გია სურგულაძე, მარინე ბიტარაშვილი,
ხატია ქრისტესიაშვილი

პროგრამული აპლიკაციების დეველოპმენტის საფუძვლები

(დამხმარე სახელმძღვანელო საკურსო პროექტის შესასრულებლად)

რეკომენდებულია სტუ-ს
სარედაქციო-საგამომცემლო საბჭოს
მიერ 03.04.13, ოქმი №2

თბილისი
2013

უაკ 004.5

განიხილება პროგრამული აპლიკაციების დეველოპმენტის (გამოყენებითი კომპიუტერული სისტემების დაპროგრამების) საფუძვლები. კერძოდ, შემოთავაზებულია აღნიშნულ დისციპლინაში საკურსო პროექტის შესრულების მეთოდოლოგია და პროგრამული ინსტრუმენული საშუალებები. ასევე სამაგიდო (Windows) და ინტერნეტული აპლიკაციების (Web) აგების ეტაპები და პროცედურები, MsVisual Studio.NET Framework 4.0/4.5 გარემოში, C#, MsAccess, ASP.NET, ADO.NET, HTML/XML პროგრამული ენების და პლატფორმების გამოყენებით.

განკუთვნილია ინფორმატიკის საგანმანათლებლო პროგრამის „მართვის ავტომატიზებული სისტემების“ (პროგრამული ინჟინერია) სპეციალობის ბაკალავრიატის მეორე კურსის სტუდენტებისთვის.

რეცენზენტები: პროფ. ე. თურქია,

პროფ. თ. სუხიაშვილი

პროფესორ გია სურგულაძის რედაქციით

ISBN 978-9941-20-251-3

http://www.gtu.ge/publishinghouse/

სარჩევი

I თავი. საკურსო პროექტის მიზანი და ამოცანის დასმა

პროექტის მიზანია Windows და Web აპლიკაციების დეველოპინგის (გამოყენებითი პროგრამული პაკეტების აგების) საფუძვლების შესწავლა. საერთაშორისო სტანდარტების მოთხოვნების გათვალისწინებით უახლესი ინფორმაციული ტექნოლოგიების (IT) პრაქტიკული გამოყენების უნარ-ჩვევების გამომუშავება. თანამედროვე უნიფიცირებული, ობიექტ-ორიენტირებული და პროცესორიენტირებული ვიზუალური მეთოდების და ინსტრუმენტული საშუალებების ათვისება.

სტუდენტი ინდივიდუალურად (ან რამდენიმე სტუდენტი ჯგუფურად) მიიღებს კონკრეტულ დავალებას (ამოცანას) პროექტის შესასრულებლად რომელიმე საპრობლემო სფეროს კომპიუტერული მართვის საინფორმაციო სისტემის დაპროექტების და რეალიზაციის მიზნით.

საპრობლემო სფერო შეიძლება იყოს ნებისმიერი დარგის ობიექტი. მაგალითად, განათლების სფერო: უნივერსიტეტი, ფაკულტეტი, კათედრა, სკოლა და ა.შ.; სამინისტროს ნებისმიერი დეპარტამენტი, სოფლის მეურნეობის სფერო: სასოფლო სამეურნეო პროდუქციის წარმოება, ფერმა და ა.შ.; საბანკო სისტემა: კლიენტთა ანაბრების მართვა, კრედიტების დეპარტამენტი და ა.შ., ბიზნესის და კომერციის სფერო: მარკეტინგის დეპარტამენტი, სასაწყობო მეურნეობა, სუპერმარკეტი და ა.შ. ან სხვა სფეროები (აეროპორტი, ტრანსპორტი, აფთიაქები, ბიბლიოთეკა, სხვადასხვა სახის ცნობარი და ა.შ.).

მაგალითისათვის ჩვენ განვიხილავთ საილუსტრაციო ამოცანას - „ფირმაში ახალი თანამშრომლის მიღების და ხელფასის დარიცხვის ავტომატიზებული სისტემა“.

