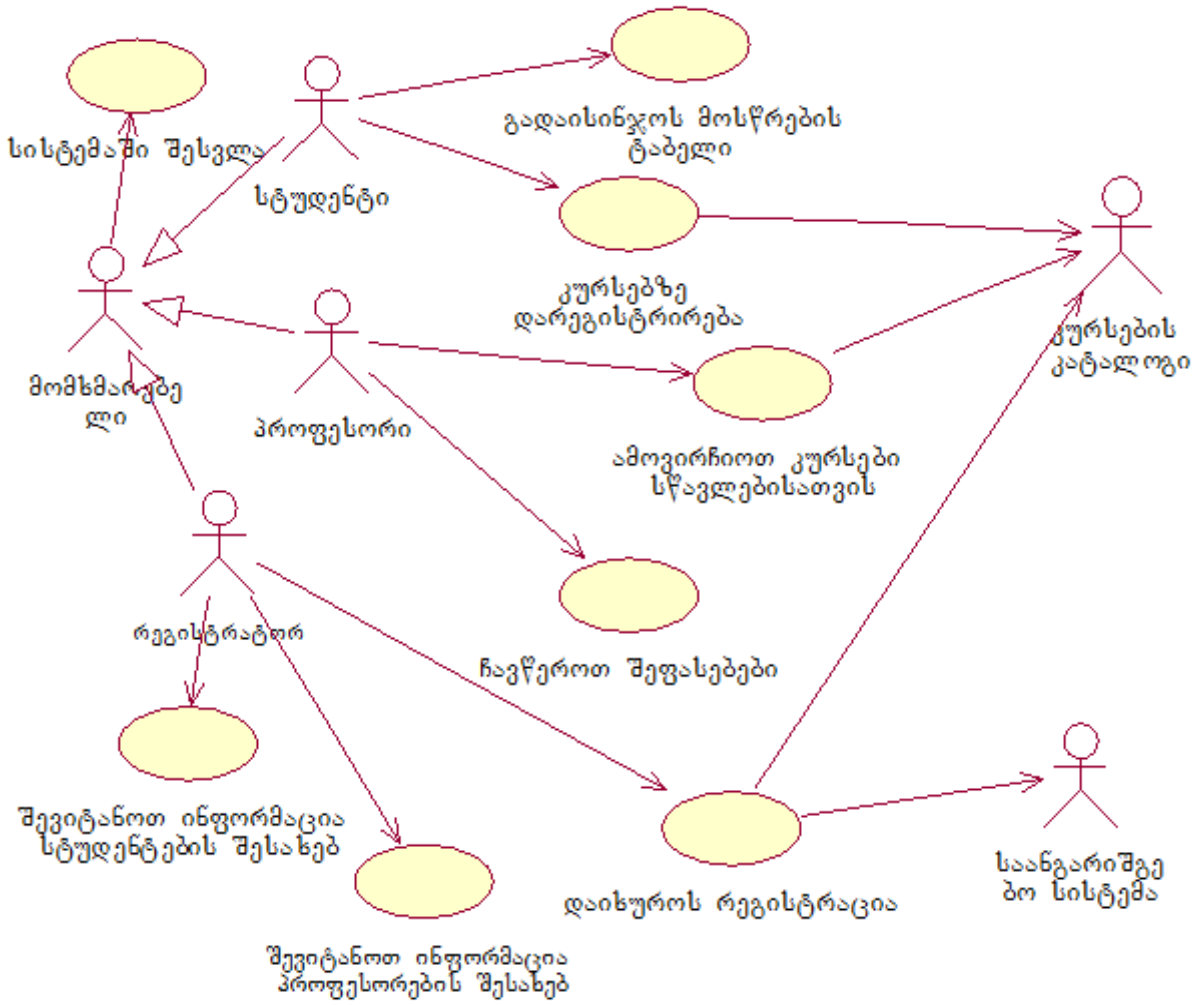
იმისათვის, რომ ავაგოთ გამოყენებით შემთხვევათა დიაგრამის საწყისი ვერსია:

1. დავაწკაპუნოთ ორჯერ დიგრამის დასახელებაზე Global View of Busines Actions and bisines Use Case ბრაუზერში, რათა გავხსნათ იგი.
2. გავასუფთაოთ იგი ყველა ელემენტისაგან.
3. გადმოვიტანოთ ბრაუზერიდან მოქმედი პირი ან გამოყენებითი შემთხვევა დიაგრამაზე.
4. მიუთითოთ ასოციაცია მოქმედ პირსა და გამოყენებით შემთხვევას შორის ღილაკით Unidirectional association (ერთმიმართულებიანი ასსოციაცია) ინსტრუმენტების პანელიდან.

მოთხოვნების დაზუსტებასთან ერთად უნდა განხორციელდეს გამოყენებით შემთხვევათა დიაგრამის მოდიფიცირება. ასე მაგალითად, რეგისტრაციის სისტემაში ამოცანის დასმის თანახმად სისტემის მომხმარებელთა შემადგენლობაში უნდა შევიტანოთ სტუდენტები და პროფესორები. გამოყენებით შემთხვევათა დიაგრამის მოდიფიცირებული ვერსია მოყვანილია ნახ.6.2.-ზე. რამდენადაც სისტემაში შესვლა რეგისტრატორისათვის, სტუდენტებისა და პროფესორებისათვის ერთიდაიგივეა, მათი ქცევა შესაძლებელია განვაზოგადოთ და შემოვიტანოთ ახალი მოქმედი პირი “მომხმარებელი” (სუპერ ტიპი) საერთო გამოყენებითი შემთხვევით “სისტემაში შესვლა”, რომლის ქვეტიპებია რეგისტრატორი, სტუდენტი და პროფესორი. მოქმედი პირები:

- სტუდენტი – ეწერება კურსებზე და ათვალიერებს მოსწრების ტაბელს.
- პროფესორი - ირჩევს კურსებს საწავლებისათვის და სვამს შეფასებებს.

- რეგისტრატორი – ახდენს სასწავლო გეგმის და კურსების კატალოგის ფორმირებას, მიყავს ყველა მონაცემები კურსებზე, პროფესორებისა და სტუდენტების შესახებ.
- საანგარიშსწორებო სისტემა – ღებულობს მოცემული სისტემისაგან ინფორმაციას კურსების გადახდის შესახებ.
- კურსების კატალოგი – მონაცემთა ბაზა, რომელიც შეიცავს ინფორმაციას კურსების შესახებ.



ნახ.6.2.

აღწერების დამატება გამოყენებით შემთხვევებზე.

აღწერების დამატებისათვის:

1. გამოვყოთ ბრაუზერში გამოყენებითი შემთხვევა “კურსებზე დარეგისტრირება”.
2. დოკუმენტაციის ფანჯარაში შევიტანოთ შემდეგი აღწერა “ეს გამოყენებითი შემთხვევა აძლევს სტუდენტს შესაძლებლობას დარეგისტრირდეს კურსებზე მიმდინარე სემესტრში”.
3. შევქმნათ MS Word – ის მეშვეობით ტექსტური ფაილები ქვემოთ მოყვანილი გამოყენებითი შემთხვევების აღწერებისათვის.

გამოყენებითი შემთხვევების განსაზღვრისას უნდა დაინიშნოს თვითეულისათვის პრიორიტეტი, რომლითაც განისაზღვრება მისი შემდგომი რეალიზაციის თანმიმდევრობა.

იმისათვის, რომ გამოყენებით შემთხვევას დაუნიშნოთ პრიორიტეტი:

1. დავაწკაპუნოთ მარჯვენა ღილაკზე გამოყენებით შემთხვევაზე ბრაუზერში ან დიაგრამაზე.
2. გახსნილ მენიუში ავირჩიოთ პუნქტი Open Specification.
3. შევიტანოთ პრიორიტეტი ველში Rank ჩანართში General.

გამოყენებით შემთხვევთა ძირითადი და ალტერნატიული ნაკადების აღწერა რეგისტრაციის სისტემისათვის მოყვანილია დანართში 2.

3.საკონტროლო კითხვები

- რა დოკუმენტებით აისახება მოთხოვნები პროგრამული უზრუნველყოფისადმი;
- რა არის სისტემის ლექსიკონი;
- რას ნიშნავს დამატებითი სპეციფიკაციები;
- როგორ ხდება გამოყენებით შემთხვევათა საწყისი ვერსიის მოდელის შექმნა;
- როგორ ხდება მოქმედი პირების შექმნა Rose გარემოში;
- როგორ ხდება გამოყენებითი შემთხვევების შექმნა Rose გარემოში.

- როგორ ხდება გამოყენებით შემთხვევათა დიაგრამის საწყისი ვერსიის აგება;
- როგორ ხდება აღწერების დამატება გამოყენებით შემთხვევებზე;
- როგორ ხდება პრიორიტეტის დანიშვნა გამოყენებით შემთხვევაზე.

4.დავალება

- შევქმნათ გამოყენებით შემთხვევათა საწყისი ვერსიის მოდელი
- შევქმნათ მოქმედი პირები სისტემისათვის
- შევქმნათ გამოყენებითი შემთხვევები სისტემისათვის
- შევქმნათ გამოყენებით შემთხვევათა საწყისი დიაგრამა
- მოვახდინოთ აღწერების დამატება გამოყენებით შემთხვევებზე.

ლაბორატორიული სამუშაო №7

მოთხოვნების ანალიზი პროგრამული უზრუნველყოფისადმი. არქიტექტურული ანალიზი

1.სამუშაოს მიზანი

სამუშაოს მიზანია შევისწავლოთ:

- მოდელირებისა და დოკუმენტირების საერთო სტანდარტები;
- მოთხოვნათა რეალიზების მექანიზმების ფორმირება(კოოპერაციები);
- არქიტექტურული მექანიზმების (ანალიზის მექანიზმები) წინასწარი გამოვლენა (ძირითადი აბსტრაქციების იდენტიფიკაცია);
- პაკეტებისა და ტრასირების დიაგრამების შექმნა;
- ანალიზის კლასების შესაბამისი დიაგრამის შექმნა;

2.მეთოდური მითითებები ლაბორატორიული სამუშაოს ჩასატარებლად

ობირქტ-ორიენტირებული ანალიზის მიზანია პროგრამოლი უზრუნველყოფისადმი ფუნქციონალური მოთხოვნების ტრანსფორმაცია წინასწარ სისტემურ პროექტში და სისტემის არქიტექტურისათვის სტაბილური საფუძვლის შექმნა. პროექტირების პროცესში სისტემური პროექტი "იტვირთება" რეალიზაციის გარემოში ყველა არაფუნქციონალური მოთხოვნების გათვალისწინებით.

ობიექტ-ორიენტირებული ანალიზი მოიცავს ორი სახის მოღვაწეობას – არქიტექტურულ ანალიზს და გამოყენებითი შემთხვევების ანალიზს. არქიტექტურული ანალიზი სრულდება სისტემის არქიტექტორის მიერ და მოიცავს:

- მოდელირებისა და დოკუმენტირების საერთო სტანდარტების (შეთანხმებების) დამტკიცება;
- არქიტექტურული მექანიზმების (ანალიზის მექანიზმები) წინასწარი გამოვლენა;
- საპრობლემო სფეროს ძირითად აბსტრაქციათა ნაკრების (ანალიზის კლასები) ფორმირება;

შეთანხმებები მოდელირებისადმი განსაზღვრავენ:

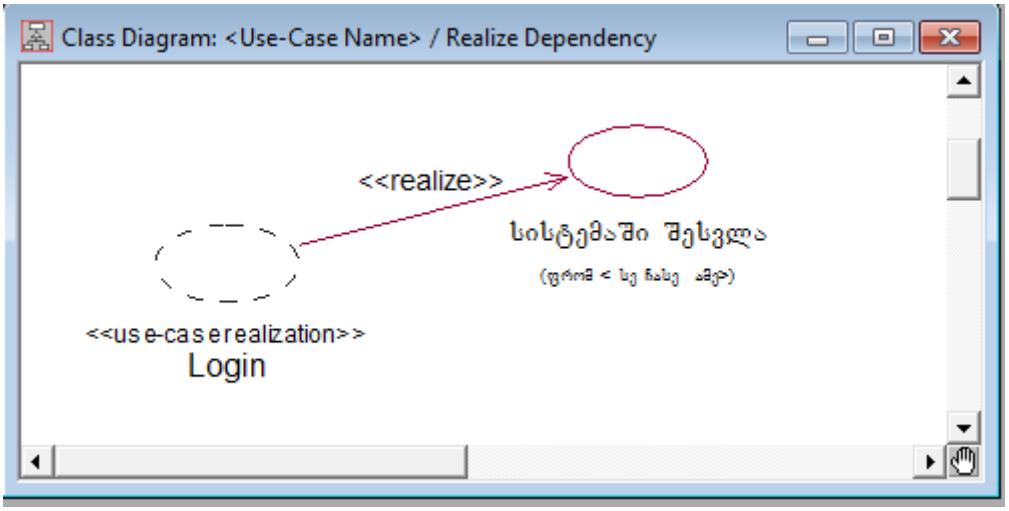
- გამოყენებულ დიაგრამებს და მოდელის ელემენტებს;
- მათი გამოყენების წესებს;
- შეთანხმებას მოდელის ელემენტების დასახელების შესახებ;
- მოდელის ორგანიზებას (პაკეტები).

მოდელირების შესახებ შეთანხმებების მაგალითი:

- გამოყენებით შემთხვევათა დასახელება უნდა იყოს მოკლე ზმნების ფრაზები;

- კლასების დასახელება უნდა იყოს არსებითი სახელი, საპრობლემო სფეროს შესაბამისი;
- კლასების დასახელება უნდა იწერებოდეს მთავრული ასოებით;
- ატრიბუტებისა და ოპერაციების დასახელება უნდა იწერებოდეს დაბალი ასოებით;
- ყველა კლასები და დიაგრამები, რომლებიც აღწერენ წინასწარ სისტემურ პროექტს, თავსდება პაკეტში დასახელებით Analysis Model; კლასების დიაგრამა, რომელიც ახდენს გამოყენებითი შემთხვევის რეალიზებას, და ურთიერთქმედების დიაგრამები, რომლებიც აღწერენ გამოყენებითი შემთხვევების სცენარების რეალიზების პროცესში ობიექტებს შორის ურთიერთქმედებას, თავსდებიან კოოპერაციებში მოცემული გამოყენებითი შემთხვევის სახელით და სტერეოტიპით “use case realization”. ყველა კოოპერაციები თავსდებიან პაკეტში Use Case Realizations. კავშირი გამოყენებით შემთხვევასა და მის რეალიზაციას შორის გამოისახება ტრასირების სპეციალურ დიაგრამაზე (ნახ.7.1.).

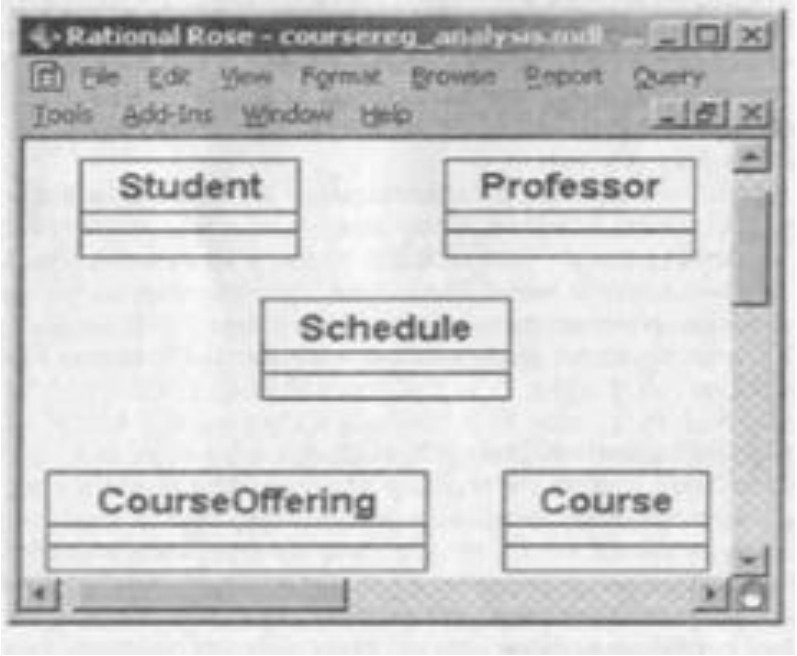
ძირითადი აბსტრაქციების იდენტიფიკაცია მდგომარეობს სისტემის კლასების ნაკრების (ანალიზის კლასები) წინასწარ განსაზღვრაში საპრობლემო სფეროს აღწერისა და სისტემისადმი მოთხოვნათა სპეციფიკაციის საფუძველზე. ძირითადი აბსტრაქციების იდენტიფიკაციის საშუალებები ანალოგიურია არსების იდენტიფიკაციის საშუალებებისა მოდელში “არსება – კავშირი”.



ნახ. 7.1. ტრასირების დიაგრამის ფრაგმენტი

ძირითადი (არაფორმალური) საშუალება არსების იდენტიფიკაციისა – ეს აბსტრაქციების პოვნაა, რომლებიც აღწერენ ფიზიკურ ან მატერიალურ ობიექტებს, პროცესებს და მოვლენებს, ადამიანთა როლებს, ორგანიზაციებს და სხვა მცნებებს. ერთადერთ ფორმალურ საშუალებათ არსთა იდენტიფიკაციისა არის საგნობრივი სფეროს ტექსტური აღწერილობების ანალიზი, აღწერიდან არსებითი სახელების გამოყოფა და მათი არჩევა როგორც "კანდიდატებისა" აბსტრაქციების როლზე. ყოველ არსს უნდა გააჩნდეს დასახელება, გამოსახული არსებითი სახელით მხოლობით რიცხვში. თუ მივყვებით ამ რეკომენდაციებს, განვსაზღვრავთ რეგისტრაციის სისტემისათვის ანალიზის ხუთ კლასს (ნახ.7.2.):

- სტუდენტი, პროფესორი, სასწავლო გრაფიკი, კურსი, შეთავაზებული კურსი.



ნახ.7.2. რეგისტრაციის სიტემის ანალიზის კლასები

მოდელის სტრუქტურის და ანალიზის კლასების შექმნა არქიტექტურული ანალიზის მოთხოვნების შესაბამისად.

იმისათვის, რომ შევქმნათ პაკეტები და ტრასირების დიაგრამები:

1. დავაწკაპუნოთ მარჯვენა ღილაკით პაკეტზე Use Case Realizations, რომელიც შედის პაკეტში Design Model ბრაუზერის ლოგიკურ წარმოდგენაში.
2. ავირჩიოთ პუნქტი New > Package გახსნილ მენიუში.
3. შევქმნათ პაკეტი დასახელებით Use Case Realization – Register for Courses, შემდეგ ასევე პაკეტები Use Case Realization – Close registration და Use Case Realization – Login.
4. შევქმნათ თითოეულ პაკეტში Use Case Realization შესაბამისი კოოპერაციები – Register for Courses, Close registration და Login (ყოველი კოოპერაცია იქმნება როგორც გამოყენებითი შემთხვევა სტერეოტიპით <<use case realization>>, რომელიც მიეთითება გამოყენებითი შემთხვევის სპეციფიკაციის

ფანჯარაში საჭირო სტერეოტიპის მითითებით სტერეოტიპების ცხრილიდან).

5. შექმნათ თითოეულ პაკეტში Use Case Realization გამოყენებით შემთხვევათა ახალი დიაგრამა დასახელებით Realize Depedency და ავაგოთ იგი ნახ. 4.1.-ის შესაბამისად, შეუსაბამოთ რა შესაბამისი კოოპერაციები და გამოყენებითი შემთხვევები.

ანალიზის კლასების და შესაბამისი დიაგრამის Key Abstractions შესაქმნელად:

- დავაწკაპუნოთ მარჯვენა ღილაკით პაკეტზე Analysis Model ბრაუზერის ლოგიკურ წარმოდგენაში.
- ავირჩიოთ პუნქტი New > Package გახსნილ მენიუში. ბრაუზერში გამოჩნდება ახალი კლასი დასახელებით NewClass.
- გამოვყოთ იგი შევიტანოთ მისი სახელი Student.
- ანალოგიურად შექმნათ კლასები Professor, Schedule, Course და CourseOffering.
- დავაწკაპუნოთ მარჯვენა ღილაკით პაკეტზე Analysis Model ბრაუზერის ლოგიკურ წარმოდგენაში.
- ავირჩიოთ პუნქტი New > Package გახსნილ მენიუში.
- ავირჩიოთ პუნქტი New > Class Diagram გახსნილ მენიუში.
- კლასების ახალ დიაგრამას დავარქვათ Key Abstraqtions.
- გავხსნათ კლასების დიაგრამა და გადავიტანოთ კლასები. კლასების დიაგრამა გამოიყურება ისე, როგორც ნახ.7.2.

3.საკონტროლო კითხვები

- რისთვის არის საჭირო შეთანხმებები მოდელირებისა და დოკუმენტირების საერთო სტანდარტებზე;
- როგორ ხდება ძირითადი აბსტრაქციების იდენტიფიკაცია;

- როგორ ხდება მოდელის სტრუქტურის და ანალიზის კლასების შექმნა არქიტექტურული ანალიზის მოთხოვნების შესაბამისად.
- როგორ ხდება პაკეტებისა და ტრასირების დიაგრამების შექმნა;
- როგორ ხდება ანალიზის კლასების შესაბამისი დიაგრამის შექმნა;

4.დავალება

- მოვახდინოთ ძირითადი აბსტრაქციების იდენტიფიკაცია;
- შევქმნათ მოდელის სტრუქტურის და ანალიზის კლასები არქიტექტურული ანალიზის მოთხოვნების შესაბამისად.
- შევქმნათ პაკეტებისა და ტრასირების დიაგრამები;
- შევქმნათ ანალიზის კლასები და შესაბამისი დიაგრამები

ლაბორატორიული სამუშაო №8

მოთხოვნების ანალიზი პროგრამული უზრუნველყოფისადმი. გამოყენებითი შემთხვევების ანალიზი

1.სამუშაოს მიზანი

სამუშაოს მიზანია შევისწავლოთ:

- კლასების იდენტიფიკაცია(მოსაზღვრე, არსი და მმართველი კლასები);
- კლასების შექმნა;
- კლასების მოვალეობების განსაზღვრა;
- ურთიერთქმედების(მიმდევრობის) დიაგრამების შექმნა;
- მოქმედი პირების, ობიექტებისა და შეტყობინებების დამატება;
- შეტყობინებების შეთავსება ოპერაციებზე.

2.მეთოდური მითითებები ლაბორატორიული სამუშაოს ჩასატარებლად

გამოყენებითი შემთხვევების ანალიზი მოიცავს:

- კლასების იდენტიფიკაციას, რომლებიც მონაწილეობენ გამოყენებითი შემთხვევების მოვლენათა ნაკადების რეალიზაციაში;
- ქცევის განაწილება, რომელიც რეალიზდება გამოყენებითი შემთხვევის მიერ, კლასებს შორის (კლასების მოვალეობების განსაზღვრა);
- კლასების ატრიბუტებისა და ასსოციაციის განსაზღვრა;
- ანალიზის კლასების უნიფიცირება.

კლასების იდენტიფიკაცია.