ამოცანის გადასაწყვეტად საჭიროა:

1) აიგოს ფირმაში ახალი თანამშრომლის მიღების ამოცანის ბიზნესპროცესის BPMN დიაგრამა, რომლის საფუძველზე შესაძლებელი იქნება თანამშრომლებზე ხელფასის დარიცხვის ფუნქციური პროცედურების სტრუქტურელი ანალიზის ჩატარება;

2) განისაზღვროს ფუნქციური პროცედურების (ფუნქციების) და მათი შემსრულებლების (როლების), აგრეთვე ხელფასის დარიცხვის ქმედების დიაგრამები (UseCase, Activity) UML ტექნოლოგიის გამოყენებით (MsVisio პაკეტით Software & Database კატეგორიაში);

3) გამოვლინდეს დოკუმენტთა ფორმები და ინფორმაცია:

- საწყისი (ნორმატიული: თანამდებობები, ხელფასები, დაქვითვების სკალა; - ოპერატიული: ყოველდღიური ან თვიური აღრიცხვის ტაბელები, შვებულების განრიგი, საავადმყოფო ბიულეტენი); - გამომავალი (უწყისი თანამშრომელთა ანგარიშებზე გადასარიცხად);

4) განისაზღვროს მონაცემთა ბაზის სტრუქტურა არსთა დამოკიდებულების მოდელით. აიგოს ER დიაგრამა MsVisio პაკეტით Database კატეგორიაში;

5) აიგოს MsAccess მონაცემთა ბაზის ლოგიკური სტრუქტურა და ცხრილები (Tables). შეივსოს ცხრილები სტრიქონებით;

6) შემუშავდეს მომხმარებელთა ინტერფეისები (სამუშაო ფორმები), MsVisual Studio.NET ინტეგრირებულ გარემოში (Windows Form) და C# ენის საფუძველზე;

7) შემუშავდეს ხელფასის დარიცხვის ამოცანის პროცესების შესრულების მეთოდები;

8) შემუშავდეს მომხმარებელთა ინტერფეისებისთვის ხელფასების MsAccess ბაზასთან კავშირის და მონაცემთა ამორჩევა/ მოდიფიკაციის მეთოდები, შესაბამისი C# კოდებით;

9) SQL-ენაზე შეიქმნას საილუსტრაციო სტანდარტული მოთხოვნები, რომლებიც განთავსდება Button ღილაკების პანელზე;

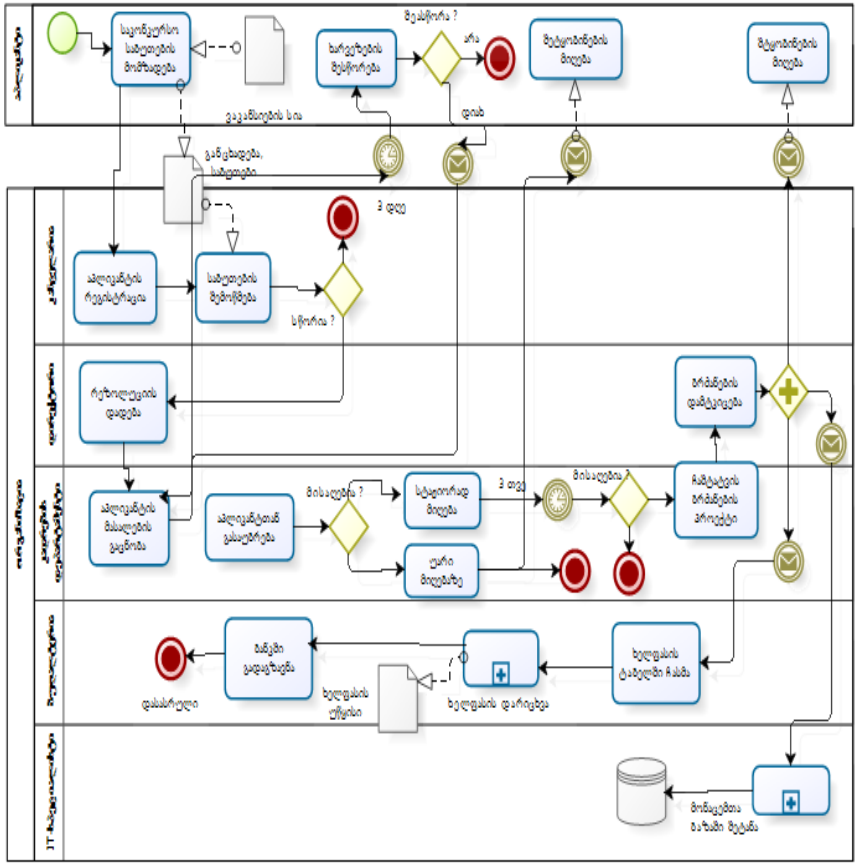
10) ჩატარდეს აგებული სისტემის ტესტირება კონკრეტულ მონაცემებზე;