გამოყენებითი შემთხვევის მოვლენათა ნაკადში სამი ტიპის კლასებს გამოავლენენ:

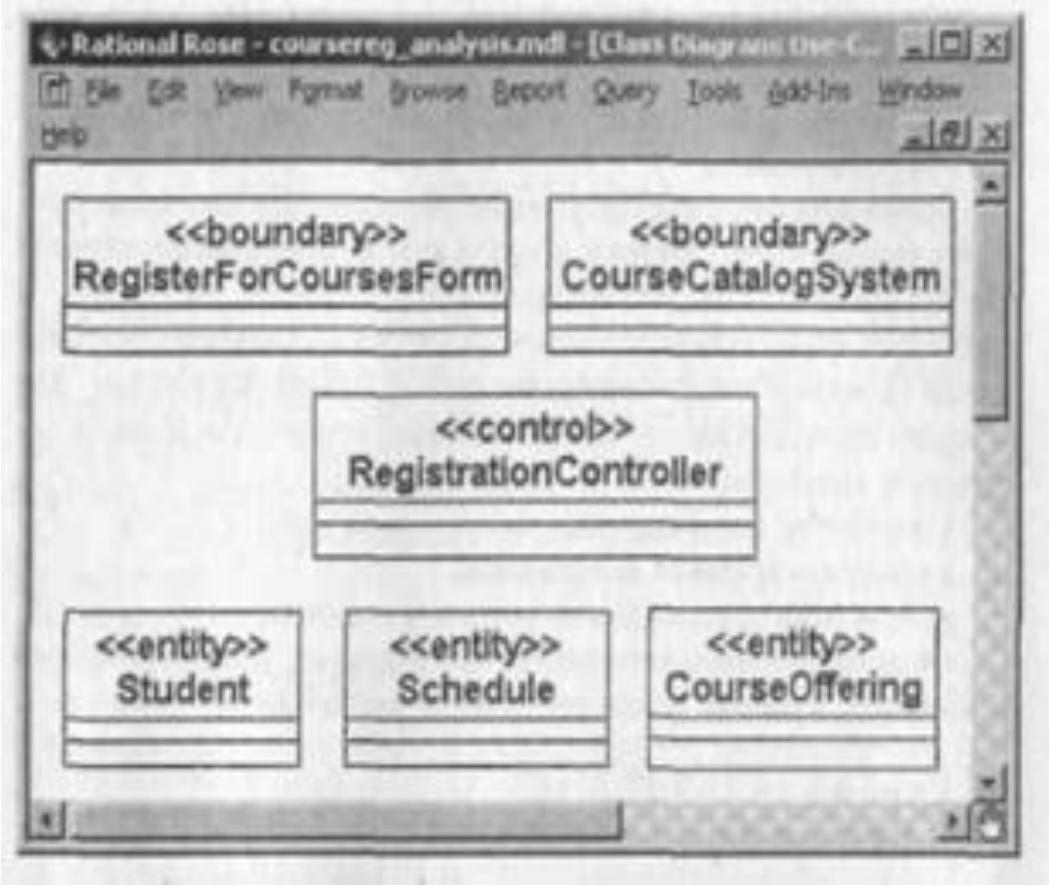
1. **მოსაზღვრე კლასები** (Boundary) - შუამავლები გარე ობიექტებსა და სისტემას შორის ურთიერთქმედებისას. როგორც წესი, ყოველი წყვილისათვის "მოქმედი პირი – გამოყენებითი შემთხვევა" განისაზღვრება ერთი მოსაზღვრე კლასი. მოსაზღვრე კლასების ტიპები: მომხმარებლის ინტერფეისი (ინფორმაციის გაცვლა მომხმარებელთან ინტერფეისის დეტალების გარეშე – ღილაკები, ცხრილები, ფანჯრები), სისტემური ინტერფეისი და აპარატული ინტერფეისი (გამოყენებული პროტოკოლები მათი რეალიზაციის გარეშე).
2. **კლასები – არსება** (Entity) – დასამუშავებელი სისტემის ძირითადი აბსტრაქციებია (მცნებები). კლასები – არსის გამოვლენის წყაროებია – აბსტრაქციები, რომლებიც იქმნება არქიტექტურული ანალიზისას, გამოყენებითი შემთხვევის მოვლენათა ნაკადის აღწერა.
3. **მმართველი კლასები** (Control) – უზრუნველყოფენ ობიექტების ქცევის კოორდინაციას სისტემაში. შეიძლება არ იყოს ზოგიერთ გამოყენებით

შემთხვევაში, შეზღუდულია მარტივი მანიპულაციებით შენახულ მონაცემებთან. როგორც წესი, ყოველი გამოყენებითი შემთხვევისათვის განისაზღვრება ერთი მმართველი კლასი. მმართველი კლასების მაგალითია: ტრანზაქციების მენეჯერი, რესურსების კოორდინატორი, შეცდომების დამმუშავებელი.

ანალიზის კლასები გამოხატავენ სისტემისადმი ფუნქციონალურ მოთხოვნებს და ახდენენ საგნობრივი სფეროს მოდელირებას. ანალიზის კლასების ერთობლიობა წარმოადგენს სისტემის საწყის კონცეპტუალურ მოდელს. კლასების ნაკრების მაგალითი, რომლებიც მონაწილეობენ გამოყენებითი შემთხვევის “კურსზე დარეგისტრირება” რეალიზებაში, მოყვანილია ნახ.4.3.-ზე.

კლასების შექმნა:

1. დავაწკაპუნოთ მარჯვენა ღილაკით პაკეტზე Analyzis Model.
2. ავირჩიოთ პუნქტი New>Clas გახსნილ მენიუში. ახალი კლასი დასახელებით NewClass გამოჩნდება ბრაუზერში
3. გამოვყოთ და შევიტანოთ დასახელება RegisterForCoursesForm.
4. დავაწკაპუნოთ მარჯვენა ღილაკით კლასზე RegisterForCoursesForm.
5. ავირჩიოთ პუნქტი Open Specification გახსნილ მენიუში.
6. ავირჩიოთ სტერეოტიპი <<Boundary>> სტერეოტიპის ველში და დავაწკაპუნოთ OK.
7. ანალოგიურად შევქმნათ კლასები Course Catalog System სტერეოტიპით <<Boundary>> და RegistrationController სტერეოტიპით Control.



ნახ.8.1.

8. დაუნიშნოთ კლასებს Schedule, CourseOffering და Student სტერეოტიპი Entity.
9. დავაწკაპუნოთ მარჯვენა ღილაკით კოოპერაციაზე Register for Courses პაკეტში Use Case Realization – Register for Courses.
10. ავირჩიოთ პუნქტი New>Class Diagram გახსნილ მენიუში.
11. დავარქვათ კლასების ახალ დიაგრამას Register for Courses – Participating Classes.
12. გავხსნათ იგი და გადავიტანოთ კლასები ნახ.4.3.-ის შესაბამისად.

კლასებს შორის მოვალეობების განაწილება.

გამოყოფილი სამი ტიპის კლასების დანიშნულებიდან გამომდინარე, შესაძლებელია დავახასიათოთ მოვალეობები მათ შორის:

- მოსაზღვრე კლასები პასუხს აგებენ ურთიერთქმედებაზე სისტემის გარე სამყაროსთან (მოქმედი პირები);

- კლასი არსებები პასუხს აგებენ მონაცემთა შენახვასა და მანიპულირებაზე;
- მმართველი კლასები კოორდინაციას უწევენ გამოყენებითი შემთხვევების მოვლენათა ნაკადებს.

მოვალეობების უფრო დეტალური განაწილება (კლასების ოპერაციების სახით) სრულდება ურთიერთქმედების დიაგრამებით. პირველ რიგში იგება დიაგრამა, რომელიც აღწერს მონაცემთა ძირითად ნაკადს (ერთი ან რამოდენიმე) და მის დაქვემდებარებულ ნაკადებს. ყოველი ალტერნატიული ნაკადისათვის იგება ცალკე დიაგრამა. მაგალითად:

- შეცდომების დამუშავება;
- შესრულების დროის კონტროლი;
- არასწორად შეტანილი მონაცემების დამუშავება.

ურთიერთქმედების დიაგრამების შექმნა.

შევქმნათ მიმდევრობის და კოოპერაციის დიაგრამები გამოყენებითი შემთხვევისათვის “კურსებზე დარეგისტრირება” ნახ.8.2.

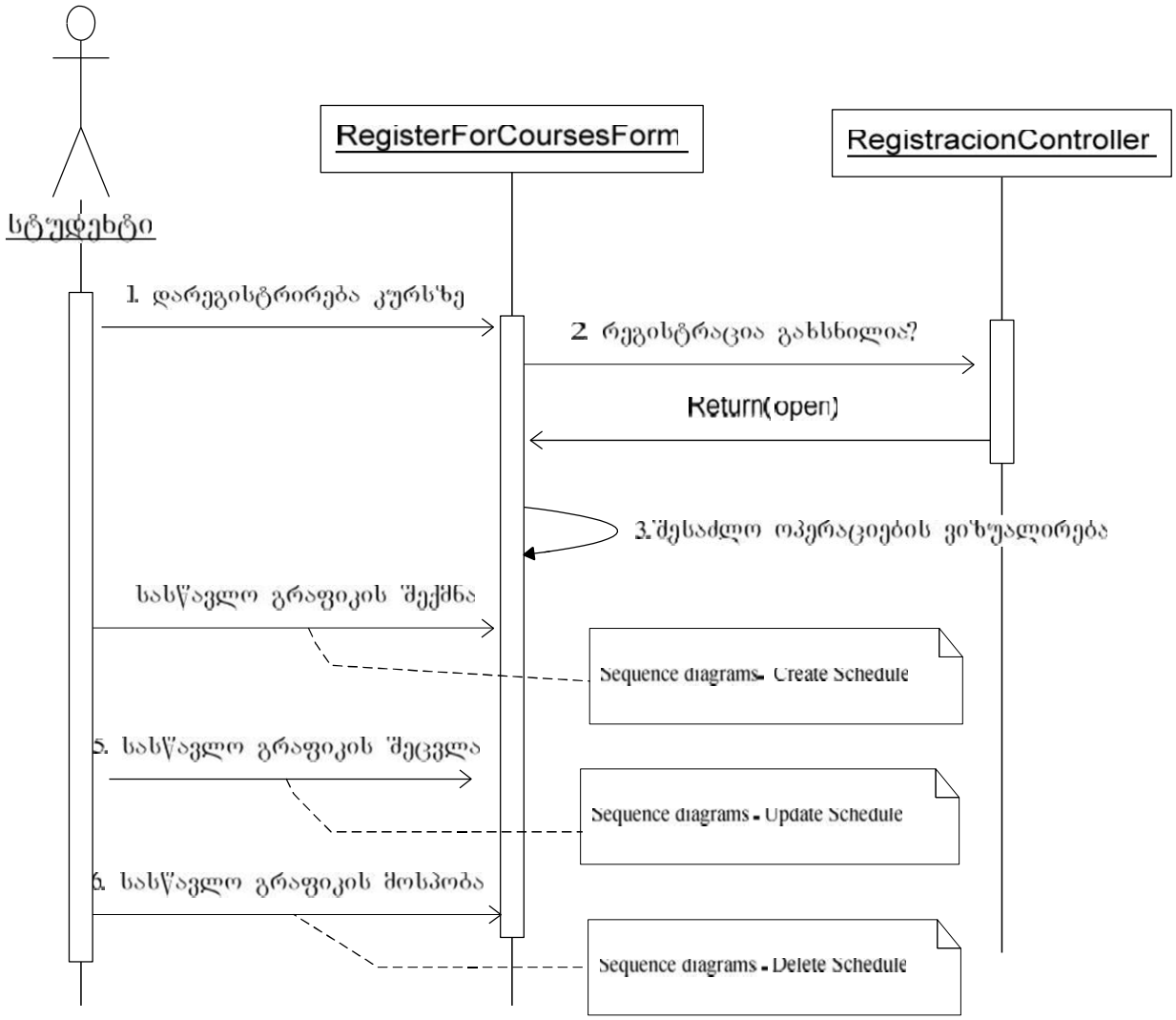
დაყენება:

1. ავირჩიოთ პუნქტი Tools>Options მოდელის მენიუში.
2. გადადით დიაგრამის ჩანართზე.
3. საკონტროლო გადამრთველები Sequence Numbering, Collaboration Numbering უნდა მონიშნულნი იყვნენ, ხოლო Focus of Control – არა.
4. დავაწკაპუნოთ ღილაკზე OK, რათა გამოვიდეთ პარამეტრების ფანჯრიდან.

მიმდევრობის დიაგრამის შექმნა:

1. დავაწკაპუნოთ მარჯვენა ღილაკით კოოპერაციაზე Register for Courses პაკეტში Use Case Realization – Register for Courses.
2. ავირჩიოთ პუნქტი New > Seguence Diagram გახსნილ მენიუში.
3. დავარქვათ ახალ დიაგრამას Register for Courses – Basic Flow.

4. დავაწკაპუნოთ მასზე ორჯერ, რათა იგი გაიხსნას.



ნახ.8.2. მიმდევრობის დიაგრამა Register for Courses – Basic Flow ძირითადი ნაკადი შეცდომების დამუშავება.

მოქმედი პირების, ობიექტებისა და შეტყობინებების დამატება:

1. გადმოვიტანოთ მოქმედი პირი “სტუდენტი” ბრაუზერიდან დიაგრამაზე.
2. გადმოვიტანოთ კლასები RegisterForCoursesForm და RegistrationController ბრაუზერიდან დიაგრამაზე. იმისათვის, რომ განვალაგოთ ობიექტები ორ არსებულ ობიექტებს შორის, საკმარისია დავაწკაპუნოთ მათ შორის.
3. დავაწკაპუნოთ ღილაკზე Objekt Message ინსტრუმენტების პანელიზე.

4. გავატაროთ თაგვი მოქმედი პირის “სტუდენტ” სიცოცხლის ხაზიდან სიცოცხლის ხაზამდე ობიექტ RegisterForCoursesForm.
5. გამოვყოთ შეტყობინება და შევიტანოთ მისი სახელი://register for courses.
6. გავიმეოროთ მოქმედებები პ.3-5 იმისათვის, რომ მოვათავსოთ დიაგრამაზე დანარჩენი შეტყობინებები, როგორც ეს ნაჩვენებია ნახ.8.3.-ზე (რეფლექსური შეტყობინებისათვის 3 გამოიყენება ღილაკი Message to self).

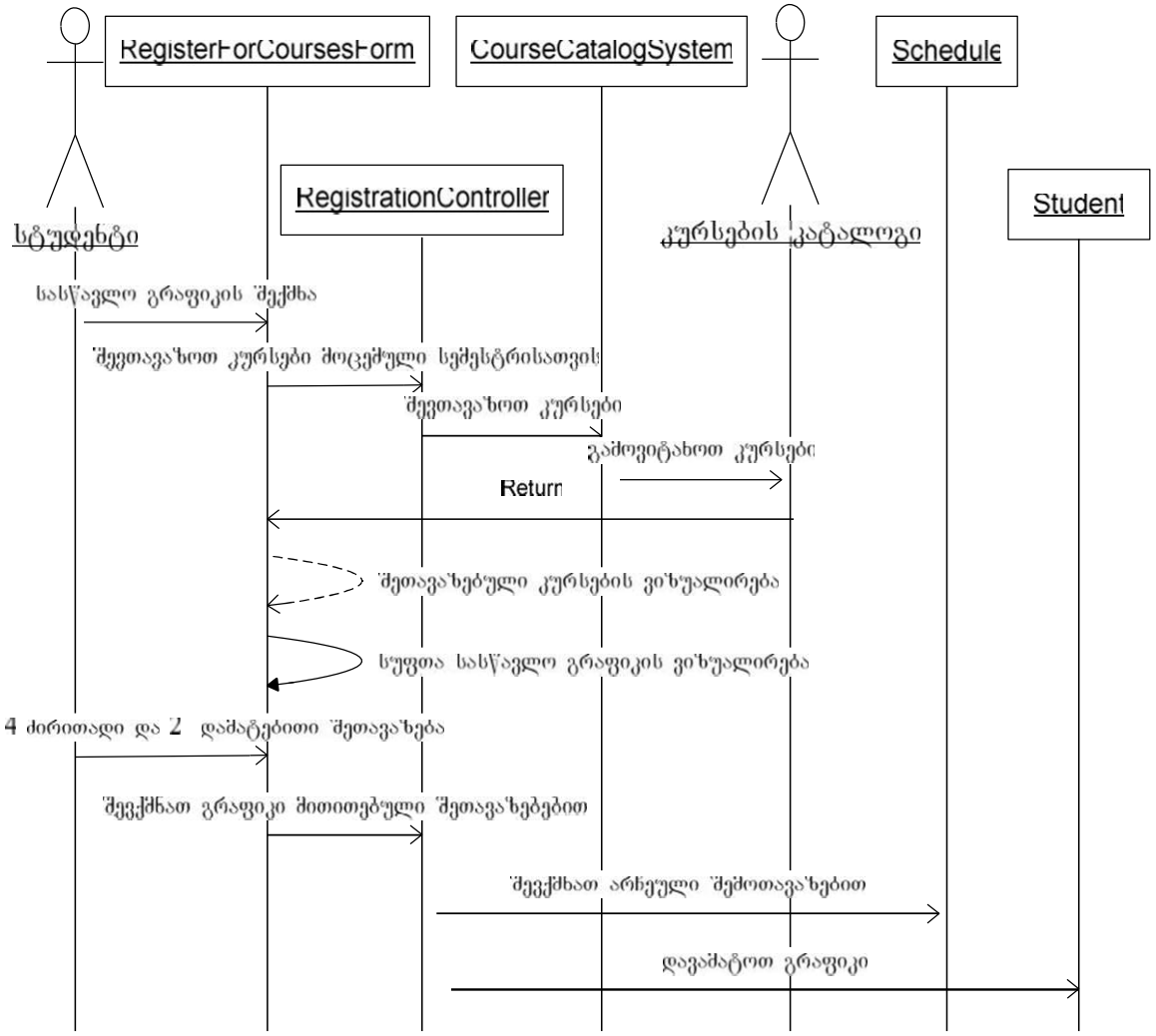
შეტყობინებების შეთავსება ოპერაციებზე:

1. დავაწკაპუნოთ მარჯვენა ღილაკით შეტყობინების ტექსტზე 1.// register for courses.
2. გახსნილ მენიუში ავირჩიოთ პუნქტი new operation. გამოჩნდება ოპერაციების სპეციფიკაციის ფანჯარა.
3. დავტოვოთ შეტყობინების სახელი - // register for courses დასახელების ველში.
4. დავაწკაპუნოთ ღილაკზე OK, რათა დავხუროთ ოპერაციების სპეციფიკაციის ფანჯარა და დავბრუნდეთ დიაგრამაზე.
5. გავიმეოროთ მოქმედებები პ. 1-4, სანამ არ შეუსაბამებთ ყველა დანარჩენ შეტყობინებებს ოპერაციებს.

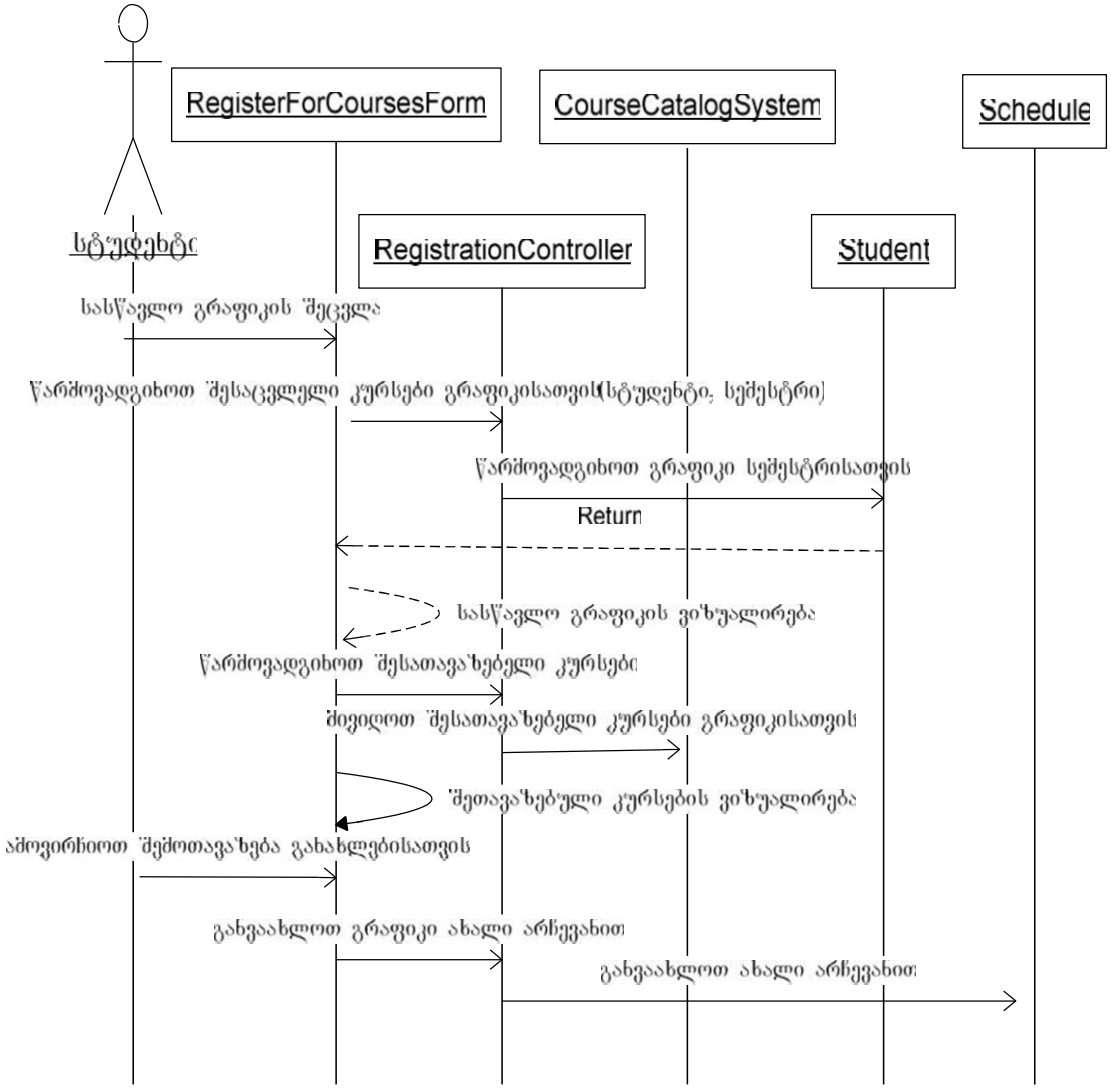
შევასრულოთ ანალოგიური მოქმედებები მიმდევრობის დიაგრამების შესაქმნელად ნახ.8.3.-8.5.

იმისათვის, რომ დავამატოთ შენიშვნები დიაგრამაზე:

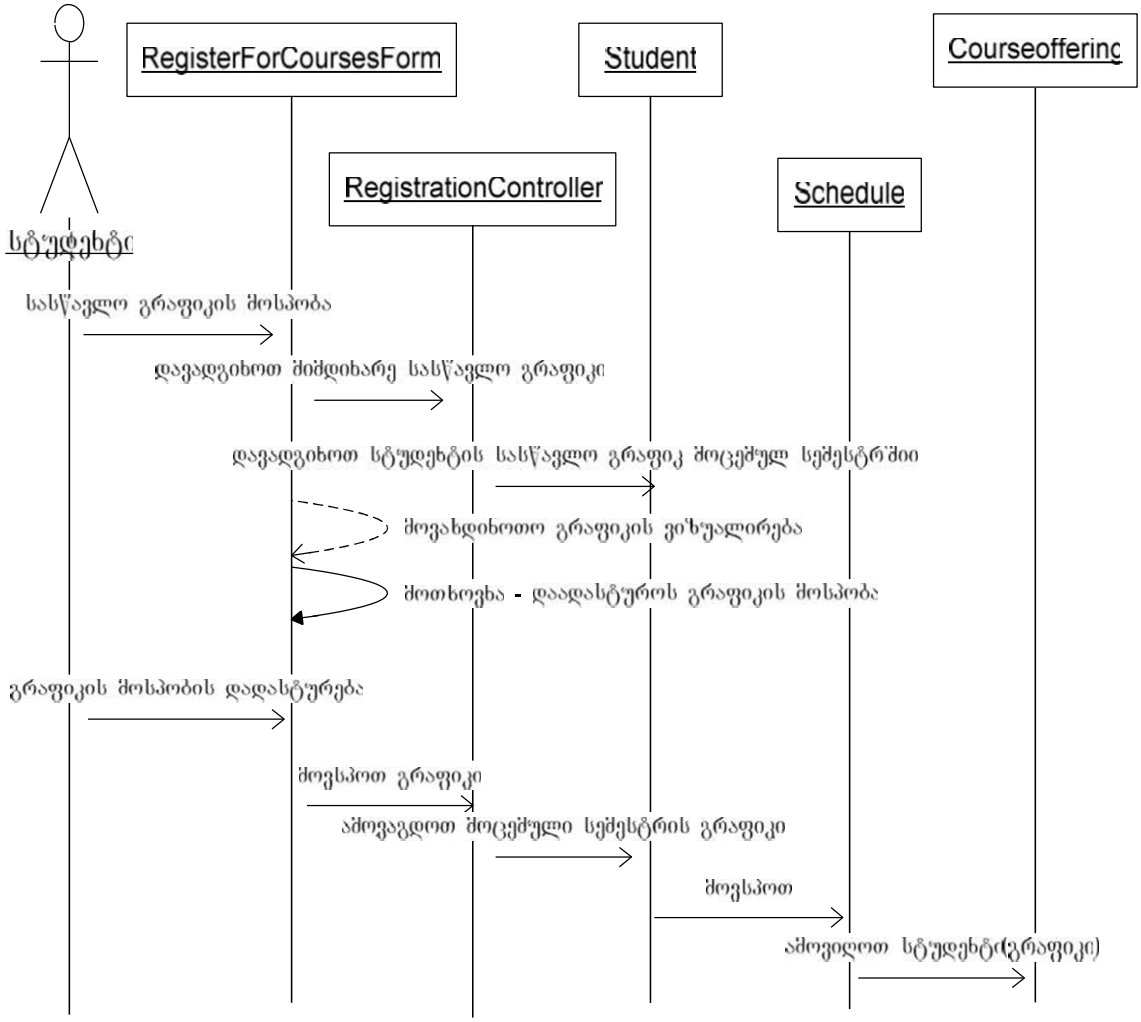
1. დავაწკაპუნოთ ინსტრუმენტების პანელის ღილაკზე Note.
2. დავაწკაპუნოთ დიაგრამის იმ ნაწილზე, სადაც ვაპირებთ შენიშვნის მოთავსებას.
3. გამოვყოთ ახალი შენიშვნა და შევიტანოთ მასში ტექსტი.
4. დავაწკაპუნოთ ღილაკზე Anchor Notes To Item ინსტრუმენტების პანელზე, რომ მივამაგროთ შენიშვნა დიაგრამის ელემენტს.