11) მომზადდეს სისტემის სადემონსტრაციო ვერსია და პროექტის აღწერა მომხმარებელთა ინსტრუქციებით.

II თავი. ბიზნესპროცესების აღწერა და სისტემის დაპროექტება

2.1. პროცესებზე ორიენტირებული მოდელი – BPMN

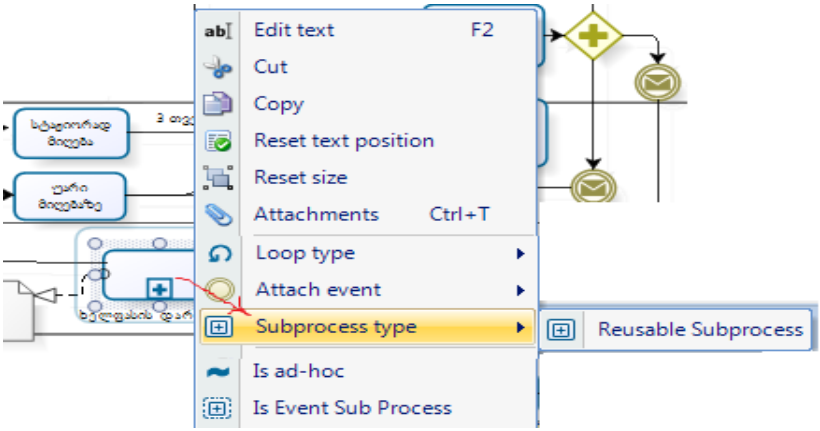
განვიხილოთ ორგანიზაციაში ახალი თანამშრომლის მიღების ამოცანის ბიზნესპროცესის BPMN დიაგრამა (ნახ.2.1–ა).



ნახ.2.1–ა. ახალი თანამშრომლის მიღების BPMN მოდელის ფრაგმენტი

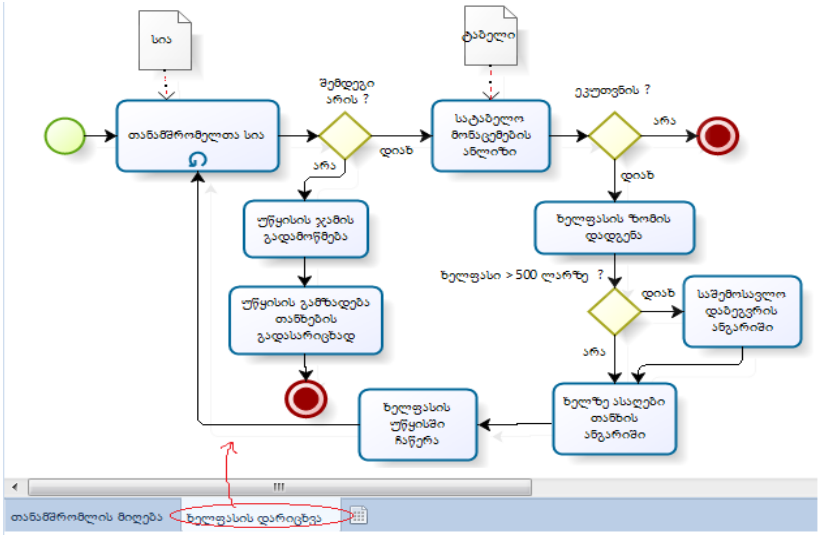
მთლიანი ბიზნესპროცესი დაყოფილია ქმედებებად (Activities), რომლებიც გამოსახულია მრგვალკუთხედებით, გადასასვლელები ქმედებებს შორის კი – ისრებით. დოკუმენტები, რომლებსაც საწყისად იყენებს ეს ქმედებები ან თვითონ ქმნის როგორც საშედეგოს, ნაჩვენებია კუთხეჩამოკეცილი ფურცლის სახით. ეს ფურცლები შეერთებულია წყვეტილი ისრებით იმ ქმედებებთან, რომლებიც მათ ქმნის (ქმედებიდან გამომავალი ისარი) ან გამოიყენებს შესასვლელზე (ქმედებისკენ მიმართული ისარი).