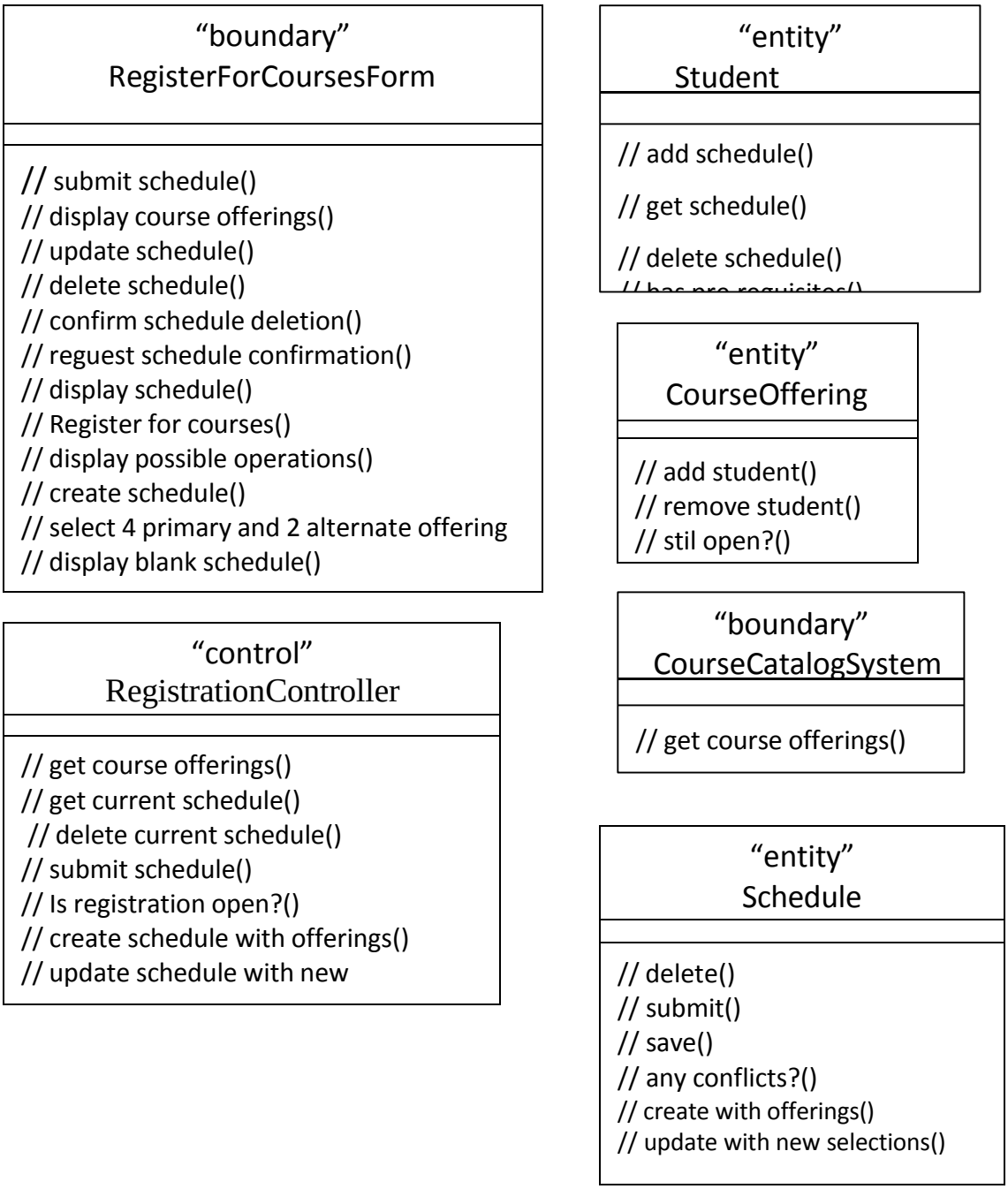
8.3. მიმდევრობის დიაგრამა Register for Courses – ძირითადი ნაკადი(გრაფიკის შექმნა), Basic Flow(Create Schedule)



8.4. მიმდევრობის დიაგრამა Register for Courses – ძირითადი ნაკადი(გრაფიკის შეცვლა) , Basic Flow(Update Schedule)



8.5. მიმდევრობის დიაგრამა Register for Courses – ძირითადი ნაკადი(გრაფიკის ამოგდება) , Basic Flow(Delete Schedule)



8.6, კლასების დიაგრამა “კურსებზე დარეგისტრირება”-ში მონაწილე “ანალიზის” ოპერაციებით

9. დავაწკაპუნოთ მარცხენა ღილაკზე და გავავლოთ მაჩვენებელი შენიშვნიდან ელემენტის დიაგრამამდე, რომელთანაც იქნება იგი დაკავშირებული.
10. შევქმნათ ცარიელი შენიშვნა (ტექსტის გარეშე) და გადავიტანოთ მასზე ბრაუზერიდან საჭირო დიაგრამა, იმისათვის რომ შევქმნათ შენიშვნა-გადასვლა სხვა დიაგრამაზე.

შენიშვნის გარდა დიაგრამაზე შესაძლებელია მოვათავსოთ ტექსტური ველი და მისი საშუალებით, მაგალითად, დაუმატოთ შეტყობინებას პირობა.

იმისათვის, რომ მოვათავსოთ დიაგრამაზე ტექსტური ველი:

1. დავაწკაპუნოთ ღილაკზე Text Box მართვის პანელიზე.
2. დავაწკაპუნოთ დიაგრამის შიგნით, რათა მოვათავსოთ მასში ტექსტური ველი.
3. გამოვყოთ ეს ველი და შევიყვანოთ მასში ტექსტი.

კოოპერაციის დიაგრამის შექმნისათვის საკმარისია გავხსნათ მიმდევრობის დიაგრამა და დავაჭიროთ F5 ღილაკს.

ურთიერთქმედების დიაგრამის აგების პროცესში (შეტყობინებების შესაბამისობა ოპერაციებთან) კლასებში ავტომატურად ჩნდება “ანალიზის” ოპერაციები. მაშასადამე, კლასების დიაგრამამ ნახ.8.1. ურთიერთქმედების დიაგრამის აგების შემდეგ, უნდა მიიღოს სახე ნახ.8.6.

3.საკონტროლო კითხვები

- რას ნიშნავს მოსაზღვრე, არსი და მმართველი კლასები;
- როგორ ხდება კლასების შექმნა;
- როგორ ხდება კლასების მოვალეობების განსაზღვრა;
- როგორ ხდება ურთიერთქმედების დიაგრამების შექმნა;
- როგორ ხდება მიმდევრობის დიაგრამის შექმნა;

- როგორ ხდება მოქმედი პირების, ობიექტებისა და შეტყობინებების დამატება;
- როგორ ხდება შეტყობინებების შეთავსება ოპერაციებზე.

4.დავალება

- შევქმნათ მოსაზღვრე, არსი და მმართველი კლასები
- განვსაზღვროთ კლასების მოვალეობები-ავაგოთ ურთიერთქმედების(მიმდევრობის) დიაგრამები
- მოვახდინოთ მოქმედი პირების, ობიექტებისა და შეტყობინებების დამატება;
- მოვახდინოთ შეტყობინებების შეთავსება ოპერაციებზე.

ლაბორატორიული სამუშაო №9
კლასების ატრიბუტებისა და ასოციაციის განსაზღვრა

1.სამუშაოს მიზანი
სამუშაოს მიზანია შევისწავლოთ:

- ატრიბუტების დამატება კლასებზე;
- კლასებს შორის კავშირების(ასოციაციის) განსაზღვრა;
- კავშირების დამატება;
- ასოციაციის შექმნა კლასების დიაგრამაზე;
- განზოგადების შექმნა კლასების დიაგრამაზე.

2.მეთოდური მითითებები ლაბორატორიული სამუშაოს შესასრულებლად

ანალიზის კლასების ატრიბუტები განისაზღვრებიან საგნობრივი სფეროს შესახებ ცოდნისა და სისტემისადმი მოთხოვნების საფუძველზე.

ატრიბუტების დამატება კლასებზე.

დაყენება:

1. ავირჩიოთ პუნქტი Tools > Options მოდელის მენიუში.
2. გადავიდეთ ჩანართზე “Diagram”.
3. დარწმუნდით, რომ გადამრთველი Show All Atributes მონიშნულია.
4. დარწმუნდით, რომ გადამრთველი Suppress Atributes და Suppress Operations არ არის მონიშნული.

ატრიბუტების დამატება:

1. დავაწკაპუნოთ მარჯვენა ღილაკით კლასზე Student.
2. ავირჩიოთ პუნქტი New Attribute გახსნილ მენიუში.
3. შევიტანოთ ახალი ატრიბუტი Address.
4. დავაჭიროთ ღილაკს Enter.
5. გავიმეოროთ მოქმედებები პ.1-4, დავამატოთ ატრიბუტები name და studentID.
6. დავამატოთ ატრიბუტები კლასებს CourseOffering და Shedule ნახ.9.1.

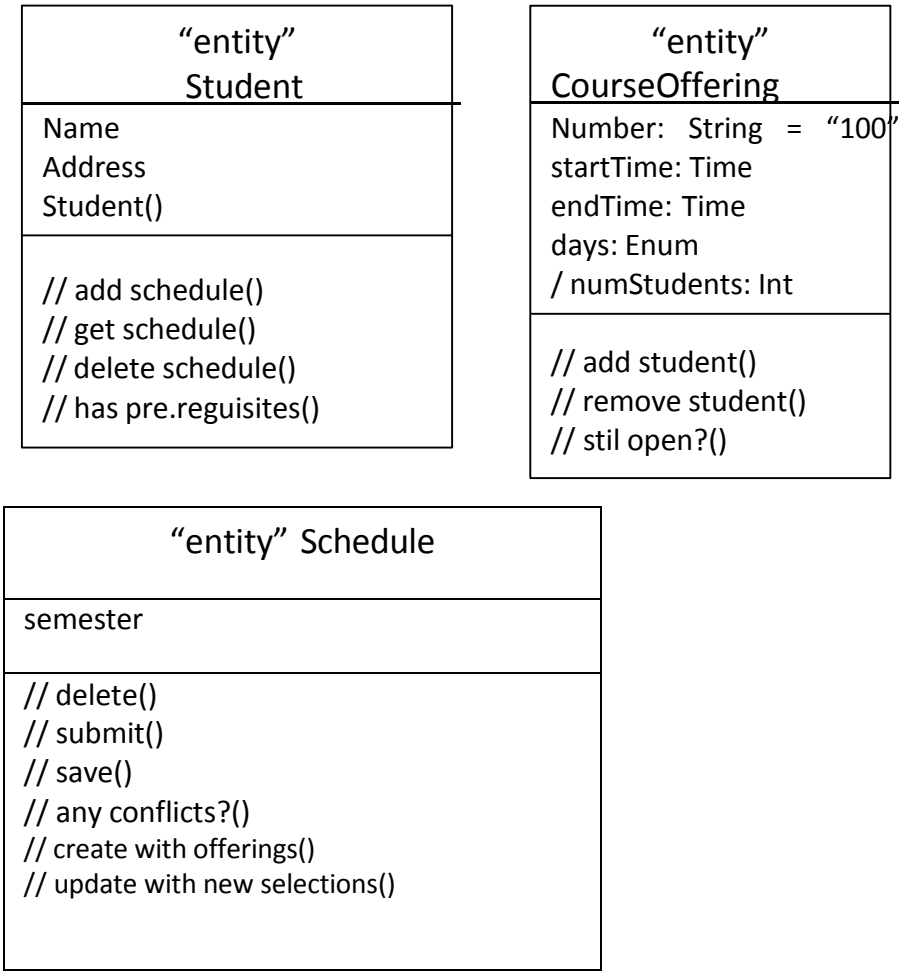
კლასებს (ასოციაციის) შორის კავშირები განისაზღვრება ორ ეტაპად:

ეტაპი 1. კავშირების საწყისი ნაკრები დგინდება კოოპერაციის დიაგრამების ანალიზის საფუძველზე. თუ ორი ობიექტი ურთიერთქმედებენ (ცვლიან შეტყობინებებს), მათ შორის კოოპერაციის დიაგრამაზე უნდა არსებობდეს კავშირი (ურთიერთქმედების გზა), რომელიც გარდაიქმნება ორმიმართულებიან ასოციაციში შესაბამის კლასებს შორის. თუ შეტყობინებები რომელიმე ობიექტებს შორის

გადაიცემიან მხოლოდ ერთი მიმართულებით, მაშინ შესაბამისი ასოციაციისათვის ინიშნება მიმართულების ნავიგაცია.

ეტაპი 2. ანალიზდება და ზუსტდება ასოციაციები კლას-არსებებს შორის.

დგინდება ასოციაციის სიმძლავრე, შეიძლება გამოყენებულ იქნას მრავლობითი ასოციაციები, აგრეგაციები, განზოგადება და ასოციაცია კლასები.



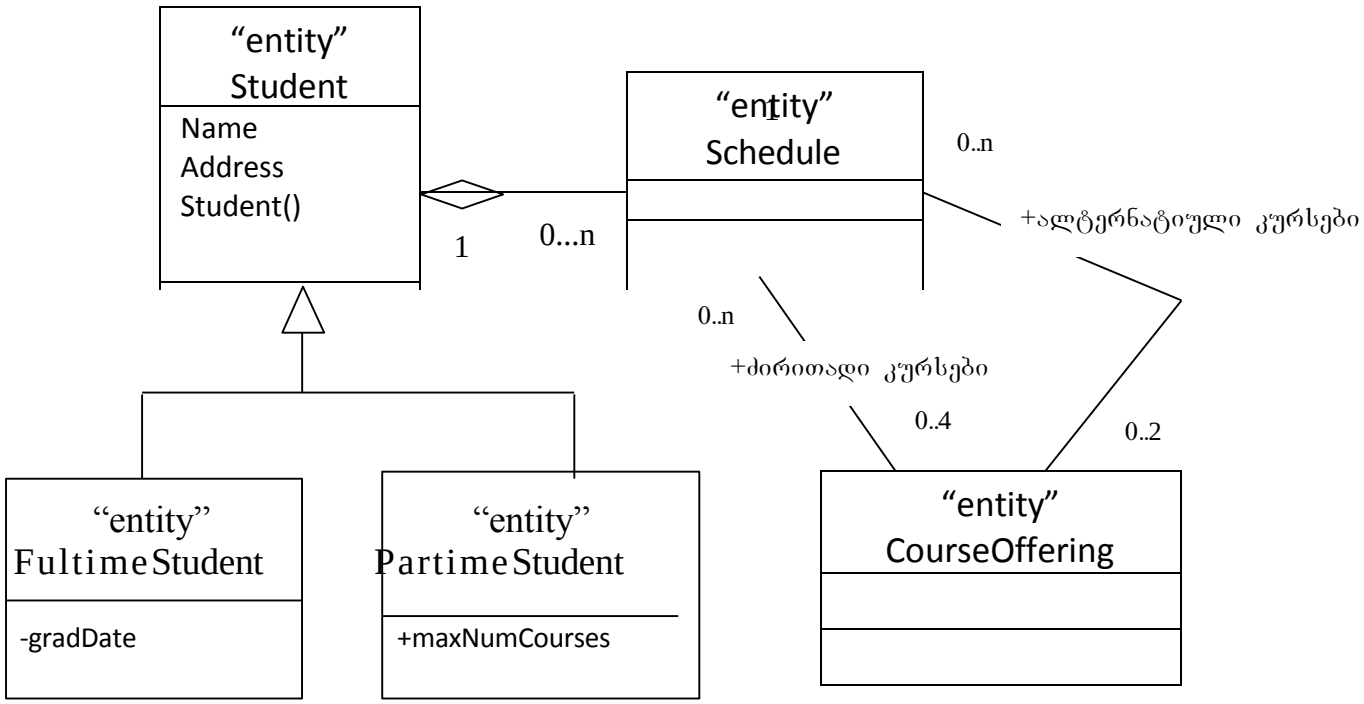
ნახ.9.1. "ანალიზის" კლასები ოპერაციებით და ატრიბუტებით

კავშირების დამატება.

დავამატოთ კავშირები კლასებს, რომლებიც მონაწილეობენ გამოყენებით შემთხვევაში Register for Courses. კლასებს შორის კავშირების გამოსახვისათვის

ავაგოთ სამი ახალი კლასების დიაგრამა კოოპერაციაში Register for Courses პაკეტში Use Case Realization - Register for Courses (ნახ.9.2 -9.3).

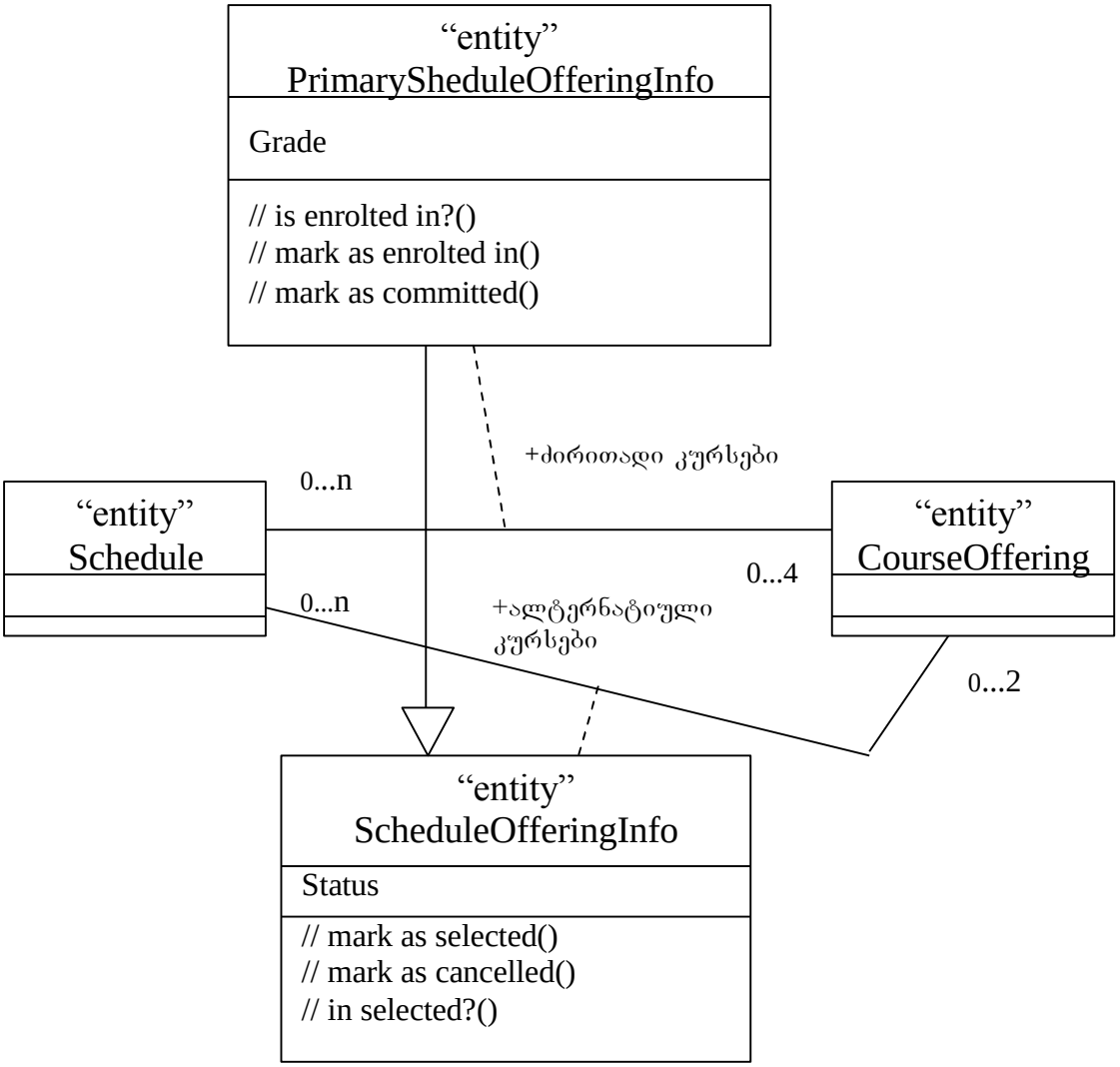
ნახ. 9.2.-ზე ნაჩვენებია მხოლოდ კლასი-არსები. აგრეგაცია კლასებს შორის Student და Schedule გამოსახავს იმ ფაქტს, რომ ყოველი გრაფიკი წარმოადგენს კონკრეტული სტუდენტის საკუთრებას, ეკუთვნის მხოლოდ მას. იგულისხმება, რომ სისტემაში შეინახება არა მარტო მიმდინარე სემესტრის გრაფიკი, არამედ სტუდენტის ყველა გრაფიკები სხვადასხვა სემესტრის. კლასებს შორი Schedule და CourseOffering შემოტანილია ორი ასოციაცია, რამდენადაც კონკრეტული კურსი შესაძლებელია შედიოდეს სტუდენტის გრაფიკში როგორც ძირითადი (არა უმეტეს ოთხი კურსისა) და როგორც ალტერნატიული (არა უმეტეს ორი კურსისა). კლასს Student ემატება ორი ახალი ქვეკლასი – FultimeStudent (დღის განყოფილების სტუდენტი) და PartimeStudent (საღამოს განყოფილების სტუდენტი).



ნახ. 9.2. კლასი – არსი დიაგრამა

ნახ.9.3.-ზე მოყვანილია ასსოციაცია – კლასები, რომლებიც წარმოადგენენ კავშირების თვისებებს კლასებს შორის Schedule და CourseOffering. ასოციაცია,

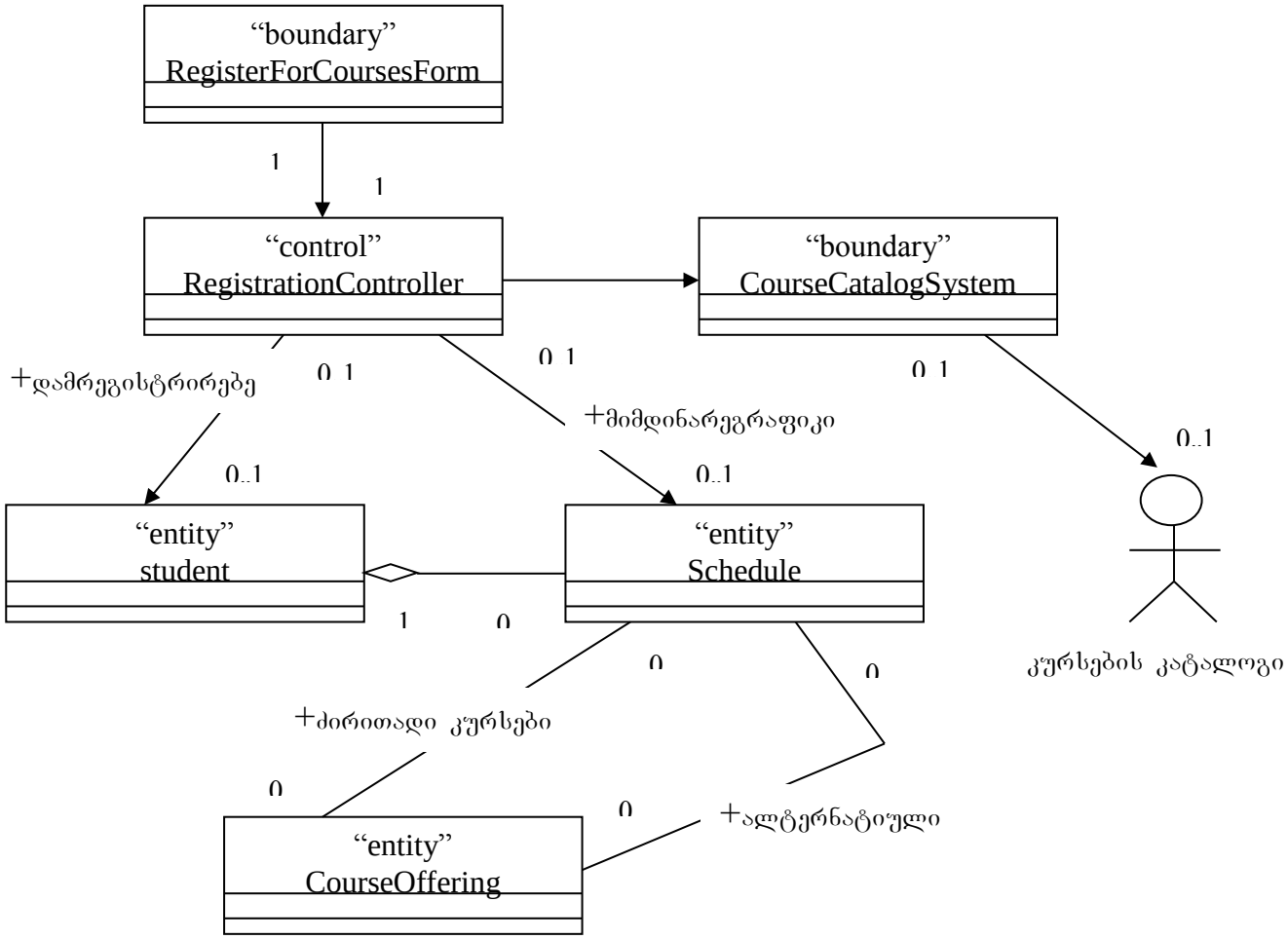
რომელიც აკავშირებს გრაფიკს და ალტერნატიულ კურსს, აქვს მხოლოდ ერთი ატრიბუტი – კურსის სტატუსი კონკრეტულ გრაფიკში (status), რომელსაც შეუძლია მიიღოს მნიშვნელობები “ჩართულია გრაფიკში”, “გადაიდო”, “შეტანილია კურსის ცხრილში” და “დაფიქსირებულია გრაფიკში”. თუ კურსი რეგისტრაციის დახურვის პროცესში გადადის ალტერნატიულიდან ძირითადში, მაშინ შესაბამის ასოციაციას ემატება ატრიბუტი “შეფასება” (grade). მაშასადამე, ასოციაცია – კლასი PrimaryScheduleOfferingInfo (ატრიბუტები და ოპერაციები, რომლებიც შედიან ამ კლასში, ეკუთვნიან როგორც ძირითად, ასევე ალტერნატიულ კურსებს) და ემატება თავისი საკუთარი (შეფასება და საბოლაო ჩართვა კურსისა გრაფიკში შესაძლებელია ადგილი ქონდეს მხოლოდ ძირითადი კურსებისათვის), რაც ნაჩვენებია განზოგადების კავშირის საშუალებით.