სქემაზე „+”-ით აღნიშნულია ქმედება, რომელიც ქვეპროცესია. მისთვის ცალკე აიგება დიაგრამა, მაგალითად, ბლოკზე „ხელფასის დარიცხვა“ მარჯვენა ღილაკით (ნახ.2.2) ავირჩევთ Subprocess პუნქტს.



ნახ.2.2. ქვეპროცესის შექმნის ინიცირება

2.3 ნახაზზე ნაჩვენებია ახლად აგებული ქვეპროცესის BPMN სქემა. ძირითადი პროცესის და ქვეპროცესების დიაგრამათა ასარჩევად გამოიყენება გადამრთველი სათაურებით (ქვედა მარცხენა კუთხეში).

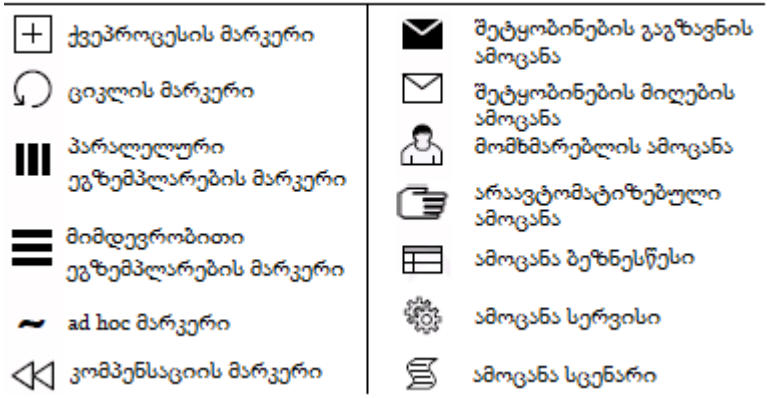


ნახ.2.3. ხელფასის დარიცხვის ქვეპროცესის სქემა

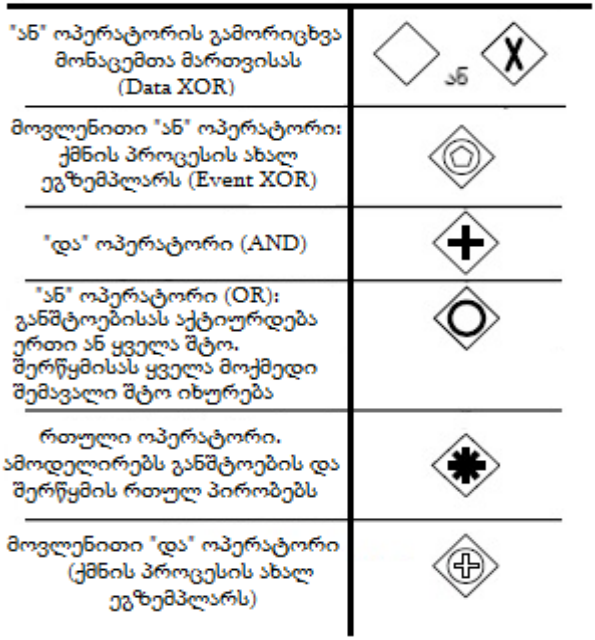
ბიზნესპროცესების მოდელის ასაგებად გამოყენებულია BPMN ნოტაციის ელემენტები, რომლებიც ასახულია ქვემოთ მოცემულ ცხრილებში:

ქმედებები	
ამოცანა	სამუშაო ერთეული. თუ მას აქვს "+"სიმბოლო, იგი ქვეპროცესითაა
ტრანზაქცია	ლოგიკურად დაკავშირებული ქმედებების ერთობლიობაა
მოვლენითი ქვეპროცესი	მოთავსებულია სხვა პროცესის შიგნით. ის სრულდება, როცა ინიცირდება მისი საწყისი მოვლენა. მას შეუძლია მშობელი ქვეპროცესის შეწყვეტა ან მის პარალელურად შესრულება
გამომძახებელი ქმედება	არის შესასვლელი წერტილი გლობალურად განსაზღვრული ქვეპროცესის, რომელიც ხელმეორედ გამოიყენება ამ პროცესში

ქმედებათა მარკერები

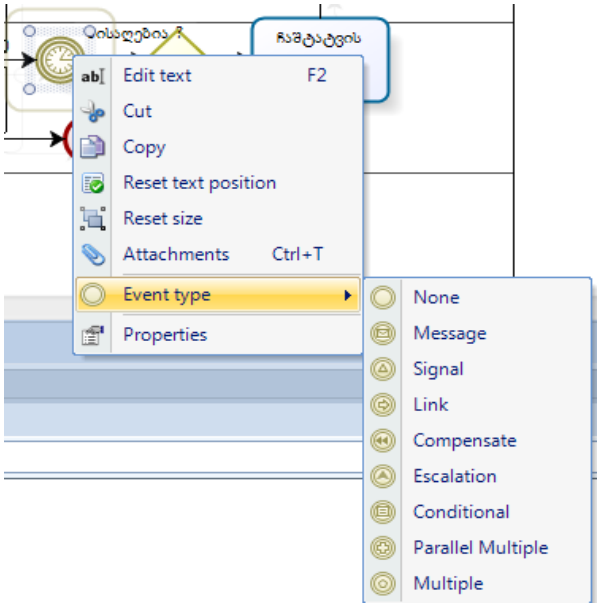


ლოგიკური ოპერატორები

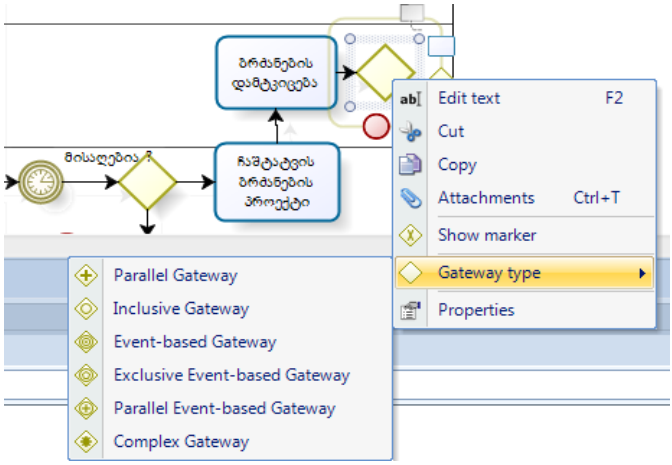


მოვლენები

მოვლენათა ტიპები	დაწყება	შუალედური	დასასრული
ჩვეულებრივი			
შეტყობინება			
წამზომი			
შეცდომა			
შეწყვეტა			
კომპენსაცია			
ბიზნესწესის შესრულება			
ბმული			
სიგნალი			
შედგენილი (სრულდება ერთი)			
პარალელური შედგენილი (სრულდება ყველა)			
ესკალაცია: საკითხის ატანა ორგ–იერარქიის ზედა დონეზე			
შეჩერება			