ნახ.9.3. დიაგრამა CourseOfferingInfo (ასოციაცია – კლასების მაგალითები)

ნახ.9.4.-ზე ნაჩვენებია კლასების გაფართოებული დიაგრამა გამოყენებითი შემთხვევისათვის “კურსებზე დარეგისტრირება” (ატრიბუტებისა და ოპერაციების გარეშე). ასოციაციები მოსაზღვრე და მმართველ კლასებს შორის, ასევე მმართველ კლასებსა და კლას-არსებს შორის შემოტანილია კოოპერაციის დიაგრამების ანალიზის საფუძველზე და მდგრადი სტრუქტურული (სემანტიკური) კავშირები არსებს შორის გამოხატავენ კავშირებს, დინამიურად აღიძვრებიან შესაბამის ობიექტებს შორის მართვის ნაკადში (დანართის მუშაობის პროცესში). რამდენადაც ასოციაციისათვის ეს

არ არის დამახასიათებელი, მომავალში (პროექტირების პროცესში) ისინი შესაძლებელია გარდაიქმნენ დამოკიდებულებაში.



ნახ.9.5. კლასების მთლიანი დიაგრამა “კურსებზე დარეგისტრირება” - მონაწილე კლასები(ატრიბუტების გარეშე)

ასოციაციებს ქმნიან უშუალოდ კლასების დიაგრამაზე. კლასების დიაგრამის ინსტრუმენტების პანელი შეიცავს ღილაკებს როგორც ერთ ისე ორმიმართულებიანი ასსოციაციების შექმნისათვის.

იმისათვის, რომ შეიქმნას ასოციაცია კლასების დიაგრამაზე:

1. დავაწკაპუნოთ ღილაკზე Association ინსტრუმენტების პანელზე:
2. გაატარეთ ასოციაციის ხაზი ერთი კლასიდან მეორეზე.

იმისათვის, რომ მიუთითოთ ასოციაციაზე ნავიგაცია:

1. დავაწკაპუნოთ ღილაკზე Association დიგრამის ინსტრუმენტების პანელზე.
2. ავირჩიოთ პუნქტი Navigable გახსნილ მენიუში.

იმისათვის, რომ მიუთითოდ რეფლექსური ასოციაცია:

1. დავაწკაპუნოთ ღილაკზე Association დიგრამის ინსტრუმენტების პანელზე.
2. გავატაროთ ასოციაციის ხაზი კლასიდან რომელიმე ადგილამდე კლასის გარეთ.
3. გაუშვათ ღილაკი.
4. გავატაროთ ასოციაციის ხაზი უკან კლასისკენ.

იმისათვის, რომ შევქმნათ აგრეგაცია:

1. დავაწკაპუნოთ ღილაკზე Aggregation ინსტრუმენტების პანელზე.
2. გავატაროთ აგრეგაციის ხაზი კლასი-ნაწილიდან მთელამდე.

იმისათვის, რომ მოვათავსოთ კლასების დიაგრამაზე რეფლექსური აგრეგაცია:

1. დავაწკაპუნოთ ღილაკზე Aggregation დიგრამის ინსტრუმენტების პანელზე.
2. გავატაროთ აგრეგაციის ხაზი კლასიდან რომელიმე ადგილამდე კლასის გარეთ.
3. გაუშვათ ღილაკი.
4. გავატაროთ აგრეგაციის ხაზი უკან კლასისკენ.

განზოგადების შექმნისას შესაძლებელია დაგვჭირდეს გადატანა ზოგიერთი ატრიბუტებისა ან ოპერაციებისა ერთი კლასიდან მეორეზე. თუ, მაგალითად დაგვჭირდა გადავიტანოთ ისინი ქვეკლასიდან სუპერკლასში,

ბრაუზერში საკმარისია უბრალოდ გადავიტანოთ ატრიბუტები და ოპერაციები ერთი კლასიდან მეორეში. ამასთან არ დაგვავიწყდეს მოვსპოთ ატრიბუტების კოპია მეორე ქვეკლასიდან, თუ ის არის.

იმისათვის, რომ მოვათავსოთ განზოგადება კლასების დიაგრამაზე:

1. დავაწკაპუნოთ ღილაკზე Generalization ინსტრუმენტების პანელიზე.
2. გავატაროთ ხაზი განზოგადება ქვეკლასიდან სუპერკლასამდე.

კავშირების სპეციფიკაცია ეხება ასოციაციის დასახელებებს, როლების დასახელებას, სიმძლავრეს და კლას – ასოციაციებს.

იმისათვის, რომ მიუთითოდ კავშირის სიმძლავრე:

1. დავაწკაპუნოთ მარჯვენა ღილაკით კავშირის ერთ ბოლოზე.
2. ავირჩიოთ პუნქტი Multiplicity გახსნილ მენიუში.
3. მიუთითოდ საჭირო სიმძლავრე.
4. გავიმეოროთ პ.1 – 3 კავშირის მეორე ბოლოსათვის.

იმისათვის, რომ მიუთითოდ კავშირის სახელი:

1. გამოვყოთ საჭირო კავშირი
2. შევიტანოთ მისი სახელი.

იმისათვის, რომ მიუთითოდ კავშირის როლური სახელი:

1. დავაწკაპუნოთ მარჯვენა ღილაკით ასოციაციაზე საჭირო ბოლოდან.
2. ავირჩიოთ პუნქტი role Name გახსნილ მენიუში.
3. ავირჩიოთ როლური დასახელება.

იმისათვის, რომ მიუთითოდ კავშირის ელემენტი (ასოციაცია – კლასი):

1. გავხსნათ საჭირო კავშირის სპეციფიკაციის ფანჯარა.
2. გადავიდეთ ჩანართზე “Detail”.
3. მიუთითედ კავშირის ელემენტი ველში Link Element.

3.საკონტროლო კითხვები

- როგორ ხდება ატრიბუტების განსაზღვრა და მათი დამატება კლასებზე;

- როგორ ხდება კლასებს შორის კავშირების (ასოციაციის) განსაზღვრა და სპეციფიკაციების (ასოციაციის დასახელება, როლების დასახელება, სიმძლავრე) დამატება;
- როგორ შეიქმნას ასოციაცია კლასების დიაგრამაზე (მიეთითოს ნავიგაცია, რეფლექსური ასოციაცია, აგრეგაცია, რეფლექსური აგრეგაცია);
- როგორ შეიქმნას კლასების განზოგადება.

4.დავალება

მოვახდინოთ გამოყენებითი შემთხვევის Close Registration ანალიზი და ავაგოთ შესაბამისი დიაგრამები ურთიერთქმედების და კლასების.

ლაბორატორიული სამუშაო №10

პარალელურად შესრულებადი მოვლენები. მართვის ნაკადების სტრუქტურის დაპროექტება

1.სამუშაოს მიზანი

სამუშაოს მიზანია შევისწავლოთ:

- ქვესისტემებისა და ინტერფეისების გამოვლენა;
- მართვის ნაკადების სტრუქტურის დაპროექტება(პარალელურად შესრულებადი მოვლენები, პროცესი, ძაფი, აქტიური კლასი).

2.მეთოდური მითითებები ლაბორატორიული სამუშაოს ჩასატარებლად

ობიექტ – ორიენტირებული დაპროექტების მიზანია წინასწარი სისტემური პროექტის (“ანალიზის” კლასების ნაკრები), რომელიც შეადგენს სისტემის არქიტექტურის სტაბილურ საფუძველს, ადაპტაცია რეალიზაციის სფეროსთან ყველა არაფუნქციონალური მოთხოვნების გათვალისწინებით.

ქვესისტემებისა და ინტერფეისების გამოვლენა

ქვესისტემების გამოვლენისას არქიტექტორის პირველ მოქმედებას წარმოადგენს “ანალიზის კლასების გარდაქმნა საპროექტო კლასებში (design classes). “ანალიზის” ყოველი კლასისათვის მიიღება ორიდან ერთი გადაწყვეტილება:

- “ანალიზის” კლასი გამოისახება საპროექტო კლასში, თუ იგი მარტივია ან წარმოადგენს ერთადერთ ლოგიკურ აბსტრაქციას;
- “ანალიზის” რთული კლასი შესაძლებელია დაიყოს რამოდენიმე კლასად, გარდაიქმნას პაკეტათ ან ქვესისტემათ.

klasebis gaerTianeba qvesistemaSi SeiZleba ganxorcieldes sxvadasxva mosazrebiT, kerZoT:

- ფუნქციონალური კავშირი: ერთიანდებიან კლასები, რომლებიც მონაწილეობენ ერთ და იმავე პრეცედენტის რეალიზებაში და ურთიერთქმედებენ მხოლოდ ერთმანეთთან;
- აუცილებლობა: კლასების ნაკრები, რომლებიც ახდენენ ერთიდაიგივე ფუნქციის რეალიზებას, რომელიც შესაძლებელია გამოვყოთ სისტემიდან ან შევცვალოთ ალტერნატიულზე;
- შეკავშირება: ქვესისტემებში აერთიანებენ მეტად დაკავშირებულ კლასებს;
- განაწილება: კლასების გაერთიანება, განლაგებულნი ქსელის კონკრეტულ კვანძზე.

შესაძლო ქვესისტემების მაგალითებია:

- კლასების ნაკრები, რომლებიც უზრუნველყოფენ ფუნქციების რთულ კომპლექს (მაგ. უშიშროება და მონაცემების დაცვა);
- მოსაზღვრე კლასები, რომლებიც ახდენენ რთულ სამომხმარებლო ინტერფეისს ან ინტერფეისს გარე სისტემებთან.

ქვესისტემების შექმნისას მოდელში სრულდება შემდეგი გარდაქმნები:

- გასაერთიანებელი კლასები თავსდებიან სპეციალურ პაკეტში ქვესისტემის დასახელებით და სტერეოტიპით „subsistem“.
- კლასების ოპერაციების სპეციფიკაციები გამოიტანება ქვესისტემის ინტერფეისში, სტერეოტიპით „interface“.

ინტერფეისის ოპერაციების გამოყენების ხასიათი და მათი შესრულების თანმიმდევრობა დოკუმენტირებულია ურთიერთქმედების დიაგრამების მეშვეობით, რომლებიც აღწერენ კლასების ურთიერთქმედებას ინტერფეისის ოპერაციების რეალიზაციისას. ეს უკანასკნელნი კი ქვესისტემის კლასების დიაგრამასთან ერთად ერთიანდებიან კოოპერაციაში ინტერფეისის დასახელებით და სტერეოტიპით „interface realization“;

ქვესისტემაში იქმნება კლასი-მოადგილე სტერეოტიპით <<subsistem proxy>>, რომელიც მართავს ინტერფეისის ოპერაციების რეალიზაციას.
ქვესისტემის ყველა ინტერფეისები უნდა იყვნენ განსაზღვრულნი არქიტექტურის დაპროექტების პროცესში, რამდენადაც ისინი გამოიყენებიან სინქრონიზაციის წერტილების სახით სისტემის პარალელური დამუშავებისას.

მართვის ნაკადების სტრუქტურის დაპროექტება

სტრუქტურის დაპროექტება სრულდება სისტემაში პარალელეური პროცესების (პარარელიზმი) არსებობისას. დაპროექტების მიზანია – სისტემაში არსებული პროცესების გამოვლენა, რეალიზაციის სფეროში მათი ურთიერთქმედების ხასიათის, შექმნის, მოსპობისა და გამოაშკარავებისათვის. პარალელიზმის მოთხოვნა აღიძვრება მაშინ როდესაც:

- საჭიროა დამუშავების განაწილება სხვადასხვა პროცესორებსა და კვანძებს შორის;
- სისტემა იმართება მოვლენათა ნაკადით (event-driven system);
- სისტემაში ერთდროულად მუშაობს მრავალი მომხმარებელი.

კურსებზე რეგისტრაციის სისტემა ფლობს პარალელიზმის თვისებას, რამდენადაც იგი უნდა უზრუნველყოფდეს მრავალი მომხმარებლის

(სტუდენტებისა და პროფესორების) ერთდროულ მუშაობას, რომელთაგან თვითეული ბადებს სისტემაში დამოუკიდებელ პროცესს.

პროცესი (process) – ეს რესურსებით უზრუნველყოფილი მართვის ნაკადია, რომელიც სრულდება სხვა პროცესების პარალელურად. მაღალი სირთულის გამო შესაძლებელია გაიყოს ორ ან მეტ ნაკადათ. ნებისმიერი კლასის ობიექტი უნდა არსებობდეს ერთი პროცესის შიგნით მაინც.

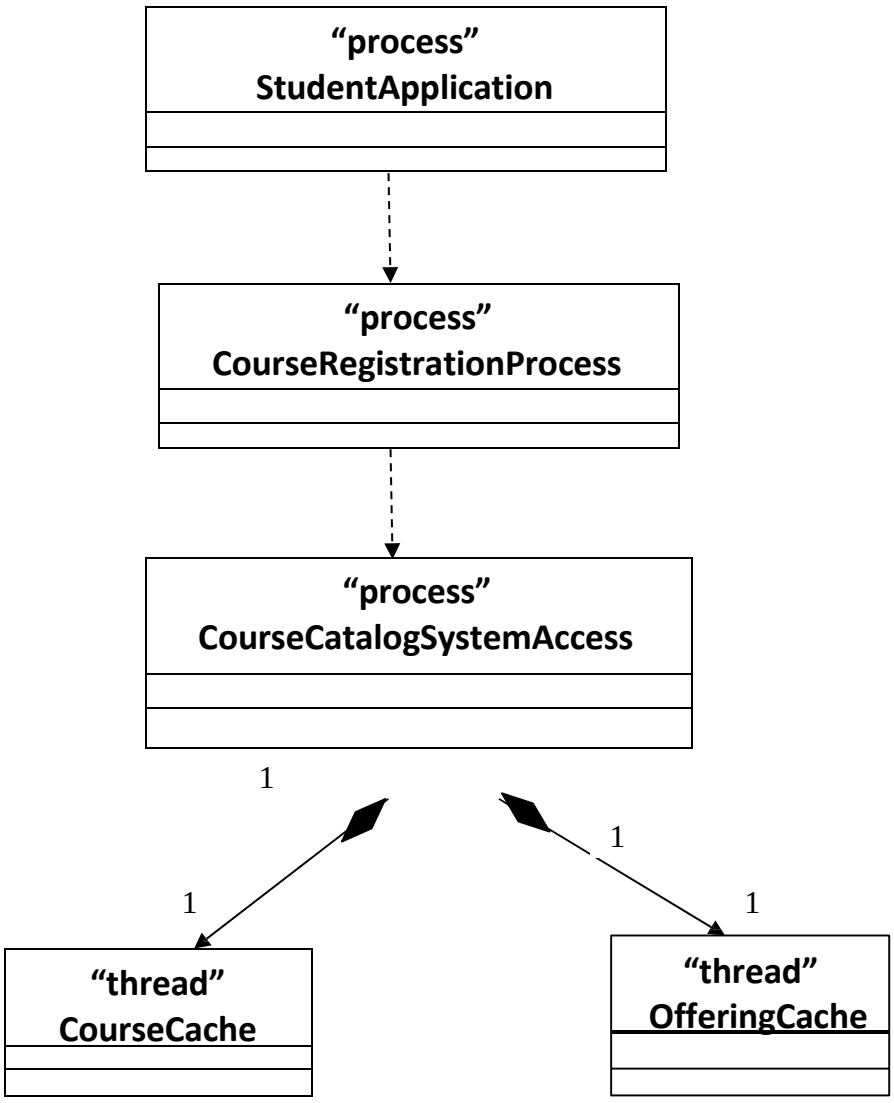
ძაფი (Thread) – ეს შედარებით მცირე მართვის ნაკადია, რომელიც სრულდება სხვა ძაფების პარალელურად ერთ და იმავე პროცესის ფარგლებში.

ნაკადების შექმნის აუცილებლობა კურსების რეგისტრაციის სისტემაში განისაზღვრება შემდეგი მოთხოვნებით:

- თუ კურსი აღმოჩნდება შევსებული იმ დროს, როდესაც სტუდენტი ახდენს სასწავლო გრაფიკის ფორმირებას, რომელიც მოიცავს მოცემულ კურსს, მაშინ იგი უნდა ინფორმირებული იყოს ამის შესახებ (საჭიროა დამოუკიდებელი პროცესი, რომელიც მართავს კონკრეტულ კურსებთან მიმართვას);
- კურსების კატალოგის არსებული მონაცემთა ბაზა ვერ უზრუნველყოფს საჭირო წარმადობას (აუცილებელია შუალედური დამუშავების პროცესი – მონაცემების ამოქაჩვა).

პროცესებისა და ნაკადების რეალიზაცია უზრუნველყოფილია ოპერაციული სისტემის საშუალებით.

მართვის ნაკადების სტრუქტურების მოდელირებისათვის გამოიყენება აქტიური კლასი – კლასი სტერეოტიპით „process“ და „thread“. აქტიური კლასი ფლობს საკუთარ პროცესს ან ნაკადს და შეუძლია მმართველი ზემოქმედების ინიციირება. კავშირები პროცესებს შორის მოდელირდება როგორც დამოკიდებულება. ნაკადები შესაძლებელია არსებობდეს მხოლოდ პროცესების შიგნით, ამიტომ კავშირები პროცესებსა და ნაკადებს შორის მოდელირდება როგორც კომპოზიცია. მართვის ნაკადების მოდელი თავსდება ლოგიკური წარმოდგენის პაკეტში Process View.

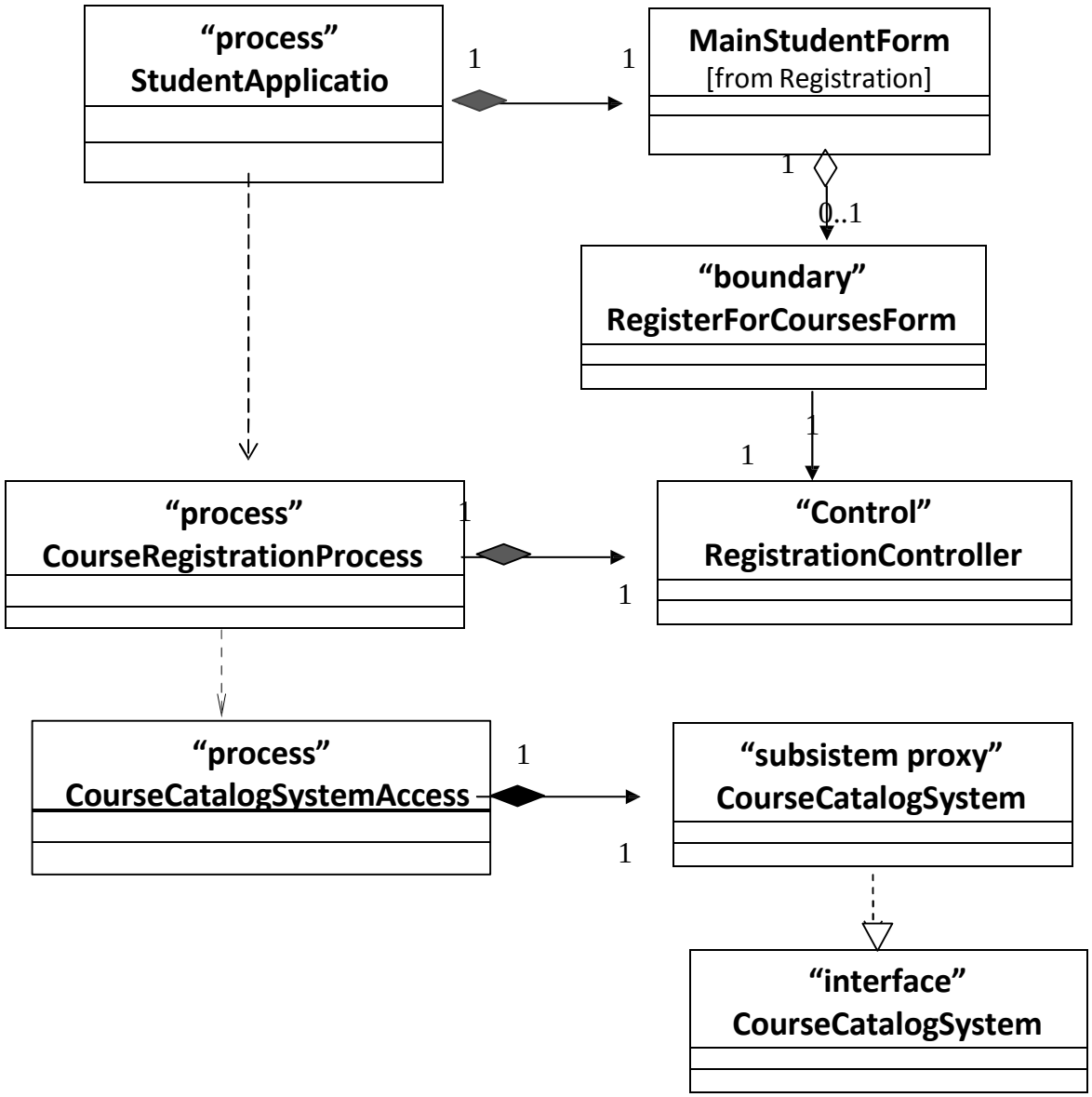


ნახ.10.1. პროცესები და ნაკადები

მაგალითისათვის ნახ.10.2-10.3..-ზე მოყვანილია კლასების დიაგრამის ფრაგმენტი, რომლებიც აღწერენ სტუდენტის კურსებზე რეგისტრაციის პროცესის სტრუქტურას. აქტიური კლასები ასრულებენ შემდეგ ფუნქციებს:

- StudentApplication - პროცესი, რომელიც მართავს სტუდენტი-მომხმარებლის ყველა ფუნქციას სისტემაში. ყოველი სტუდენტისათვის, რომელიც იწყებს რეგისტრაციას კურსებზე, იქმნება ერთი ობიექტი მოცემული კლასის;

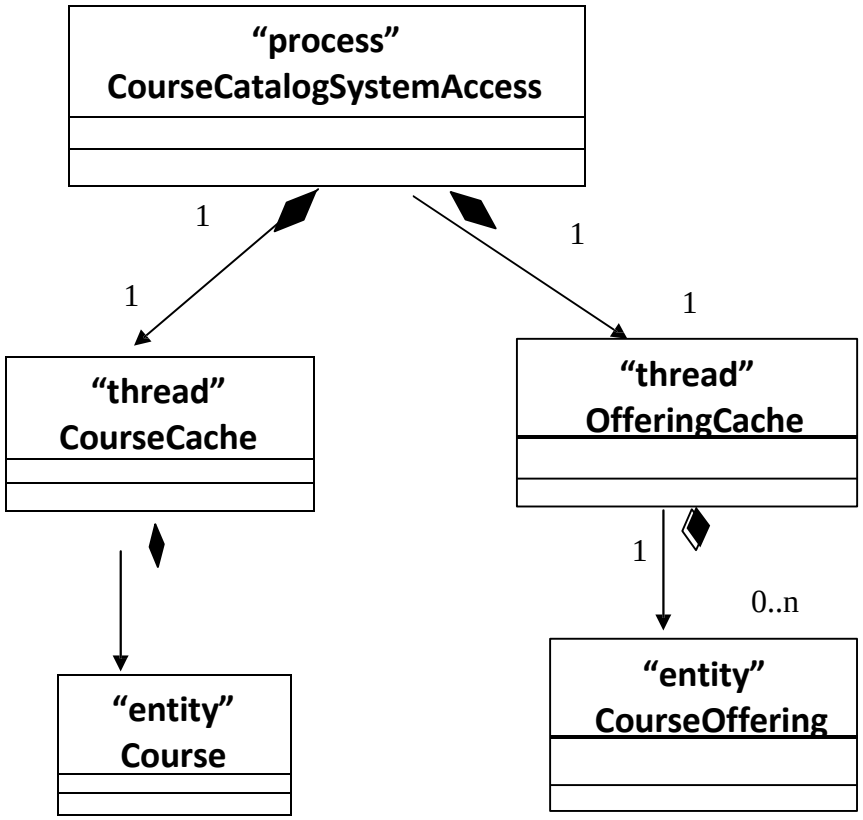
- CourseRegistrationProcess - პროცესი, რომელიც მართავს უშუალოდ სტუდენტის რეგისტრაციას. ყოველი სტუდენტისათვის, რომელიც იწყებს რეგისტრაციას კურსებზე, ასევე იქმნება ერთი ობიექტი მოცემული კლასის;



ნახ.10.2. კლასები, დაკავშირებულნი პროცესებთან

- CourseCatelogSystemAccess - მართავს კურსების კატალოგთან მიმართვას, მოცემული კლასის ერთიდაიგივე ობიექტი გამოიყენება ყველა მომხმარებლის მიერ კურსების კატალოგთან მიმართვისას.