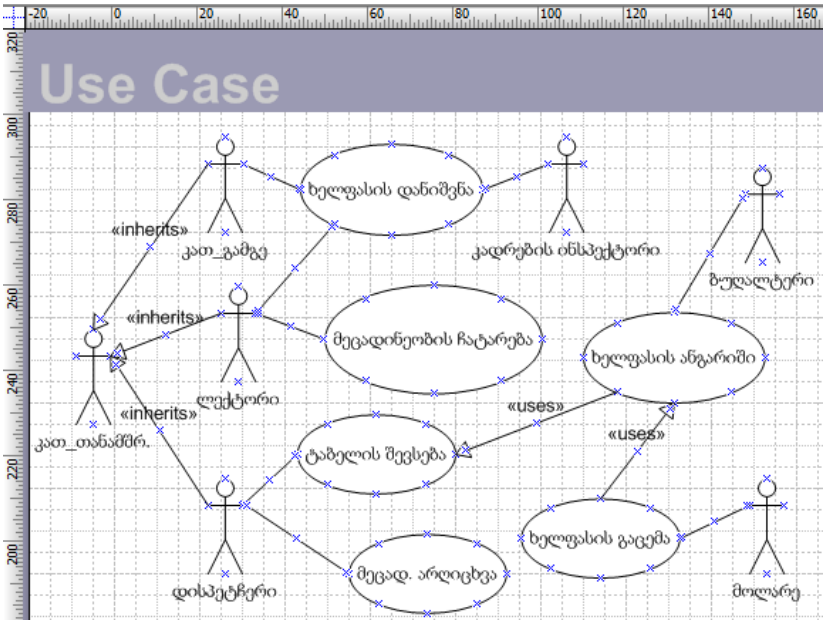
ნახ.2.4. მოვლენის ტიპის შერჩევა (Bizag გარემოში)



ნახ.2.5. ლოგიკური ოპერატორის ტიპის შერჩევა (Bizag გარემოში)

2.2. ობიექტებზე ორიენტირებული მოდელი – UML: როლები / ფუნქციები (Use Case / Activity-D)

ახლა განვიხილოთ ბიზნესპროცესების ობიექტ-ორიენტირებული ანალიზისა და დაპროექტების MsVisio სისტემაში ჩვენი ამოცანის მოდელირების საკითხები. კერძოდ, პირველ რიგში უნდა განისაზღვროს ასაგები კომპიუტერული სისტემის ფუნქციური მოთხოვნები, თავისი ბიზნესპროცესებით და ბიზნესწესებით. 2.6 ნახაზზე მოცემულია ხელფასის დარიცხვის ამოცანის ბიზნესპროცესში მონაწილე მომხმარებელთა როლები (Actors) და ფუნქციები (Actions), გამოყენებით-შემთხვევათა (UseCase-D) დიაგრამის საფუძველზე.



ნახ.2.6. UseCase დიაგრამის ფრაგმენტი (MsVisio გარემოში)

როლი განსაზღვრავს კომპიუტერთან მომუშავის სტატუსს, მაგალითად, ლექტორი, დისპეტჩერი, ბუღალტერი და სხვ. ფუნქცია არის პროცედურა, რომელსაც როლი ასრულებს. მაგალითად, დისპეტჩერი აღრიცხავს ლექტორის მიერ ჩატარებულ მეცადინეობებს, ბუღალტერი ანგარიშობს თანამშრომელთა ხელფასს და ა.შ. როლებისა და ფუნქციების საშუალებით განისაზღვრება კომპიუტერული სისტემის მომხმარებელთა პრივილეგიები და ბაზიდან ამა თუ იმ მონაცემთა მიღების უფლებები. კათედრის გამგე, ლექტორები და დისპეტჩერი ქვეკლასებია განზოგადებული (Subclass, inheritance) კლასისა - კათედრის თანამშრომელი, ამიტომაც ისარი მიმართულია აქეთ.

ფუნქცია (ოვალი) არის ქმედება, რომელსაც როლი ასრულებს. მაგალითად, "ხელფასის დანიშვნა": პიროვნება გარკვეული საკონკურსო წესების საფუძველზე და პირადი განცხადებით საბუთებს წარუდგენს კადრების განყოფილებას (ან საკონკურსო კომისიას), სადაც გარკვეული ეტაპების გავლის შემდეგ, თუ საკითხი დადებითად გადაწყდა, კადრების ინსპექტორი მოამზადებს ბრძანების პროექტს და რექტორის ხელმოწერის შემდეგ ახალ თანამშრომელს, მაგალითად, ლექტორს დაენიშნება თვიური ხელფასი (დავუშვათ 500 ლარი).

ფუნქცია „მეცადინეობის ჩატარება" ევალება ლექტორს და იგი სადისპეტჩერო კომპიუტერში „თითის დაჭერით" აფიქსირებს „ხელმოწერას". არდაფიქსირება ნიშნავს „გაცდენას".

ფუნქციები „მეცადინეობის ჩატარების აღრიცხვა" და „ელ-ტაბელის შევსება" ევალება კომპიუტერ-დისპეტჩერს, რომელიც ლექტორთა ელ-აღრიცხვის ჟურნალიდან მონაცემებს გადაიტანს ელ-ტაბელში. ესაა ყოველთვიური ელ-დოკუმენტი (ფაილი), რომელიც გადაეცემა ბუღალტერიას.

ფუნქცია „ხელფასის ანგარიში" ევალება ბუღალტერს. კათედრაზე N თანამშრომელია სხვადასხვა ხელფასით. ამიტომ იგი კათედრიდან გადმოცემულ სატაბელო მონაცემებს შეადარებს თავის კომპიუტერში არსებულ ინფორმაციას და შემდეგ დაიანგარიშებს თითოეული ლექტორისათვის დარიცხულ თანხებს, დაქვითვებს, სხვადასხვა გადასახადს და ბოლოს ხელზე

ასაღებ თანხას. შედეგები გადაიგზავნება შესაბამის ლექტორთა კონკრეტულ ანგარიშებზე ელექტრონული ანგარიშსწორების მიზნით. 2.6 ნახაზზე ყოველ გამოყენებით შემთხვევას ანუ ფუნქციას (ოვალს) შეესაბამება ერთი აქტიურობის ანუ ქმედების დიაგრამა (Activity Diagram).

თუ ფუნქციას რამდენიმე როლი ასრულებს (ნახ.2.7), მაშინ საჭიროა განისაზღვროს თითოეულის კონკრეტული ოპერაცია (მოქმედება) და შესრულების მიმდევრობის რეგლამენტი (ვინ, რა, როდის უნდა შეასრულოს).