- CourseCashe და OfferjngCashe - გამოიყენებიან მონაცემებთან ასინქრონული მიმართვისათვის მონაცემთა ბაზაში სისტემის წარმადობის გაზრდის მიზნით. ისინი წარმოადგენენ კეშ მონაცემთა ბაზიდან აღებული კურსების შესახებ მონაცემების შუალედური შენახვისათვის



ნახ.10.3. კლასები, დაკავშირებულნი ნაკადებთან

3.საკონტროლო კითხვები

- რა მოსაზრებით ხდება კლასების გაერთიანება ქვესისტემებში;
- როგორ ხდება ინტერფეისების გამოვლენა;
- რა გარდაქმნები ხდება კლასების გაერთიანებისას;

- რა შემთხვევაში და როგორ ხდება მართვის ნაკადების სტრუქტურის დაპროექტება(პარალელურად შესრულებადი მოვლენების გამოვლენა, აქტიური კლასის შექმნა).

4.დავალება

- მოვახდინოთ ქვესისტემებისა და ინტერფეისების გამოვლენა;
- მოვახდინოთ მართვის ნაკადების სტრუქტურის დაპროექტება-შევქმნათ აქტიური კლასები

ლაბორატორიული სამუშაო №11

კლასების დაპროექტება

1.სამუშაოს მიზანი

სამუშაოს მიზანია შევისწავლოთ:

- საპროექტო კლასების დეტალიზირება
- ოპერაციებისა და ატრიბუტების დაზუსტება

2.მეთოდური მითითებები ლაბორატორიული სამუშაოს ჩასატარებლად

ყოველი მოსაზღვრე კლასი გარდაიქმნება კლასების გარკვეულ ნაკრებში თავისი დანიშნულებიდან გამომდინარე. ეს შესაძლებელია იყოს მომხმარებლის ინტერფეის ელემენტების ნაკრები, დამოკიდებული დამუშავების გარემოს შესაძლებლობებზე, ან კლასების ნაკრები, რომელიც ახდენს სისტემურ ან აპარატულ ინტერფეისს.

კლასი – არსები წარმადობისა და მონაცემთა დაცვის მოსაზრებებიდან გამომდინარე შესაძლებელია დაიყოს რიგ კლასებად. საფუძველს

გაყოფისათვის წარმოადგენს კლასებში ატრიბუტების სხვადასხვა სიხშირით ან ხედვის გამოყენება. ასეთი ატრიბუტები, როგორც წესი, გამოიყოფიან ცალკე კლასებათ.

მმართველი კლასები, რომლებიც ახდენენ ინფორმაციის უბრალო გადაცემას მოსაზღვრე კლასებიდან არსებზე, შეიძლება გაძევებულ იქნან. შენარჩუნებული იქნება კლასები, რომლებიც ასრულებენ არსებით სამუშაოს მონაცემთა ნაკადების სამართავად (ტრანზაქციების მართვა, განაწილებული დამუშავება და ა.შ.).

მიღებული ამ დაზუსტებების შედეგად კლასები ექვემდებარებიან უშუალო რეალიზაციას სისტემის კოდში.

ატრიბუტებისა და ოპერაციების დაზუსტება

კლასების მოვალეობები, გარდაიქმნებიან ოპერაციებში, რომლებიც რეალიზებულნი უნდა იყვნენ კოდში. ამასთან:

- ყოველ ოპერაციას ენიჭება მოკლე სახელი, რომელიც ახასიათებს მის შედეგს;
- განისაზღვრება ოპერაციების მთელი სიგნატურა (UML ენაში მიღებული ნოტაციის შესაბამისად);
- იქმნება ოპერაციის მოკლე დასახელება, მისი ყველა პარამეტრების შინაარსის გათვალისწინებით;
- იქმნება ოპერაციების ხედვა: public, private ან proteqted;
- განისაზღვრება ოპერაციების მოქმედების არე: ეკზემპლიარი (ობიექტის ოპერაცია) ან კლასიფიკატორი (კლასების ოპერაცია);
- შესაძლებლია აღიწეროს ოპერაციის შესრულების ალგორითმი (მოღვაწეობის დიაგრამის ან ურთიერთქმედების დიაგრამის გამოყენებით).

კლასების ატრიბუტების დაზუსტება (ნახ.11.1) მდგომარეობს შემდეგში:

- ატრიბუტის დასახელების გარდა მიეთითება მისი ტიპი და მნიშვნელობა დუმილით (არააუცილებელი);

- გათვალისწინებულია შეთანხმება ატრიბუტების დასახელების შესახებ, რომელიც მიღებული იყო პროექტში და რეალიზაციის ენაში;
- მიეთითება ატრიბუტების ხედვა: public, private ან proteqted;
- აუცილებლობის შემთხვევაში განისაზღვრება წარმოებული (გამოთვლითი) ატრიბუტები.

“entity” **Student**
-name : string -address : string <<class>> - nextAailID : Int -studentId : int -dateofBirth : Date
+ getTuition() : double + addSchedule(theSchedule : Schedule) + getSchedule(forSemester : Semester) : Schedule + deleteSchedule(forSemester : Semester) + hasPrereguisites(forCourseOffering : CourseOffering) : boolean // passed(the CourseOffering : CourseOffering) : Boolean <<class>> + getNextAailID() : Int + getStudentID() : Int + getName() : string +getAddress() : string

ნახ.11.1. კლასი Student ატრიბუტებითა და ოპერაციებით

ატრიბუტებისა და ოპერაციების განსაზღვრა კლასისათვის Student

იმისათვის, რომ მიუთითოდ მონაცემის ტიპი, მნიშვნელობა დუმილით და ატრიბუტის ხედვა:

1. დავაწკაპუნოტ მარჯვენა ღილაკით ატრიბუტზე ბრაუზერში.
2. ავირჩიოთ პუნქტი Open Specification გახსნილ მენიუში.

3. მიუთითედ მონაცემის ტიპი ტიპების გახსნილ ცხრილში ან შეიტანეთ საკუთარი მონაცემის ტიპი.
4. შეიტანეთ ატრიბუტის მნიშვნელობა დუმილით ველში Initial Field (საწყისი მნიშვნელობა).
5. ავირჩიოთ ატრიბუტის ხედვა: public, private, proteqted ან Implementation ველში Export Control. დუმილით ყველა ატრიბუტების ხედვა შეესაბამება private.

იმისათვის, რომ შევცვალოთ ნოტაცია ხედვის აღნიშვნისათვის:

1. ავირჩიოთ პუნქტი Tools > Options მოდელის მენიუში.
2. გადავიდეთ ჩანართზე „Notation“.
3. მოვნიშნოთ საკონტროლო გადამრთველი Visibility as Icons, რათა გამოვიყენოთ ნოტაცია Rose, ან მოვხსნათ ნიშნული, რათა გამოვიყენოთ ნოტაცია UML.

შენიშვნა. ამ პარამეტრის ცვლილება მიგვიყვანს ნოტაციის შეცვლაზე მხოლოდ ახალი დიაგრამებისათვის და არ შეეხება არსებულ დიაგრამას.

იმისათვის, რომ მიუთითოდ დასაბრუნებელი მნიშვნელობის პარამეტრი, სტერეოტიპი ან ხედვა:

1. დავაწკაპუნოთ მარჯვენა ღილაკით ოპერაციაზე ბრაუზერში.
2. გავხსნათ ამ ოპერაციის კლასის სპეციფიკაციის ფანჯარა.
3. მიუთითედ დასაბრუნებელი მნიშვნელობის ტიპი გახსნილ ცხრილში ან შეიტანეთ თქვენი ტიპი.
4. მიუთითედ სტერეოტიპი შესაბამის გახსნად ცხრილში ან შეიტანეთ ახალი.
5. მიუთითედ ოპერაციის ხედვა ველში Export Control: public, private, proteqted ან Implementation. დუმილით ყველა ოპერაციების ხედვა შეესაბამება public.

იმისათვის, რომ დავამატოთ ოპერაციას არგუმენტი:

1. გავხსნათ ოპერაციის სპეციფიკაციის ფანჯარა.
2. გადავიდეთ ჩანართზე „Detail“.

3. დავაწკაპუნოთ მარჯვენა ღილაკით არგუმენტებზე, გახსნილ მენიუში ავირჩიოთ Insert.
4. შევიტანოთ არგუმენტის სახელი.
5. დავაწკაპუნოთ სვეტზე Data type და შევიტანოთ იქ მოცემული არგუმენტების ტიპი.
6. დავაწკაპუნოთ სვეტზე default აუცილებლობის შემთხვევაში და შევიტანოთ არგუმენტის მნიშვნელობა დუმილით.

1. საკონტროლო კითხვები

1. როგორ ხდება ატრიბუტებისა და ოპერაციების განსაზღვრა;
2. როგორ ხდება მონაცემის ტიპის და ატრიბუტის ხედვის მითითება;
3. როგორ ხდება ოპერაციაზე არგუმენტის დამატება, დასაბრუნებელი მნიშვნელობის პარამეტრის მითითება;

2. დავალება

1. დავადგინოთ ატრიბუტები და ოპერაციები კლასისათვის ;
2. მიუთითოდ მონაცემის ტიპი და ატრიბუტის ხედვა;

ლაბორატორიული სამუშაო №12

მდგომარეობების მოდელირება კლასებისათვის

1.სამუშაოს მიზანი

სამუშაოს მიზანია შევისწავლოთ:

- მდგომარეობების მოდელირება კლასებისათვის
- მდგომარეობის დიაგრამის შექმნა კლასისათვის

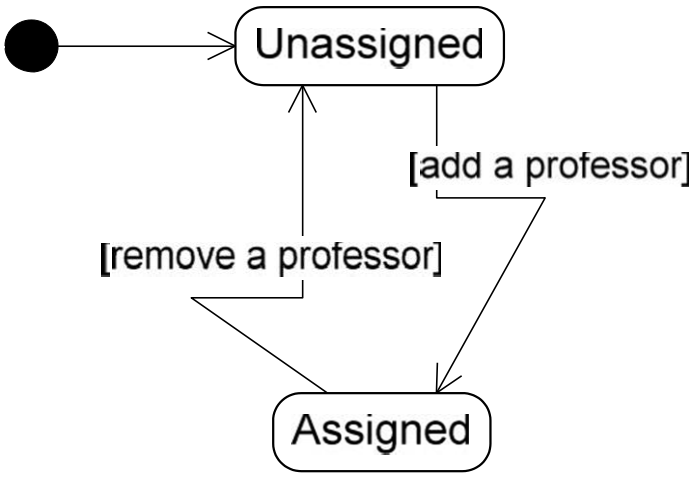
2.მეთოდური მითითებები ლაბორატორიული სამუშაოს ჩასატარებლად

თუ სისტემაში გვაქვს მდგომარეობისაგან დამოკიდებული ობიექტები ქცევის რთული დინამიკით, მაშინ მათთვის შესაძლებელია ავაგოთ მოდელი, რომელიც აღწერს ობიექტის მდგომარეობებს და გადასვლებს მათ შორის. ეს მოდელი წარმოსდგება მდგომარეობის დიაგრამის სახით.

მაგალითისატვის განვიხილოთ კლასი CourseOffering – ის ობიექტის ქცევა. მდგომარეობის დიაგრამა იგება რამოდენიმე ეტაპად.

1 ეტაპი. მდგომარეობების იდენტიფიცირება. მდგომარეობათა გამოვლენის ნიშნება ობიექტის ატრიბუტების მნიშვნელობების შეცვლა და კავშირების გაწყვეტა სხვა ობიექტებთან. ასე მაგალითად, ობიექტი CourseOffering შეიძლება იმყოფებოდეს მდგომარეობაში Open (კურსზე მიღება გახსნილია) მანამდე, სანამ მასზე დარეგისტრირებულ სტუდენტთა არ გადააჭარბებს 10-ს, ხოლო როგორც კი გადააჭარბებს 10-ს, ობიექტი გადადის მდგომარეობაში Closed (კურსზე მიღება დახურულია). ამას გარდა, ობიექტი CourseOffering შესაძლებელია იმყოფებოდეს მდგომარეობაში Unassigned (იგი არავის არ მიყავს ანუ არ არსებობს კავშირი Professor – ის რომელიმე ობიექტთან) ან Assigned (ასეთი კავშირი არსებობს).

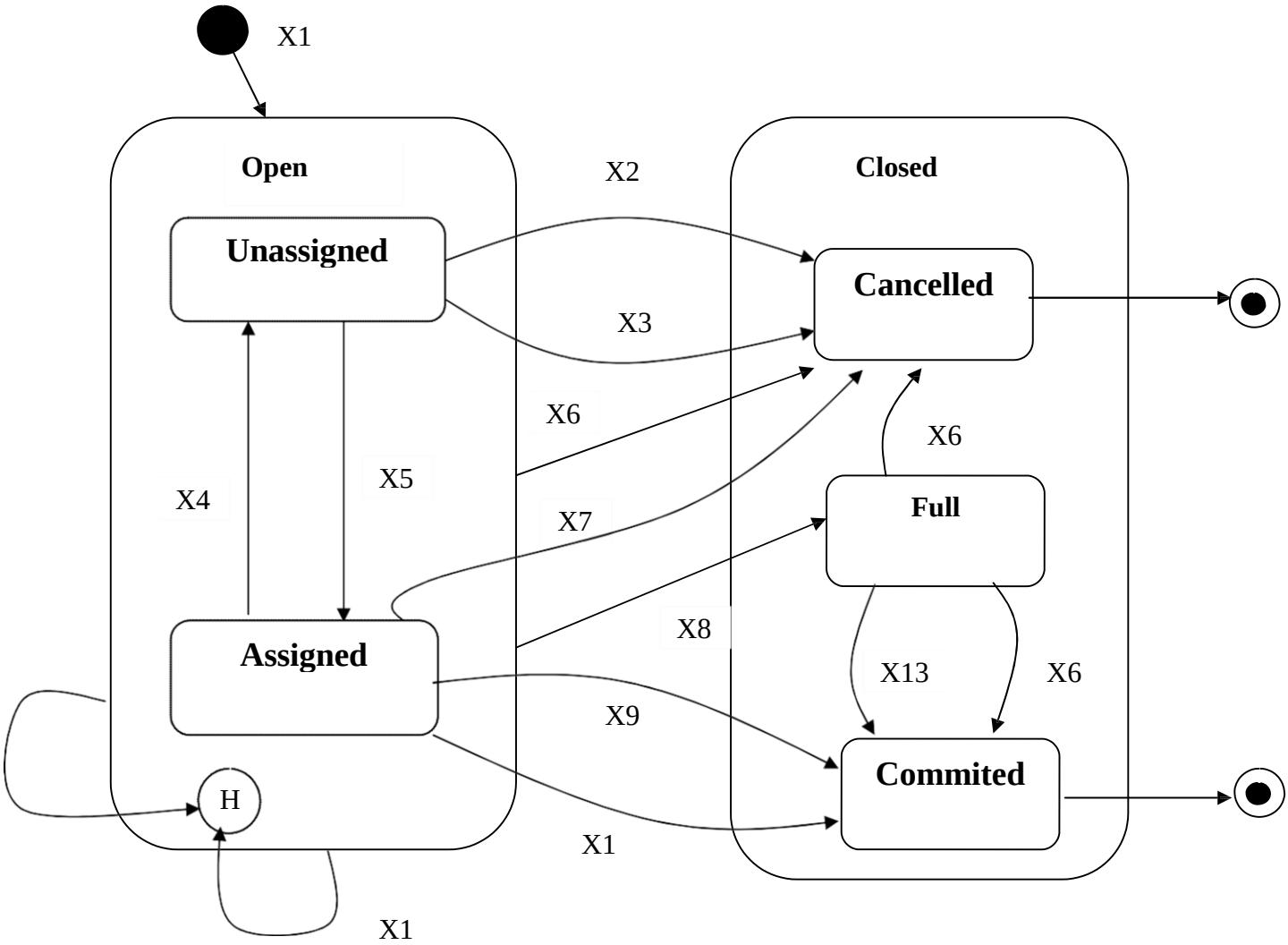
ეტაპი 2. მოვლენათა იდენტიფიცირება. მოვლენები, როგორც წესი დაკავშირებული არიან გარკვეული ოპერაციების შესრულებასთან. კლასი Course Offering მოვალეობების განაწილების შედაგად გამოყენებითი შემთხვევა „ავირჩიოთ კურსები სწავლებისათვის“ ანალიზისას განისაზღვრა ორი ოპერაცია – addProfessor და removeProfessor, დაკავშირებულნი გარკვეული პროფესორის მიერ კურსების არჩევასთან (ახალი კავშირის შექმნა) და არჩეული კურსზე უარის თქმა (კავშირის გაწყვეტა). ამ ოპერაციებს შეუსაბამებენ ორ მოვლენას – addProfessor და removeprofessor.



ნახ.12.1. გადასვლები მდგომარეობებს შორის

ეტაპი 3. მდგომარეობებს შორის გადასვლების იდენტიფიცირება. გადასვლები გამოიძახებიან მოვლენებით. მაშასადამე, Unassigned და Assigned ერთდებიან ორი გადასვლით (ნახ.12.1)

.CourseOfering ობიექტის ქცევის შემდგომი დეტალიზებას მივყავართ მდგომარეობის დიაგრამის აგებამდე (ნახ.12.2). მოცემულ დიაგრამაზე გამოყენებულია მოდელირების ისეთი შესაძლებლობები, როგორიც არის კომპოზიციური (composite state) და ისტორიული მდგომარეობები (history state). მოცემულ შემთხვევაში კომპოზიციური მდგომარეობებია Open და Closed, ხოლო ჩართული მდგომარეობებია – Unassigned, Assigned, Cancelled (კურსი გაუქმდა), Full (კურსი შევსებულია) და Committed (კურსი ჩართულია განრიგში). კომპოზიციურ მდგომარეობები საშუალებას იძლევიან გავამარტივოთ დიაგრამა, შევამციროთ რა გადასვლების რაოდენობა, რამდენადაც ჩართული მდგომარეობები მემკვიდრეობით იძენენ კომპოზიციური მდგომარეობების ყველა თვისებებსა და გადასვლებს.



ნახ.12.2. მდგომარეობის დიაგრამა კომპოზიციური მდგომარეობებით

ისტორიული მდგომარეობა – ეს პსევდომდგომარეობაა, რომელიც აღადგენს წინამდებარე აქტიურ მდგომარეობას კომპოზიტურ მდგომარეობაში. იგი საშუალებას აძლევს კომპოზიტურ მდგომარეობას Open დაიმახსოვროს რომელი ჩართული მდგომარეობა (Unassigned ან Assigned) იყო მიმდინარე Open – დან გამოსვლის მომენტში, იმისათვის რომ ნებისმიერი გადასვლა Open – ში (add student an remove student) ბრუნდებოდეს სწორედ ამ ჩართულ მდგომარეობაში, და არა საწყის მდგომარეობაში.

მდგომარეობის დიაგრამის შექმნა კლასისათვის CourseOffering

იმისათვის, რომ შევქმნათ მდგომარეობის დიაგრამა:

1. დავაწკაპუნოთ მარჯვენა ღილაკით ბრაუზერში საწირო კლასზე.

2. ავირჩიოთ პუნქტი New > Statechart diagram გახსნილ მენიუში.

იმისათვის, რომ დავამატოთ მდგომარეობა:

1. დავაწკაპუნოთ ღილაკზე State ინსტრუმენტების პანელზე.
2. დავაწკაპუნოთ მდგომარეობის დიაგრამაზე იმ ადგილზე, სადაც გსურთ მისი მოთავსება.

მდგომარეობის ყველა ელემენტები შესაძლებელია დავამატოთ ჩანართი „Detail“ საშუალებით მდგომარეობის სპეციფიკაციის ფანჯარაში.

იმისათვის, რომ დავამატოთ მოქმედება:

1. გავხსნათ საჭირო მდგომარეობის სპეციფიკაციის ფანჯარა.
2. გადავიდეთ ჩანართზე „Detail“.
3. დავაწკაპუნოთ მარჯვენა ღილაკით ფანჯარაში „Actions“.
4. ავირჩიოთ პუნქტი Insert გახსნილ მენიუში.
5. დავაწკაპუნოთ ორჯერ ახალ მოქმედებაზე.
6. შევიტანოთ მოქმედება ველში Actions.
7. მიუთითოდ Do ფანჯარაში „When“ იმისათვის, რომ გავხადოთ ახალი მოქმედება მოღვაწეობათ.

იმისათვის, რომ დავამატოთ შემავალი მოქმედება, ფანჯარაში „When“ მიუთითოდ On Entry.

იმისათვის, რომ დავამატოთ გამომავალი მოქმედება, ფანჯარაში „When“ მიუთითოდ On Exit.

იმისათვის, რომ გავგზავნოთ მოვლენა:

1. გავხსნათ საჭირო მდგომარეობის სპეციფიკაციის ფანჯარა.
2. გადავიდეთ ჩანართზე „Detail“.
3. დავაწკაპუნოთ მარჯვენა ღილაკით ფანჯარაში „Actions“.
4. ავირჩიოთ პუნქტი Insert გახსნილ მენიუში.
5. დავაწკაპუნოთ ორჯერ ახალ მოქმედებაზე.
6. მიუთითოდ Send Event მოქმედების ტიპის სახით .

7. შევიტანოთ მოვლენა (Event) , არგუმენტები და მიზნობრივი ობიექტი (Target) შესაბამის ველებში.

იმისათვის, რომ დავამატოთ გადასვლა:

1. დავაწკაპუნოთ ღილაკზე Transition ინსტრუმენტების პანელზე.
2. დავაწკაპუნოთ მდგომარეობაზე, საიდანაც ხორციელდება გადასვლა.
3. გავატაროთ გადასვლის ხაზი იმ მდგომარეობამდე, სადაც ის მთავრდება.

იმისათვის, რომ დავამატოთ რეფლექსური გადასვლა:

1. დავაწკაპუნოთ ღილაკზე Transition to Self ინსტრუმენტების პანელზე.
2. დავაწკაპუნოთ მდგომარეობაზე, საიდაც ხორციელდება რეფლექსური გადასვლა.

იმისათვის, რომ დავამატოთ მოვლენა, მისი არგუმენტები, შემზღუდავი პირობა და მოქმედება:

1. დავაწკაპუნოთ ორჯერ გადასვლაზე, რათა გაიხსნას მისი სპეციფიკაციის ფანჯარა.
2. გადავიდეთ ჩანართზე „General“.
3. შევიტანოთ მოვლენა ველში Event.
4. შევიტანოთ არგუმენტები ველში Arguments.
5. შევიტანოთ შემზღუდავი პირობა ველში Condition.
6. შევიტანოთ მოქმედება ველში Action.

იმისათვის, რომ გავგზავნოთ მოვლენა:

1. დავაწკაპუნოთ ორჯერ გადასვლაზე, რათა გაიხსნას მისი სპეციფიკაციის ფანჯარა.
2. გადავიდეთ ჩანართზე „General“.
3. შევიტანოთ მოვლენა ველში Send Event.
4. შევიტანოთ არგუმენტები ველში Send Arguments.
5. მიუთითოდ მიზანი ველში Send Target.

იმისათვის, რომ მიუთითოდ საწყისი ან საბოლაო მდგომარეობა:

1. დავაჭიროთ ღილაკზე Start State ან End State ინსტრუმენტების პანელზე.
2. დავაწკაპუნოთ მდგომარეობის დიაგრამაზე იმ ადგილზე, სადაც გსურთ მისი მოთავსება.

3.საკონტროლო კითხვები

1. რისთვის გამოიყენება მდგომარეობის დიაგრამა;
2. როგორ დავამატოთ მდგომარეობა, მოქმედება;
3. როგორ გავგზავნოთ მოვლენა, დავამატოთ გადასვლა.

4.დავალება

1. მოახდინეთ მდგომარეობების მოდელირება კლასებისათვის
2. ავაგოთ მდგომარეობის დიაგრამა კლასისათვის

ლაბორატორიული სამუშაო №13

კლასების დაპროექტება. კლასებს შორის კავშირების დაზუსტება

1.სამუშაოს მიზანი

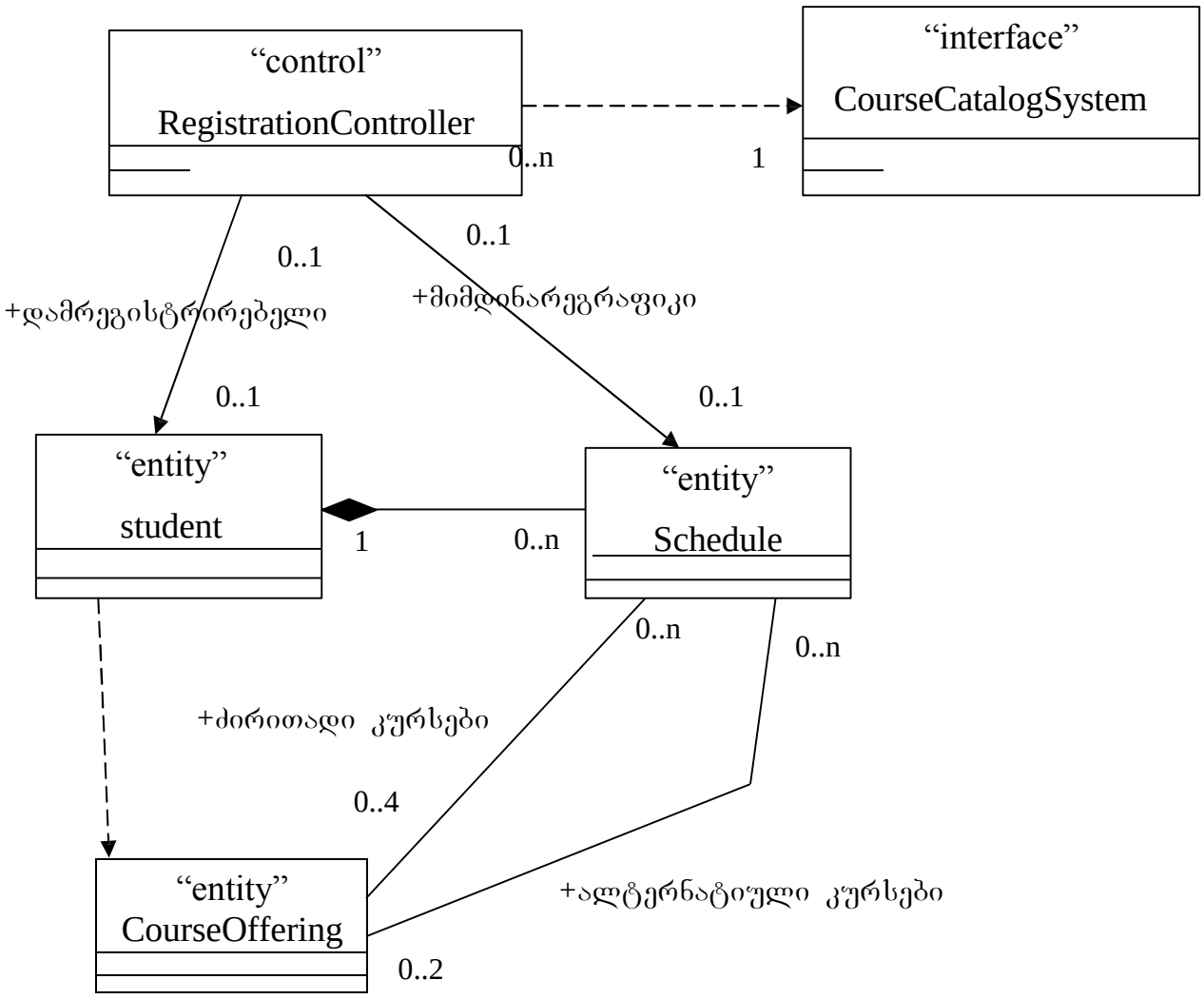
სამუშაოს მიზანია შევისწავლოთ:

- კავშირების დაზუსტება კლასებს შორის (ასოციაცია, აგრეგაცია და განზოგადება);
- აგრეგაციის გარდაქმნა კომპოზიციაში;

2.მეთოდური მითითებები ლაბორატორიული სამუშაოს ჩასატარებლად

პროექტირების პროცესში კავშირები კლასებს შორის (ასოციაცია, აგრეგაცია და განზოგადება) მოითხოვს დაზუსტებას:

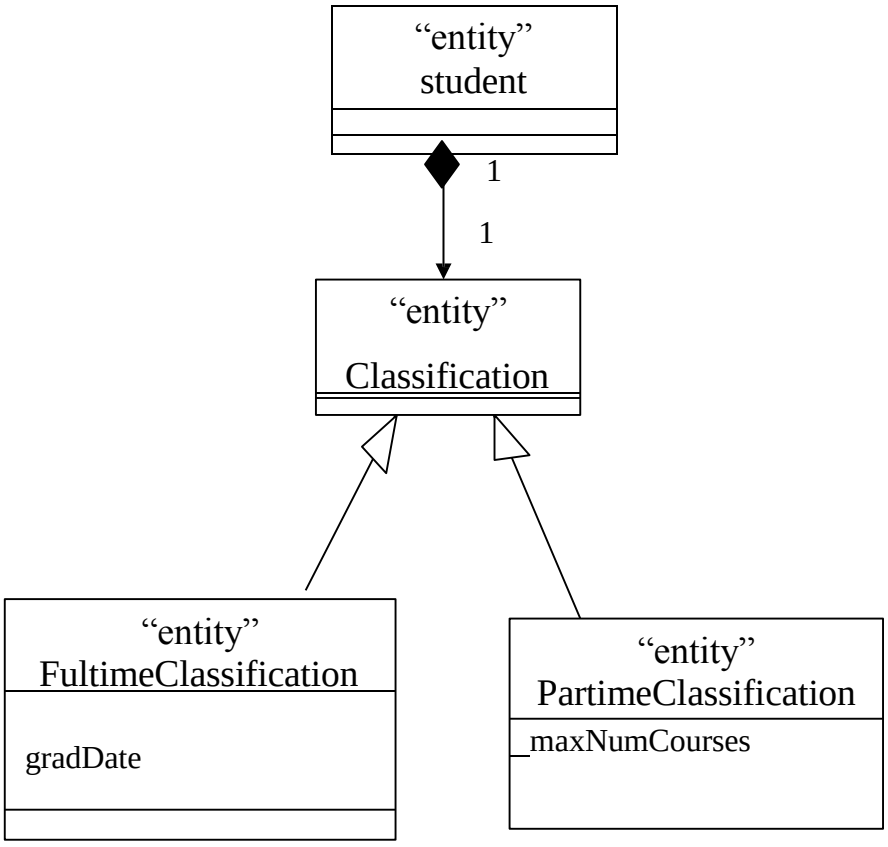
- ასოციაცია მოსაზღვრე და მმართველ კლასებს შორის გამოხატავენ კავშირებს, რომლებიც დინამიურად აღიძვრებიან შესაბამის ობიექტებს შორის მართვის ნაკადში. ასეთი კავშირებისათვის საკმარისია უზრუნველვყოთ კლასების ხედვა, ამიტომ ისინი გარდაიქმნებიან დამოკიდებულებაში;
- თუ ზოგიერთი ასოციაციისათვის არ არის აუცილებელი ორმიმართულებიანი კავშირისა, მაშინ შემოიტანება ნავიგაციის მიმართულება;
- აგრეგაციები, რომლებსაც აქვთ კომპოზიციის თვისებები, გარდაიქმნებიან კომპოზიციურ კავშირებში.



ნახ. 13.1. ასოციაციებისა და აგრეგაციების გარდაქმნის მაგალითი

კავშირების გარდაქმნის მაგალითი მოყვანილი რეკომენდაციების შესაბამისად კლასებისათვის გამოყენებითი შემთხვევიდან „კურსებზე დარეგისტრირება“ მოყვანილია ნახ.13.2.-ზე. ასოციაცია მმართველ და მოსაზღვრე კლასებს შორის გარდაიქმნა დამოკიდებულებათ. აგრეგაცია კლასებს შორის Student და Schedule ფლობს კომპოზიციის თვისებას. ნავიგაციის მიმართულება ასოციაციებზე კლასებს შორის Schedule და CourseOffering შემოტანილია შემდეგი მოსაზრებებით: არ არის აუცილებელი გრაფიკების ცხრილის მიღება, რომლებშიც არის რომელიმე კურსი; გრაფიკების რაოდენობა შედარებით ცოტაა კონკრეტული კურსების რაოდენობასთან შედარებით.

შენიშვნა. მნიშვნელობა By Value გულისხმობს, რომ მთელი ნაწილი იქმნებიან და ისპობიან ერთდროულად, რაც შეესაბამება კომპოზიციას. აგრეგაცია (By Reference) გულისხმობს, რომ მთელი ნაწილი იქმნებიან და ისპობიან სხვადასხვა დროს.



ნახ. 13.2. განზოგადების გარდაქმნა

განზოგადების კავშირები შესაძლებელია გარდაიქმნან მეტამორფოზული ქვეტიპების სიტუაციებში. მაგალითად, რეგისტრაციის სისტემის შემთხვევაში სტუდენტს შეუძლია გადავიდეს დღის სწავლებიდან საღამოზე ე.ი. ობიექტ Students შეუძლია შეცვალოს თავისი ქვეტიპი. ასეთი ცვლილებისას მოგვიხდება ობიექტის აღწერის მოდიფიცირება სისტემაში. იმისათვის, რომ თავი ავარიდოთ ასეთ მოდიფიკაციას და ამით გავზარდოთ სისტემის მდგრადობა, მემკვიდრეობის იერარქია რეალიზდება კლასიფიკაციის საფუძველზე.

1. საკონტროლო კითხვები

1. რატომ არის საჭირო კავშირების დაზუსტება კლასებს შორის (ასოციაცია, აგრეგაცია და განზოგადება);
2. როგორ მოვახდინოთ აგრეგაციის გარდაქმნა კომპოზიციაში;

2. დავალება

1. მოვახდინოთ კავშირების დაზუსტება კლასებს შორის (ასოციაცია, აგრეგაცია და განზოგადება);
2. მოვახდინოთ აგრეგაციის გარდაქმნა კომპოზიციაში.

ლაბორატორიული სამუშაო №14

მონაცემთა ბაზების დაპროექტება

1.სამუშაოს მიზანი

სამუშაოს მიზანია შევისწავლოთ:

- მონაცემთა ბაზების დაპროექტება;
- რელაციური ბაზის დაპროექტება.
- მონაცემთა ბაზის აღწერის გენერაცია SQL ენაზე;

2.მეთოდური მითითებები ლაბორატორიული სამუშაოს ჩასატარებლად

მონაცემთა ბაზების დაპროექტება დამოკიდებულია იმაზე, თუ რა მბმს გამოიყენება მონაცემთა შენახვისათვის – ობიექტური თუ რელაციური. ობიექტური მბ არავითარი დაპროექტება არ არის საჭირო, რამდრნადაც კლასი არსები უშუალოდ აისახებიან მბ-ში. რელაციური მბ კლასი არსები ობიექტური მოდელისა უნდა გამოსახულნი იქნან რელაციურ მბ-ში. ცხრილების და მათ შორის კავშირების ერთობლიობა შესაძლებელია წარმოდგენილი იქნან კლასების დიაგრამის სახით, რომელიც წარმოადგენს ფაქტიურად ER – დიაგრამას. წესების ერთობლიობა, რომლებიც მიიღება კლასების მბ-ს ცხრილებში გამოსახვისას, ფაქტიურად ეთანხმებიან მბ-ს ცხრილებში არსებისა და კავშირების გარდაქმნის წესებს. RUP ტექნოლოგიაში ასეთი გამოსახვისათვის გამოიყენება სპეციალური ინსტრუმენტი – Data Modeler. იგი ასრულებს გარდაქმნას კლას-არსებისა კლას-ცხრილებში მისი შემდომი გენერაციით მბ აღწერისა როგორც სტანდარტულ ენაზე SQL (ANSI SQL), ასევე მის დიალექტებზე სხვადასხვა მბმს (Oracle, IBM DB2, Sybase Adaptive Server, MS SQL Server). მბ-ს სქემის

აღწერისათვის გამოიყენება UML ენის შემდეგი ელემენტების ნაკრები თავიანთი სტერეოტიპებით:

- ცხრილი წარმოიდგინება კლასის სახით სტერეოტიპით <<Table>>;
- წარმოდგენა გამოისახება კლასის სახით სტერეოტიპით <<View>>;
- ცხრილის სვეტი წარმოიდგინება კლასის ატრიბუტის სახით შესაბამისი მონაცემთა ტიპით;
- ჩვეულებრივი ასოციაცია და აგრეგაცია წარმოიდგინება ასოციაციის სახით სტერეოტიპით <<Non-Identifying>>;
- კომპოზიცია წარმოიდგინება ასოციაციის სახით სტერეოტიპით << Identifying>>;
- მბ-ის სქემა წარმოიდგინება პაკეტის სახით სტერეოტიპით <<Schema>>, რომელიც შეიცავს კლასებს-ცხრილებს;
- შესანახი პროცედურების კონტეინერი წარმოიდგინება კლასის სახით სტერეოტიპით <<SP Container>>;
- მთლიანობის შეზღუდვა, ინდექსები და ტრიგერები წარმოიდგინება კლასი-ცხრილების ოპერაციის სახით სტერეოტიპით <<PK>>, <<FK>>, <<Unique>>, <<Check>>,<<Index>> და <<Trigger>>;
- ფიზიკური მონაცემთა ბაზა წარმოიდგინება კომპონენტის სახით სტერეოტიპით <<Database>>.

რელაციური ბაზის დაპროექტება

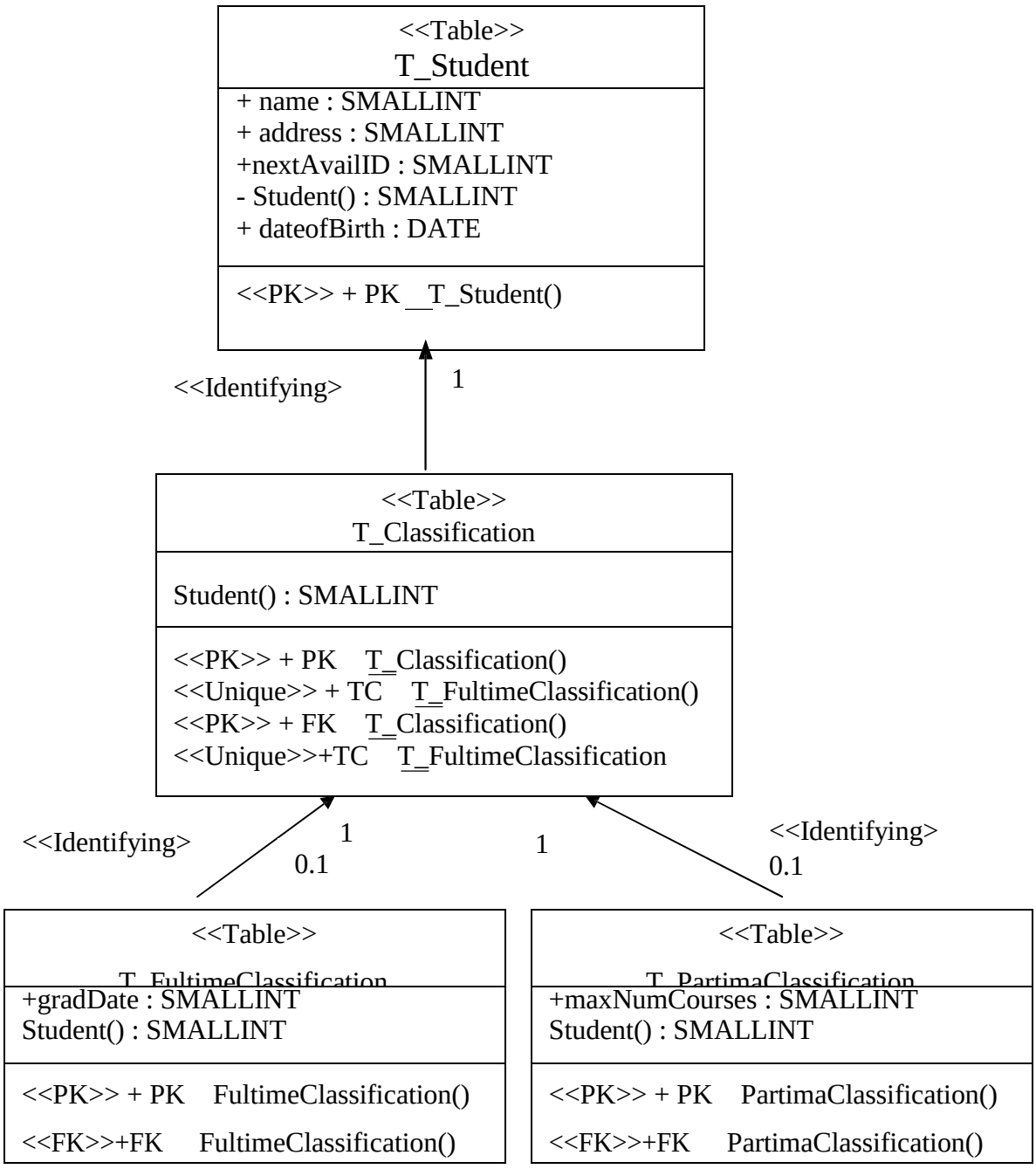
მბ დაპროექტება შედგება შემდეგი ბიჯებისაგან.

ბიჯი 1. ახალი კომპონენტის შექმნა – მონაცემთა ბაზის;

1. დავაწკაპუნოთ მარჯვენა ღილაკით კომპონენტების წარმოდგენაზე.
2. ავირჩიოთ პუნქტი Data Modeler > new > Database გახსნილ მენიუში.
3. გავხსნათ სპეციფიკაციის ფანჯარა ახლად შექმნილი კომპონენტის DB_0 და სიაში Target ავირჩიოთ Oracle *.x.

ბიჯი 2. მდგრადი კლასების განსაზღვრა:

1. გავხსნათ სპეციფიკაციის ფანჯარა კლასის Student პაკეტში University Artifacts.
2. გადავიდეთ ჩანართზე „Detail“.



ნახ.14.1.

3. დიაგრამა „არსი-კავშირი“ დავაყენოთ გადამრთველის მნიშვნელობა Persistence Persistent-ზე.

4. ჩავატაროთ ასეთივე მოქმედებები სხვა კლასებზე.
5. გავხსნათ კლასი Student ბრაუზერში, დავაჭიროთ „+“.
6. დავაწკაპუნოთ მარჯვენა ღილაკით ატრიბუტზე studentID.
7. ავირჩიოთ პუნქტი Data Modeler > Part of Object Identity გახსნილ მენიუში.

ბიჯი 3. მბ-ს სქემის შექმნა:

1. დავაწკაპუნოთ მარჯვენა ღილაკით პაკეტზე University Artifacts.
2. ავირჩიოთ პუნქტი Data Modeler > Transform to Data Model გახსნილ მენიუში.
3. მიუთითედ DB_0 და დავაწკაპუნოთ ღილაკზე OK გახსნილ ფანჯარაში სიაში Target Database. შედეგად ლოგიკურ წარმოდგენაში გაჩნდება ახალი პაკეტი Schemas.
4. გავხსნათ პაკეტი Schemas და დავაწკაპუნოთ მარჯვენა ღილაკით პაკეტზე <<Schema>> S_0.
5. ავირჩიოთ პუნქტი Data Modeler > New > Data Model Diagram გახსნილ მენიუში.
6. გავხსნათ პაკეტი, შემდეგ ახლად შექმნილი დიაგრამა „არსი-კავშირი“ NewDiagram და გადავიტანოთ მასზე ყველა კლასი-ცხრილები, რომლებიც იმყოფებიან პაკეტში <<Schema>> S_0. მიღებული დიაგრამა ნაჩვენებია ნახ.5.16.-ზე.

მონაცემთა ბაზის დაპროექტების შემდეგ შესაძლებელია მონაცემთა ბაზის აღწერის გენერირება SQL ენაზე.

მბ აღწერის გენერაციისათვის:

1. დავაწკაპუნოთ მარჯვენ ღილაკით პაკეტზე <<Schema>>S_0.
2. ავირჩიოთ პუნქტი Data Modeler > Forward Engineer გახსნილ მენიუში.
3. დავაწკაპუნოთ ღილაკზე Next ოსტატის გახსნილ ფანჯარაში „Forward Engineering Wizard“

4. მონაცემთა აღწერის ენის (DDL) გენერაციის ყველა დროშები დავტოვოთ მონიშნული და დავაწკაპუნოთ ღილაკზე Next.
5. მიუთითოდ დასახელება და ტექსტური ფაილის განლაგება გენერაციის შედეგებით და დავაწკაპუნოთ ღილაკზე Next.
6. გენერაციის დამთავრებისთანავე, გახსენით შექმნილი ტექსტური ფაილი და ნახეთ შედეგები.

გენერაციის შედეგები უნდა გამოიყურებოდნენ შემდეგნაირად:

. Результаты генерации должны выглядеть следующим образом:

```
CREATE TABLE T_ParttimeClassification (
maxNumCourses SMALLINT NOT NULL,
studentID SMALLINT NOT NULL,
CONSTRAINT PK_T_ParttimeClassification29 PRIMARY KEY
(studentID)
)
/
CREATE TABLE T_Classification (
studentID SMALLINT NOT NULL,
CONSTRAINT PK_T_Classification23 PRIMARY KEY
(studentID),
CONSTRAINT TC_T_Classification44 UNIQUE (studentID)
)
/
```

```
    CREATE TABLE T_FulltimeClassification (
    gradDate SMALLINT NOT NULL,
    studentID SMALLINT NOT NULL,
    CONSTRAINT PK_T_FulltimeClassification28 PRIMARY KEY
(studentID)
    )
    /
    CREATE TABLE T_Student (
    name SMALLINT NOT NULL,
    address SMALLINT NOT NULL,
    nextAvailID SMALLINT NOT NULL,
    studentID SMALLINT NOT NULL,
    dateofBirth DATE NOT NULL,
    CONSTRAINT PK_T_Student25 PRIMARY KEY (studentID)
    )
    /
    CREATE INDEX TC_T_Classification43 ON T_Classification
(studentID)
    /
    ALTER TABLE T_ParttimeClassification ADD (CONSTRAINT
FK_T_ParttimeClassification29 FOREIGN KEY (studentID) REFER-
ENCES T_Classification (studentID))
    /
    ALTER TABLE T_Classification ADD (CONSTRAINT
FK_T_Classification23 FOREIGN KEY (studentID) REFERENCES
T_Student (studentID))
    /
    ALTER TABLE T_FulltimeClassification ADD (CONSTRAINT
FK_T_FulltimeClassification28 FOREIGN KEY (studentID) REFER-
ENCES T_Classification (studentID))
    /
```

3.საკონტროლო კითხვები

- როგორ ხორციელდება მონაცემთა ბაზების დაპროექტება;
- როგორ მოვახდინოთ რელაციური ბაზის დაპროექტება.

- როგორ მოვახდინოთ მონაცემთა ბაზის აღწერის გენერაცია SQL ენაზე;

4.დავალება

- მოვახდინოთ მონაცემთა ბაზის აღწერის გენერაცია SQL ენაზე;
- მოვახდინოთ მონაცემთა ბაზების(რელაციური ბაზის) დაპროექტება.

ლაბორატორიული სამუშაო №15
კოდის გენერაცია
1.სამუშაოს მიზანი

სამუშაოს მიზანია შევისწავლოთ:

- მოდელის გასინჯვა;
- კომპონენტების შექმნა;
- კლასების შესაბამისობა კომპონენტებთან;
- გენერაციის კოდის თვისებები;
- დანართის კოდის გენერაცია.

2.მეთოდური მითითებები ლაბორატორიული სამუშაოს ჩასატარებლად

დანართის კოდის გენერაციის პროცესი შედგება შემდეგი ბიჯებისაგან:

ბიჯი 1. მოდელის გასინჯვა.

ბიჯი 2. კომპონენტების შექმნა.

ბიჯი 3. კლასების შესაბამისობა კომპონენტებთან.

ბიჯი 4. გენერაციის კოდის თვისებების დაყენება.

ბიჯი 5. კლასის, კომპონენტის ან პაკეტის ამორჩევა.

ბიჯი 6. კოდის გენერაცია.

მოდელის გასინჯვა

ROSE-ში არსებობს მოდელის შემოწმების საშუალება, რომელიც ენისაგან დამოუკიდებლად გამოიყენება მოდელის კორექტულობის უზრუნველსაყოფად კოდის გენერაციის წინ. რეკომენდირებულია ყოველთვის ასეთი შემოწმების შესრულება, რამდენადაც იგი გვეხმარება გამოვავლინოთ მოდელში უზუსტობანი და შეცდომები, რომლებიც არ გვაძლევენ სათანადოთ კოდის გენერირების საშუალებას.

მოდელის შემოწმებისათვის:

1. ავირჩიოთ მენიუში Tools > Check Model.
2. გავაანალიზოთ ყველა შეცდომები, რომლებიც გამოჩნდება ჟურნალის ფანჯარაში.

ყველაზე გავრცელებული შეცდომებია, მაგალითად, შეტყობინება მიმდევრობის ან კოოპერაციის დიაგრამაზე არ არის გამოსახული ოპერაციაზე, ან ამ დიაგრამის ობიექტები, არ არის ასახული კლასზე.

მენიუს პუნქტით Check Model შეიძლება გამოვავლინოთ უზუსტობისა და შეცდომების უმეტესი ნაწილი მოდელში. მენიუს პუნქტი Access Violations საშუალებას იძლევა დავადგინოთ დარღვევები მიმართვის წესებში, რომლებიც აღიძვრება მაშინ როდესაც არსებობს კავშირი ორ კლასს შორის სხვადასხვა პაკეტიდან, მაგრამ კავშირი თვით პაკეტებს შორის არ არის.

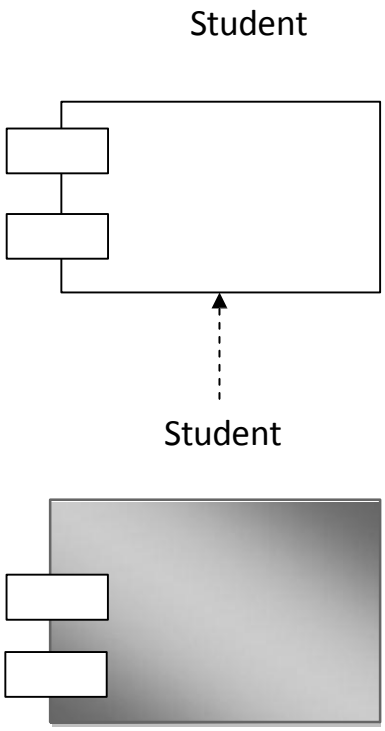
იმისათვის, რომ აღმოვაჩინოთ დარღვევები მიმართვის წესებში:

1. ავირჩიოთ მენიუში Report > Show Access Violations.
2. გავაანალიზოთ ყველა შეცდომები, რომლებიც გამოჩნდება ჟურნალის ფანჯარაში.

კომპონენტების შექმნა და კლასების შესაბამისობა კომპონენტებთან

ROSE-ში კომპონენტების დიაგრამები იქმნება კომპონენტების წარმოდგენაში. ცალკეული კომპონენტები შეიძლება შევქმნათ უშუალოდ დიაგრამაზე ან გადავიტანოთ იქ ბრაუზერიდან.

ავირჩიოთ დაპროგრამების ენათ C++ და შევქმნათ კლასი Student ამ ენის შესაბამისი კომპონენტები.



ნახ.15.1. კომპონენტების დიაგრამა

კომპონენტების დიაგრამის შესაქმნელად:

1. დავაწკაპუნოთ ორჯერ კომპონენტების მთავარ დიაგრამაზე Main კომპონენტების წარმოდგენაში.
2. დავაწკაპუნოთ ღილაკზე Paskage Specification ინსტრუმენტების პანელზე.
3. მოვათავსოთ ელემენტი Paskage Specification დიაგრამაზე.
4. გავხსნათ ელემენტის სპეციფიკაცია, შევიტანოთ სახელო Student და მიუთითოდ ფანჯარაში ენა ANSI C++.
5. დავაწკაპუნოთ ღილაკზე Paskage Body ინსტრუმენტების პანელზე.
6. მოვათავსოთ ელემენტი Paskage Body დიაგრამაზე.
7. გავხსნათ ელემენტის სპეციფიკაცია, შევიტანოთ სახელო Student და მიუთითოდ ფანჯარაში ენა ANSI C++.

8. დავაწკაპუნოთ ღილაკზე Dependence ინსტრუმენტების პანელზე.
9. გავატაროთ დამოკიდებულების ხაზი პაკეტის ტანიდან პაკეტის სპეციფიკაციასთან.

კლასის კომპონენტთან შესაბამისობისათვის:

3. ვიპოვოთ კლასი Student ბრაუზერის ლოგიკურ წარმოდგენაში.
4. გადავიტანოთ ეს კლასი პაკეტის კომპონენტ Student სპეციფიკაციაზე ბრაუზერის კომპონენტების წარმოდგენაში. შედეგად კლასი Student შეუსაბამდება სპეციფიკაციას და პაკეტ კომპონენტ Student-ის ტანს.

კოდის გენერაცია

ყოველი ენისათვის Rose-ში გათვალისწინებულია კოდის გენერაციის განსაზღვრული თვისებები. კოდის გენერაციის წინ რეკომენდირებულია გავაანალიზოთ ეს თვისებები და შევიტანოთ აუცილებელი ცვლილებები.

კოდის გენერაციის თვისებების ანალიზისათვის ავირჩევთ Tools > Options, ხოლო შემდეგ შესაბამისი ენის ჩანართი. სიის ფანჯარაში შეიძლება ავირჩიოთ კლასი, ატრიბუტი, ოპერაცია და მოდელის სხვა ელემენტები. ყოველი ენისათვის ამ სიაში მითითებულია მოდელის თავისი საკუთარი ელემენტები. სხვადასხვა მნიშვნელობების არჩევისას ეკრანზე გამოჩნდება თვისებათა სხვადასხვა ნაკრები. ნებისმიერი ცვლილებები, რომლებიც შეიტანება თვისებათა ნაკრებში ფანჯარაში Tools > Options, იმოქმედებენ მოდელის ყველა ელემენტზე, რომლებისთვისაც გამოიყენება მოცემული ნაკრები.

კოდის გენერაციისას Rose ირჩევს ინფორმაციას მოდელის ლოგიკური და კომპონენტური წარმოდგენებიდან და ახდენს დიდი მოცულობის „ჩონჩხი“ კოდის გენერირებას:

- კლასები. გენერირდება მოდელის ყველა კლასები.

- ატრიბუტები. კოდი მოიცავს ყოველი კლასის ატრიბუტებს, მათ შორის ხედვას, მონაცემის ტიპს და მნიშვნელობას დუმილით.
- ოპერაციის სიგნატურები. კოდი მოიცავს ოპერაციის განმარტებას მთელი პარამეტრებით, მოცემული პარამეტრების ტიპებით და ოპერაციის დასაბრუნებელი მნიშვნელობის ტიპით.
- კავშირები. მოდელის ზოგიერთი კავშირები იწვევენ ატრიბუტების შექმნას კოდის გენერაციისას.
- კომპონენტები. ყოველი კომპონენტი რეალიზდება შესაბამისი ფაილის სახით საწყისი კოდით.

C++ კოდის გენერაციისათვის:

1. გახსენით სისტემის კომპონენტების დიაგრამა.
2. ამოირჩიეთ ყველა ობიექტები კომპონენტების დიაგრამაზე.
3. ამოვირჩიოთ Tools > ANSI C++ > Generate kode მენიუში.
4. ავირჩიოთ კატალოგი კოდის გენერაციისათვის.
5. დავაწკაპუნოთ ღილაკზე OK და შეასრულეთ გენერაცია კოდის გენერაციის ფანჯარაში.
6. ვნახოთ გენერაციის შედეგები (მენიუ Tools > C++ Browse Header და Tools > C++ > Browse Body).

გენერაციის შედეგები უნდა გამოიყურებოდეს შემდეგნაირად:

Φαίλ Student.h:

```
#ifndef STUDENT_H_HEADER_INCLUDED_BE29599B
#define STUDENT_H_HEADER_INCLUDED_BE29599B

//##ModelId=35A6336C03DE
class Student
{
public:
//##ModelId=360EBEFA015E
double getTuition();

//##ModelId=374AFB93006B
addSchedule(Schedule theSchedule);

//##ModelId=374AFBDA0117
Schedule getSchedule(Semester forSemester);

//##ModelId=374B00540183
deleteSchedule(Semester forSemester);

//##ModelId=374B02690049
boolean hasPrerequisites(CourseOffering forCourseOffering);

//##ModelId=37812F3903C4
int getNextAvailID();

//##ModelId=379779F60364
int getStudentID();

//##ModelId=37BE859E00CA
string getName();
```

```
//##ModelId=37BF215800D8
string getAddress();

protected:
//##ModelId=37812764010E
boolean passed(CourseOffering theCourseOffering);

private:
//##ModelId=35E9A8EA00BE
string name;

//##ModelId=374D92DE019A
string address;

//##ModelId=37812F2301D8
int nextAvailID;

//##ModelId=378130B900EB
int studentID;

//##ModelId=378BE6A5015D
Date dateofBirth;

};

#endif /* STUDENT_H_HEADER_INCLUDED_BE29599B */
```

Файл Student.cpp:

```
#include "Student.h"

//##ModelId=360EBEFA015E
double Student::getTuition()
{
}

//##ModelId=374AFB93006B
Student::addSchedule(Schedule theSchedule)
{
}
```

```
//##ModelId=374AFBDA0117
Schedule Student::getSchedule(Semester forSemester)
{
}

//##ModelId=374B00540183
Student::deleteSchedule(Semester forSemester)
{
}

//##ModelId=374B02690049
boolean Student::hasPrerequisites(CourseOffering forCourseOffering)
{
}

//##ModelId=37812F3903C4
int Student::getNextAvailID()
{
}

//##ModelId=379779F60364
int Student::getStudentID()
{
}

//##ModelId=37BE859E00CA
string Student::getName()
{
}

//##ModelId=37BF215800D8
string Student::getAddress()
{
}

//##ModelId=37812764010E
boolean Student::passed(CourseOffering theCourseOffering)
{
}
```

3.საკონტროლო კითხვები

- როგორ ხდება მოდელის გასინჯვა
- როგორ ხდება კომპონენტების შექმნა
- როგორ ხდება კომპონენტების შესაბამისობა კომპონენტებთან
- როგორ ხდება დანართის კოდის გენერაცია

ლაბორატორიული სამუშაო №16
სისტემის განაწილებული კონფიგურაციის მოდელირება

1.სამუშაოს მიზანი

სამუშაოს მიზანია შევისწავლოთ:

- სისტემის განაწილებული ქსელური კონფიგურაციის მოდელირება
- განლაგების დიაგრამა
- პროცესების განაწილება კვანძების მიხედვით
- განლაგების დიაგრამის შექმნა

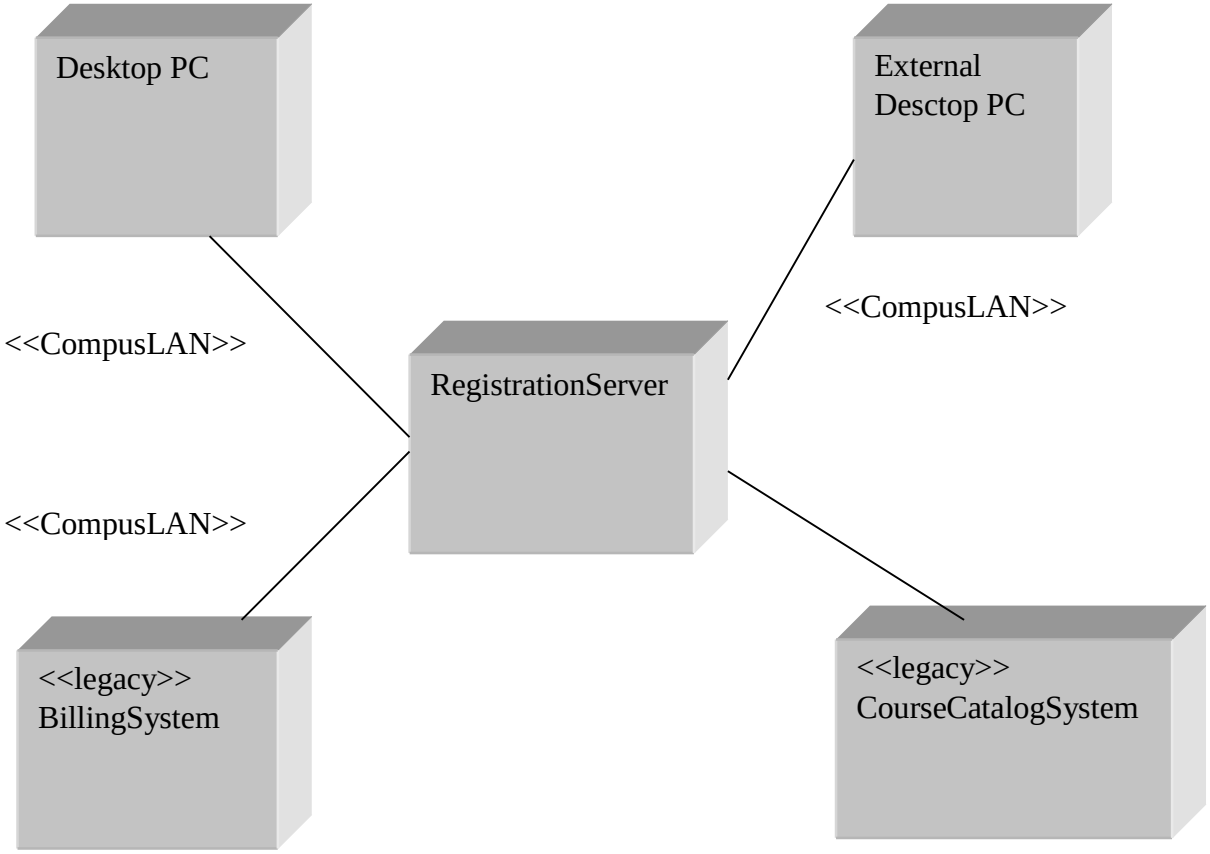
2.მეთოდური მითითებები ლაბორატორიული სამუშაოს ჩასატარებლად

თუ შესაქმნელი სისტემა წარმოადგენს განაწილებულს, საჭიროა მისი კონფიგურაციის დაპროექტება გამოთვლით გარემოში, ე.ი. აღვწეროთ გამოთვლითი რესურსები, მათ შორის კომუნიკაცია და რესურსების გამოყენება სხვადასხვა სისტემური პროცესების მიერ. სისტემის განაწილებული ქსელური კონფიგურაცია მოდელირდება განლაგების დიაგრამით. მისი ძირითადი ელემენტებია:

- კვანძი (node) – გამოთვლითი რესურსი (პროცესორი ან სხვა მოწყობილობა (დისკური მეხსიერება, სხვადასხვა მოწყობილობების

კონტროლერები და ა.შ.). კვანძისათვის შესაძლებელია მიუთითოთ მათზე მიმდინარე პროცესები;

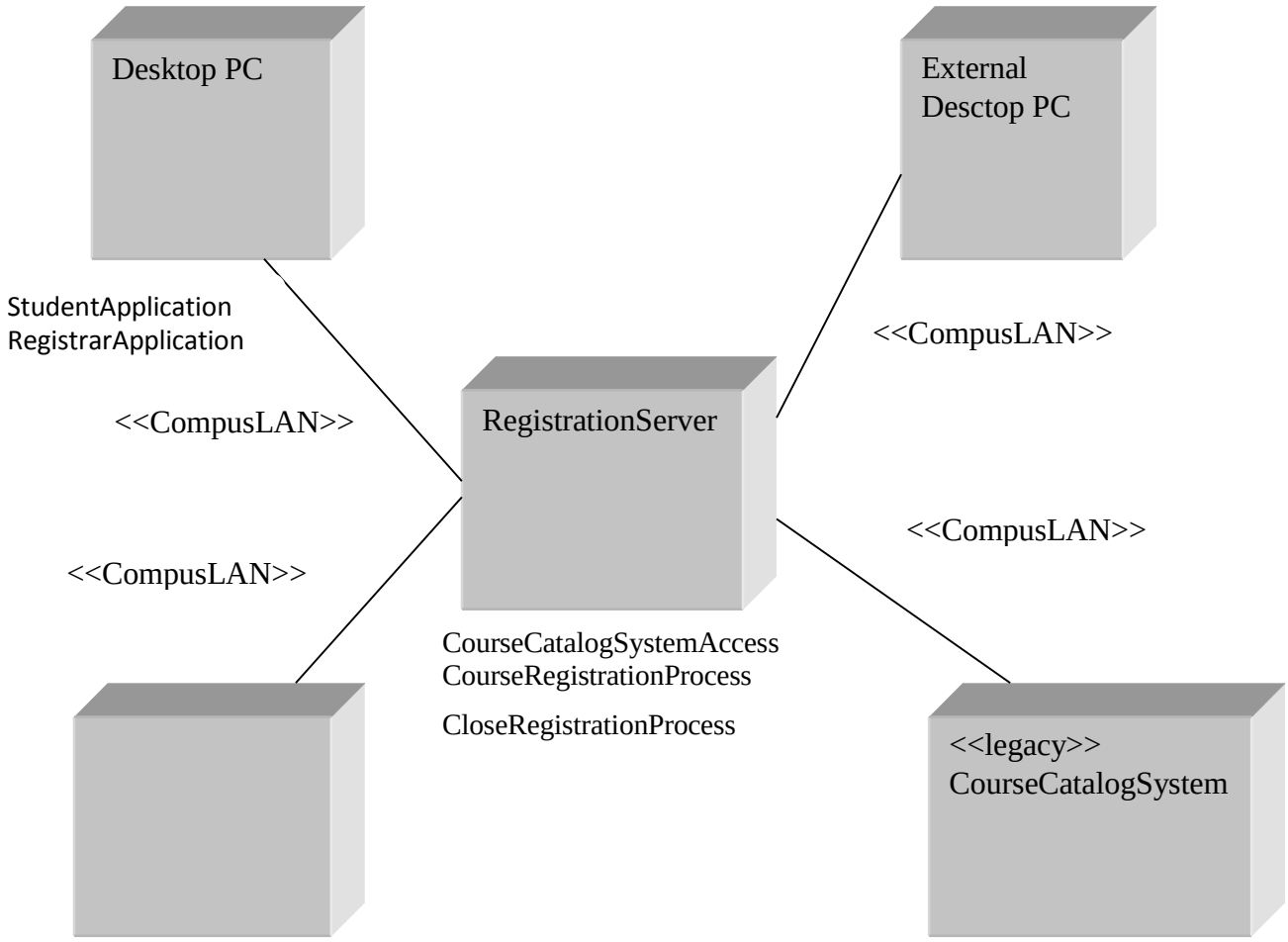
- შეერთება (connection) – კვანძების ურთიერთქმედების არხი (ქსელი). ასეთი დიაგრამის მაგალითი რეგისტრაციის სისტემისათვის მოყვანილია ნახ.5.9-ზე.



ნახ.16.1. რეგისტრაციის სისტემის ქსელური კონფიგურაცია

პროცესების განაწილება კვანძების მიხედვით ხდება შემდეგი ფაქტორების გათვალისწინებით:

- გამოყენებული განაწილების ნიმუშები (სამრგოლიანი კლიენტ – სერვერული კონფიგურაცია, „მსხვილი“ და „თხელი“ კლიენტები, და ა. შ);
- გამოძახების დრო;
- კვანძის სიმძლავრე;
- მოწყობილობის და კომუნიკაციის საიმედობა.



ნახ.16.2. რეგისტრაციის სისტემის ქსელური კონფიგურაცია პროცესების კვანძების მიხედვით განაწილებით

რეგისტრაციის სისტემის განლაგების დიაგრამის შექმნა

იმისათვის, რომ გავხსნათ განლაგების დიაგრამა, უნდა ორჯერ დავაწკაპუნოთ წარმოდგენაზე Deployment View (განლაგების წარმოდგენა) ბრაუზერში.

იმისათვის, რომ მოვათავსოთ დიაგრამაზე პროცესორი:

1. დავაწკაპუნოთ ინსტრუმენტების პანელზე ღილაკზე დიაგრამა Processor.
2. დავაწკაპუნოთ განლაგების დიაგრამაზე იმ ადგილზე, სადაც გინდათ მისი მოთავსება.
3. შევიტანოთ პროცესორის დასახელება.

პროცესორის სპეციფიკაციებში შეიძლება შევიტანოთ ინფორმაცია მისი სტერეოტიპის, მახასიათებლებისა და გეგმის შესახებ. სტერეოტიპები

გამოიყენებიან პროცესორების კლასიფიკაციისათვის (მაგ., კომპიუტერები Unix ან PK მართვით).

პროცესორის მახასიათებლები შესაძლებელია შეიცავდნენ მის სიჩქარეს და მეხსიერების მოცულობას.

პროცესორის დაგეგმვის ველი (scheduling) შეიცავს იმის აღწერას, თუ როგორ ხორციელდება მისი პროცესების დაგეგმვა:

- პრიორიტეტით (Preemptive).
- პრიორიტეტის გარეშე (Non Preemptive).
- ციკლური (Cyclic).
- შემსრულებელი (Executive).
- ხელით (Manual).

იმისათვის, რომ მიუთითოდ პროცესორს სტერეოტიპი:

1. გახსენით პროცესორის სპეციფიკაციის ფანჯარა.
2. გადადით ჩანართზე „General“.
3. შევიტანოთ სტერეოტიპი ველში Stereotipe.

იმისათვის, რომ მიუთითოდ მახასიათებლები და პროცესორის დაგეგმვა:

1. გახსენით სპეციფიკაციის ფანჯარა პროცესორის.
2. გადადით ჩანართზე „General“.
3. შევიტანოთ მახასიათებლები ველში.
4. მიუთითოდ დაგეგმვის ერთ ერთი ტიპი.

იმისათვის, რომ მიუთითოდ დაგეგმვა დიაგრამაზე:

1. დავაწკაპუნოთ მარჯვენა ღილაკით პროცესორზე.
2. გახსნილ მენიუში ავირჩიოთ პუნქტი Show Scheduling.

იმისათვის, რომ მიუთითოდ კავშირი დიაგრამაზე:

1. დავაწკაპუნოთ ღილაკზე Connection ინსტრუმენტების პანელზე.
2. დავაწკაპუნოთ დიაგრამის კუთხეზე.
3. გავატაროთ კავშირის ხაზი სხვა კვანძამდე.

იმისათვის, რომ მიუთითოდ სტერეოტიპი კავშირს:

1. გავხსნათ კავშირის სპეციფიკაციის ფანჯარა.
2. გადადით ჩანართზე „General“.
3. შევიტანოთ სტერეოტიპი ველში Stereotype.

იმისათვის, რომ დავამატოთ პროცესი:

1. დავაწკაპუნოთ მარჯვენა ღილაკით პროცესორზე ბრაუზერში.
2. გახსნილ მენიუში ავირჩიოთ პუნქტი New > Process.
3. შევიტანოთ ახალი პროცესის დასახელება.

იმისათვის, რომ მიუთითოდ პროცესები დიაგრამაზე:

3. დავაწკაპუნოთ მარჯვენა ღილაკით პროცესორზე ბრაუზერში.
4. გახსნილ მენიუში ავირჩიოთ პუნქტი Show Process.

3.საკონტროლო კითხვები

- როგორ ხდება სისტემის განაწილებული ქსელური კონფიგურაციის მოდელირება
- რისგან შესდგება განლაგების დიაგრამა
- როგორ ხდება პროცესების განაწილება კვანძების მიხედვით
- როგორ ხდება განლაგების დიაგრამის შექმნა

4.დავალება

- მოვახდინოთ სისტემის განაწილებული ქსელური კონფიგურაციის მოდელირება;
- ავაგოთ განლაგების დიაგრამა;
- მოვახდინოთ პროცესების განაწილება კვანძების მიხედვით.

დანართი 1

1. ამოცანის დასმა - უმაღლეს სასწავლებლებში სტუდენტების დამატებით ფასიან კურსებზე რეგისტრაციის სისტემის დამუშავება

ამჟამად სტუდენტების რეგისტრაციის პროცესი, რომლებსაც სურთ მოისმინონ ესა თუ ის დამატებითი კურსები, და კურსების განაწილება პროფესორებს შორის რთული და ხანგრძლივია.

მას შემდეგ რაც პროფესორები მიიღებენ გადაწყვეტილებას, თუ რომელი კურსების წაკითხვას აპირებენ მომდევნო სემესტრში, დეკანატის წარმომადგენელს შეყავს მიღებული ინფორმაცია მონაცემთა ბაზაში, რომელიც შეიცავს მთელ ინფორმაციას კურსების შესახებ(კურსების კატალოგი) და ამობეჭდავს ანგარიშს, რომელიც შეიცავს ცნობებს პედაგოგიური დატვირთვების განაწილების შესახებ. შესაბამისი კურსების კატალოგი ასევე გამოქვეყნდება და გადაეცემა სტუდენტებს.

ყოველი სტუდენტი ავსებს სპეციალურ ფორმას, აღნიშნავს ამორჩეულ კურსებს და გადასცემს ფორმას დეკანატს. სტუდენტს შეუძლია სემესტრის განმავლობაში დაესწროს ოთხი კურსის მეცადინეობებს. მონაცემები ფორმიდან, აღებული სტუდენტებისაგან, შეიტანება სისტემაში. როგორც კი მთელი ინფორმაცია იქნება შეტანილი, სრულდება სასწავლო გეგმის ფორმირების პროცედურა. უმეტეს შემთხვევაში ამორჩევის პირველი ვარიანტი, წარმოდგენილი სტუდენტის მიერ, ხდება საბოლაო. მაგრამ თუ წარმოიშვება წინააღმდეგობა ან შეუსაბამობა, სტუდენტს იბარებენ დეკანატში, სადაც მისი მოთხოვნები და სურვილები ზუსტდება და განიხილება ხელახლა. შეთანხმების დასრულების შემდეგ სტუდენტებს ეგზავნებათ მეცადინეობების ცხრილის “მყარი” კოპიები. მთელ პროცესზე ჩვეულებრივ მიდის ერთი-ორი კვირა.

მას შემდეგ რაც სტუდენტთა რეგისტრაციის პროცესი დამთავრებულია, დეკანატის წარმომადგენელი აგზავნის ინფორმაციას

საანგარიშო სისტემაში, იმისათვის, რომ სტუდენტს შეეძლოს შეიტანოს გადასახადი სემესტრზე. რეგისტრაციის დამთავრების შემდეგ პროფესორებს გადაეცემათ სტუდენტების სიები ყოველი კურსის მიხედვით.

კურსებზე რეგისტრაციის პროცესის დაჩქარებისა და ეფექტურობის გაზრდისათვის დაისვა ამოცანა დამუშავდეს სტუდენტთა რეგისტრაციის კლიენტ-სერვერული სისტემა. სისტემის საშუალებით სტუდენტები უნდა დარეგისტრირდნენ კურსებზე და ნახონ მოსწრების ტაბელი პერსონალური კომპიუტერიდან, დაკავშირებული უნივერსიტეტის ლოკალურ ქსელთან. პროფესორებს უნდა შეეძლოთ ონლაინ სისტემასთან მიმართვა, რათა მიუთითონ კურსები, რომლებსაც ისინი წაიკითხავენ, და შეიტანენ შეფასებებს შესაბამის კურსზე. ყოველი სემესტრის დასაწყისში სტუდენტებს შეუძლიათ მოითხოვონ კურსების კატალოგი, რომელიც შეიცავს კურსების ცხრილს და წარედგინება სწავლებისათვის მოცემულ სემესტრში. ინფორმაცია ყოველი კურსის შესახებ უნდა შეიცავდეს პროფესორის სახელს, კათედრის დასახელებას და მოთხოვნებს მომზადების წინასწარი დონის შესახებ.

სისტემა საშუალებას უნდა აძლევდეს სტუდენტებს აირჩიონ ოთხი კურსი მომდევნო სემესტრისათვის. დამატებით ყოველ სტუდენტს უნდა შეეძლოს მიუთითოს ორი ალტერნატიული კურსი იმ შემთხვევისათვის, თუ რომელიმე არჩეული კურსი უკვე აღმოჩნდება შევსებული ან გამორიცხული. ყოველ კურსზე შესაძლებელია ჩაეწეროს არა უმეტეს 10 და არა ნაკლები 3 სტუდენტისა. ყოველ სემესტრში არის დროის მონაკვეთი, როდესაც სტუდენტებს შეუძლიათ შეცვალონ თავიანთი გეგმები. ამ დროს სტუდენტებს უნდა ქონდეთ სისტემასთან მიმართვის საშუალება, რათა დაამატონ ან ამოიღონ არჩეული კურსები. მას შემდეგ რაც რეგისტრაციის პროცესი რომელიმე სტუდენტის დამთავრდება, რეგისტრაციის სისტემა აგზავნის ინფორმაციას საანგარიშსწორებო სისტემაში, რათა სტუდენტს შეეძლოს შეიტანოს ანგარიში სემესტრზე. თუ კურსი აღმოჩნდება შევსებული რეგისტრაციის პროცესში, სტუდენტი უნდა ინფორმირებული

იყოს ამის შესახებ მანამდე, ვიდრე მისი პირადი სასწავლო გეგმა საბოლაოდ იქნება ფორმირებული.

სემესტრის ბოლოს სტუდენტებს უნდა ქონდეთ სისტემასთან მიმართვის საშუალება თავიანთი მოსწრების ტაბელების ნახვისათვის. რამდენადაც ეს ინფორმაცია კონფიდენციალურია, სისტემა უნდა უზრუნველყოფდეს მის დაცვას არასანქცირებული მიმართვისაგან.

პროფესორებს უნდა ქონდეთ ონლაინ სისტემასთან მიმართვის საშუალება, რათა მიუთითონ კურსები, რომლებსაც ისინი წაიკითხავენ, და გადასინჯონ სტუდენტთა ცხრილი, რომლებიც ჩაეწერნენ მათ კურსებზე. ამას გარდა, პროფესორებს უნდა შეეძლოთ ნახონ შეფასებები კურსზე.

დანართი 2

აღწერების დამატებისათვის:

4. გამოვყოთ ბრაუზერში გამოყენებითი შემთხვევა “კურსებზე დარეგისტრირება”.
5. დოკუმენტაციის ფანჯარაში შევიტანოთ შემდეგი აღწერა “ეს გამოყენებითი შემთხვევა აძლევს სტუდენტს შესაძლებლობას დარეგისტრირდეს კურსებზე მიმდინარე სემესტრში”.
6. შევქმნათ MS Word – ის მეშვეობით ტექსტური ფაილები ქვემოთ მოყვანილი გამოყენებითი შემთხვევების აღწერებისათვის.

გამოყენებითი შემთხვევების განსაზღვრისას უნდა დაინიშნოს თვითეულისათვის პრიორიტეტი, რომლითაც განისაზღვრება მისი შემდგომი რეალიზაციის თანმიმდევრობა.

იმისათვის, რომ გამოყენებით შემთხვევას დაუნიშნოთ პრიორიტეტი:

4. დავაწკაპუნოთ მარჯვენა ღილაკზე გამოყენებით შემთხვევაზე ბრაუზერში ან დიაგრამაზე.
5. გახსნილ მენიუში ავირჩიოთ პუნქტი Open Specification.
6. შევიტანოთ პრიორიტეტი ველში Rank ჩანართში General.

გამოყენებითი შემთხვევა "სისტემაში შესვლა"

მოკლე აღწერა:

გამოყენებით შემთხვევათა მოცემული ვარიანტი აღწერს მომხმარებლის შესვლას კურსების რეგისტრაციის სისტემაში.

მოვლენათა ძირითადი ნაკადი:

გამოყენებით შემთხვევათა მოცემული ვარიანტი იწყებს შესრულებას, როდესაც მომხმარებელს სურს კურსების რეგისტრაციის სისტემაში შესვლა.

1. სისტემა ჩაეკითხება მომხმარებლის სახელს და პაროლს.
2. მომხმარებელს შეყავს სახელი და პაროლი.
3. სისტემა ადასტურებს სახელს და პაროლს, რომლის შემდეგ იხსნება სისტემასთან მიმართვის შესაძლებლობა.

ალტერნატიული ნაკადები:

არასწორი დასახელება/პაროლი:

თუ ძირითადი ნაკადის შესრულებისას აღმოჩნდება, რომ მომხმარებელმა შეიტანა არასწორად სახელი/პაროლი, სისტემას გამოყავს შეტყობინება შეცდომის შესახებ. მომხმარებელს შეუძლია დაბრუნდეს ძირითადი ნაკადის დასაწყისთან ან უარი თქვას სისტემაში შესვლაზე, ამასთან გამოყენებითი შემთხვევის შესრულება მთავრდება.

გამოყენებითი შემთხვევა "კურსებზე დარეგისტრირება"

მოკლე აღწერა:

გამოყენების მოცემული ვარიანტი საშუალებას აძლევს სტუდენტს დარეგისტრირდეს შემოთავაზებულ კურსებზე მოცემულ სემესტრში. სტუდენტს შეუძლია შეცვალოს თავისი არჩევანი (განაახლოს ან გამორიცხოს კურსები), თუ ცვლილება სრულდება დადგენილ დროს

სემესტრის დასაწყისში. კურსების კატალოგის სისტემა წარადგენს მიმდინარე სემესტრის ყველა შეთავაზებული კურსების ცხრილს.

მოვლენათა ძირითადი ნაკადი:

გამოყენების მოცემული ვარიანტი იწყებს შესრულებას, როდესაც სტუდენტს სურს დარეგისტრირდეს კონკრეტულ კურსებზე ან შეცვალოს თავისი კურსების გრაფიკი.

1. სისტემა გამოკითხავს საჭირო მოქმედებას (შექმნას გრაფიკი, განაახლოს გრაფიკი, მოსპოს გრაფიკი).
2. როდესაც სტუდენტი უთითებს მოქმედებას, სრულდება ერთ ერთი დაქვემდებარებული ნაკადი (შეიქმნას, მოისპოს, ან მიიღოს გრაფიკი).

შეიქმნას გრაფიკი:

1. სისტემა ასრულებს შესათავაზებელი კურსების ძებნას კურსების კატალოგში და გამოყავს მათი ცხრილი.
2. სისტემას გამოყავს ცარიელი გრაფიკი შევსებისათვის.
3. სტუდენტი არჩევს ცხრილიდან ოთხ ძირითად და ორ ალტერნატიულ კურსს გრაფიკში ჩასართველად.

ყოველი არჩეული კურსისათვის სრულდება დაქვემდებარებული ნაკადი “დაემატოს კურსი გრაფიკში”.

სისტემა ინახავს სტუდენტის გრაფიკს.

გრაფიკის განახლება:

1. სისტემას გამოყავს სტუდენტის მიმდინარე გრაფიკი.
2. სისტემა ასრულებს კურსების კატალოგში დასაშვები შესათავაზებელი კურსების ძებნას და გამოყავს მათი ცხრილი.
3. სტუდენტს შეუძლია განაახლოს თავისი კურსების არჩევანი, მოსპოს ან დაამატოს შეთავაზებული კურსები.
4. ყოველი არჩეული კურსისათვის სრულდება დაქვემდებარებული ნაკადი “დაემატოს კურსი გრაფიკში”.
5. სისტემა ინახავს სტუდენტის გრაფიკს.

გრაფიკის მოსპობა:

1. სისტემას გამოყავს სტუდენტის მიმდინარე გრაფიკი.
2. სისტემა ეკითხება სტუდენტს დაადასტუროს გრაფიკის მოსპობა.
3. სტუდენტი ადასტურებს მოსპობას.
4. სისტემა სპობს გრაფიკს. თუ გრაფიკი შეიცავს შეთავაზებულ კურსებს, რომლებზედაც ჩაეწერა სტუდენტი, ის უნდა პოისპოს ამ კურსების ცხრილიდან.

დაუმატოთ კურსი გრაფიკს:

ყოველი არჩეული კურსისათვის სისტემა ამოწმებს სტუდენტის მიერ წინასწარი მოთხოვნების შესრულების ფაქტს (გარკვეული კურსების გავლა) და შეთავაზებულ კურსებზე მიღების არსებობას. შემდეგ სისტემა ამატებს სტუდენტს არჩეული კურსის ცხრილში. კურსი აღინიშნება გრაფიკში როგორც “დარეგისტრირებული”.

ალტერნატიული ნაკადები:

შევინახოთ გრაფიკი:

სტუდენტს შეუძლია შეინახოს გრაფიკი ნებისმიერ მომენტში, ისე რომ არ დააფიქსიროს ამორჩეული კურსები. ამ შემთხვევაში გრაფიკი ინახება სისტემაში, მაგრამ სისტემა არ ამატებს სტუდენტს ამორჩეული კურსების ცხრილში. კურსები აღინიშნებიან გრაფიკში “ამორჩეულები”.

არ შესრულებულა წინასწარი მოთხოვნები ან კურსი შევსებულია:

თუ დაქვემდებარებული ნაკადის “დაუმატოთ კურსი გრაფიკში” შესრულებისას სისტემა აღმოაჩენს, რომ სტუდენტს არ შეუსრულებია აუცილებელი წინასწარი მოთხოვნები ან ამორჩეული მის მიერ კურსი შევსებულია, მაშინ გამოიცემა შეტყობინება შეცდომის შესახებ. სტუდენტს შეუძლია ან ამოირჩიოს სხვა კურსი და განაგრძოს გამოყენებითი შემთხვევის შესრულება, ან უარყოს ოპერაცია, რომლის შემდეგაც ძირითადი ნაკადი დაიწყება ახლიდან.

გრაფიკი არ იქნა ნანახი:

თუ დაქვემდებარებული ნაკადის “გრაფიკის განახლება” ან “მოვსპოთ გრაფიკი” შესრულებისას სისტემას არ შეუძლია იპოვოს სტუდენტის

გრაფიკი, მაშინ გამოიცემა შეტყობინება შეცდომის შესახებ. მას შემდეგ რაც სტუდენტი დაადასტურებს ამ შეტყობინებას, ძირითადი ნაკადი დაიწყება თავიდან.

კურსების კატალოგის სისტემა მიუწვდომელია:

თუ აღმოჩნდება, რომ კურსების კატალოგის სისტემასთან კავშირის დამყარება შეუძლებელია, მაშინ გამოიცემა შეტყობინება შეცდომის შესახებ. მას შემდეგ რაც სტუდენტი დაადასტურებს ამ შეტყობინებას, გამოყენებითი შემთხვევა დამთავრდება.

კურსებზე რეგისტრაცია დამთავრდა:

თუ გამოყენებითი შემთხვევის შესრულების დასაწყისში აღმოჩნდება, რომ რეგისტრაცია მიმდინარე სემესტრზე დამთავრდა, გამოიცემა შეტყობინება და გამოყენებითი შემთხვევა დამთავრდება.

წინაპირობები:

მოცემული გამოყენებითი შემთხვევის დაწყების წინ სტუდენტი უნდა შევიდეს სისტემაში.

გამოყენებითი შემთხვევა "დავხუროთ რეგისტრაცია"

მოკლე აღწერა:

მოცემული გამოყენებითი შემთხვევა საშუალებას აძლევს რეგისტრატორს დახუროს რეგისტრაციის პროცესი. შეთავაზებული კურსები, რომლებზედაც არ ჩაწერილანსტუდენტების საკმარისი რაოდენობა (სამზე ნაკლები), გამოირიცხება. ანგარიშსწორების სისტემაში გადაიცემა ინფორმაცია ყოველ სტუდენტზე თითოეული შეთავაზებული კურსის მიხედვით, იმისათვის, რომ სტუდენტებს შეეძლოს შეიტანონ გადასახადი კურსებზე.

მოვლენათა ძირითადი ნაკადი:

მოცემული გამოყენებითი შემთხვევა იწყებს შესრულებას, როცა რეგისტრატორი მოითხოვს რეგისტრაციის შეწყვეტას.

სისტემა დაადასტურებს რეგისტრაციის პროცესის დასრულებას.

ყოველი შეთავაზებული კურსისათვის სისტემა ამოწმებს, მიყავს იგი რომელიმე პროფესორს თუ არა და ჩაეწერა თუ არა მასზე არა ნაკლებ სამი სტუდენტისა. თუ ეს პირობები სრულდება, სისტემა საბოლაოდ აფიქსირებს კურსს ყოველ გრაფიკში, რომელიც მოიცავს მოცემულ კურსს.

სისტემა ხურავს ყველა კურსებს, ანგარიშობს გადასახადს სწავლაზე ყოველი სტუდენტისათვის მიმდინარე სემესტრში და აგზავნის ინფორმაციას სააგარიშსწორებო სისტემაში. მოცემული სისტემა უგზავნის სტუდენტებს ანგარიშს მათ საბოლაო გრაფიკის კოპიოსთან ერთად.

ალტერნატიული ნაკადები:

რეგისტრაცია არ დასრულებულა:

თუ რეგისტრაციის პროცესის დასრულების შემოწმებისას აღმოჩნდება, რომ რეგისტრაცია კიდევ სრულდება, გამოიცემა შეტყობინება და გამოყენებითი შემთხვევა სრულდება.

კურსზე ჩაეწერა სამ სტუდენტზე ნაკლები:

თუ ძირითადი ნაკადის შესრულების დროს აღმოჩნდება, რომ რომელიმე კურსზე ჩაეწერა სამ სტუდენტზე ნაკლები, მაშინ ეს კურსი უქმდება და სრულდება დაქვემდებარებული ნაკადი “კურსის გაუქმება”.

კურსი არავის არ მიყავს:

თუ ძირითადი ნაკადის შესრულების დროს აღმოჩნდება, რომ რომელიღაც კურსი არც ერთ პროფესორს არ მიყავს, მაშინ ეს კურსი უქმდება და სრულდება დაქვემდებარებული ნაკადი “კურსის გაუქმება”.

კურსის გაუქმება:

სისტემა აუქმებს შეთავაზებულ კურსს. ყოველი სტუდენტისათვის, რომელიც ჩაეწერა გაუქმებულ კურსზე, სისტემა ახდენს მისი გრაფიკის მოდიფიცირებას. პირველი მოპოვებული ალტერნატიული კურსი ჩაისმება გაუქმებული კურსის მაგივრათ. თუ ალტერნატიული კურსები არ არის, ჩასმა არ ხდება და მართვა გადაეცემა მოვლენათა ძირითად ნაკადში შემდეგი შეთავაზებული კურსის დამუშავებისათვის.

მიმდინარე სემესტრის ყველა გრაფიკების დამუშავების შემდეგ სისტემა ელექტრონული ფოსტით ატყობინებს სტუდენტებს მათ გრაფიკებში ცვლილებების შესახებ.

საანგარიშსწორებო სისტემა მიუწვდომელია:

თუ საანგარიშსწორებო სისტემასთან კავშირის დამყარება შეუძლებელია, სისტემა ცდილობს ხელახლა დაუკავშირდეს მას გარკვეული დადგენილი დროის შემდეგ. დაკავშირების ცდები განმეორდებიან მანამდე, სანამ კავშირი არ დამყარდება.

წინაპირობები:

მოცემული გამოყენებითი შემთხვევის შესრულების დაწყების წინ რეგისტრატორი უნდა შევიდეს სისტემაში.

დანართი 3

ტერმინი	მნიშვნელობა
კურსი	სასწავლო კურსი, რომელიც შეთავაზებულია უნივერსიტეტის მიერ
შეთავაზებული კურსი	კონკრეტულ სემესტრში მოცემული კურსის წაკითხვის შეთავაზება. მოიცავს კვირის კონკრეტულ დღეებსა და დროს.
კურსების კატალოგი	ყველა კურსების მთლიანი კატალოგი, რომელსაც სთავაზობს უნივერსიტეტი
პროფესორი	უნივერსიტეტის პედაგოგი

სტუდენტი	პიროვნება, რომელიც გადის სწავლებას უნივერსიტეტში
კურსის სია	ყველა სტუდენტები, რომლებიც ჩაეწერნენ შეთავაზებულ კურსზე
სასწავლო გრაფიკი	კურსები, რომლებიც აირჩია სტუდენტმა მიმდინარე სემესტრში

ფუნქციონალური შესაძლებლობები

- სისტემა უნდა უზრუნველყოფდეს მუშაობის მრავალმომხმარებლის რეჟიმს.
- გამოყენების მოხერხებულობა.
- მომხმარებლის ინტერფეისი უნდა იყოს Windows-შეთავსებადი.
- სისტემის მომხმარებლის ინტერფეისი უნდა იყოს მარტივი და არ მოითხოვდეს მომხმარებლისაგან, რომელსაც აქვს კომპიუტერული განათლება, დამატებით შესწავლას.
- სისტემის ყოველი ფუნქცია უნდა უზრუნველყოფილი იყოს ჩართული ონლაინ დახმარებით, რომელიც უნდა შეიცავდეს ინსტრუქციას სისტემასთან მუშაობისათვის.

საიმედობა

- სისტემა უნდა იყოს სამუშაო მდგომარეობაში 24 სთ კვირაში 7 დღე, უქმად დგომის დრო არ უნდა აღემატებოდეს 10. საიმედო მუშაობის საშუალო დრო არ უნდა აღემატებოდეს 300 სთ-ს.

წარმადობა

- სისტემა უნდა აკავებდეს დაახლოებით 2000 მომხმარებელს, რომლებიც ერთდროულად იმუშავებენ მონაცემთა ცენტრალურ ბაზასთან, და

დაახლოებით 500 მომხმარებელს, რომლებიც ერთდროულად იმუშავებენ ლოკალურ სერვერებთან.

- სისტემა უნდა უზრუნველყოფდეს კურსების კატალოგის მონაცემთა ბაზასთან მიმართავას დაყოვნების დროით არა უმეტეს 10 წმ.
- სისტემას უნდა შეეძლოს ყველა ტრანზაქციების 80 –ის დასრულებას 2 წთ-ის განმავლობაში.

უშიშროება

- სისტემა არ უნდა აძლევდეს საშუალებას სტუდენტებს შეცვალონ ნებისმიერი სასწავლო გრაფიკები, საკუთარის გარდა, ასევე პროფესორებმა მოახდინონ კონკრეტული კურსების მოდიფიცირება, რომლებიც სხვა პროფესორების მიერ არის არჩეული.
- მხოლოდ პროფესორებს აქვთ უფლება დაუსვან შეფასებები სტუდენტებს.
- მხოლოდ რეგისტრატორს შეუძლია შეცვალოს ნებისმიერი ინფორმაცია სტუდენტების შესახებ.
- საპროექტო შეზღუდვები.
- სისტემა ინტეგრირებული უნდა იყოს კურსების კატალოგის არსებულ სისტემასთან, რომელიც ფუნქციონირებს რელაციური მონაცემთა ბაზების მართვის სისტემის საფუძველზე.

ლიტერატურა

1. Буч Г., Рамбо Д., Джекобсон А. Язык **UML**. Руководство пользователя.// Серия “Объектно- ориентированные технологии в программировании”. Москва, 2004.
2. А.М.Вендров. Практикум по проектированию програмного обеспечения зкономических информационных систем.Москва. Финансы и статистика.2006.

3. **Леоненков. Самоучитель UML. UML** Teach Yourself. **...** Особенности реализации языка **UML** в CASE-инструментарии Rational Rose 98/2000 · Заключение · Каталог · Индекс раздела. khpi-iip.mipk.kharkiv.edu/library/case/leon/index.html - 3k -
4. Кватрани Т. Rational Rose и UML. Визуальное моделирование; Пер. с англ.М.:ДМК Пресс, 2001.-176 с:нл.
5. სუხიაშვილი თ. სისტემების ობიექტზე ორიენტირებული ანალიზი და დაპროექტება. სახელმძღვანელო. დამტკიცებულია სტუ-ს სარედაქციო-საგამომცემლო საბჭოს მიერ. გამომცემლობა "ტექნიკური უნივერსიტეტი", 2009, 170 გვ.
6. სუხიაშვილი თ. მოდელირების უნიფიცირებული ენა(UML).პრაქტიკული ობიექტ-ორიენტირებული ანალიზი და დაპროექტება. თბილისი, "ტექნიკური უნივერსიტეტი". 2018, 210გვ. ელექტრონული სახელმძღვანელო.