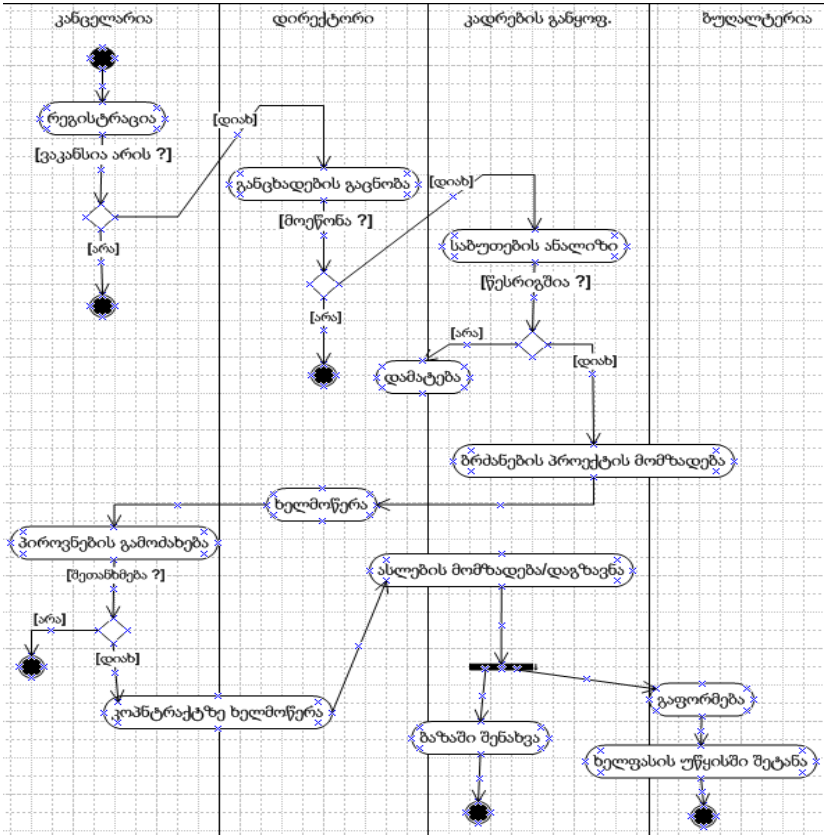
ნახ.2.7

ქმედებათა დიაგრამა (Activity Diagram) ჰგავს ბიზნესპროცესების აღწერის BPMN სქემას, მაგრამ განსხვავებული ფორმით გამოირჩევა (Ms Visio ინსტრუმენტი).

2.8 ნახაზზე განიხილება ამ ფუნქციის შესაბამისი აქტიურობის დიაგრამა: „კონტრაქტის გაფორმება ოფისში ახალი თანამშრომლის მისაღებად“.

აქტიურობის დიაგრამაში ბილიკები ასახავს როლების მართვის სფეროებს. მაგალითად, კანცელარიაში შემოდის პიროვნების განცხადება ამა თუ იმ თანამდებობის დაკავების შესახებ. განცხადება გადის რეგისტრაციას კანცელარიაში. თუ ორგანიზაციაში არ არის ვაკანტური ადგილი, განმცხადებელი უარს ღებულობს. თუ ვაკანსია არსებობს, განცხადება გადაეცემა დირექტორს, რომელიც გადახედავს რა კანდიდატების მონაცემებს, პირადი მოსაზრებით აარჩევს საუკეთესოს, დაადებს რეზოლუციას და გადასცემს კადრების განყოფილებას. კადრების ინსპექტორი გადაამოწმებს ვაკანსიის არსებობას და პიროვნების შრომის წიგნაკს.

Activity-D: ამოცანა - „ახალი თანამშრომლის მიღება“

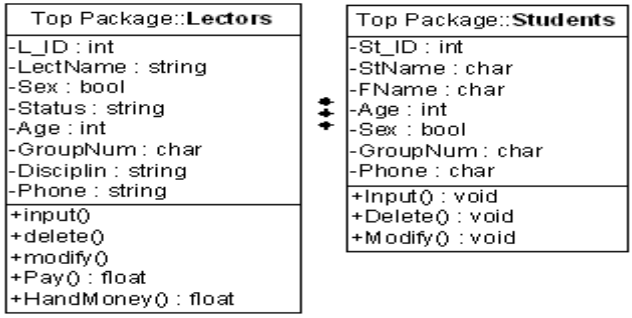


ნახ.2.2.8. Activity დიაგრამის ფრაგმენტი (MsVisio გარემოში)

თუ ყველაფერი წესრიგშია, მოამზადებს ბრძანების პროექტს და გადასცემს დირექტორს ხელმოსაწერად. კანცელარია უგზავნის შეტყობინებას (ურეკავს ტელეფონზე) განმცხადებელს კონტრაქტზე ხელმოწერის მიზნით. ხელმოწერის შემდეგ, კადრების ინსპექტორი ამზადებს ბრძანების ასლებს, რომელთაგან ერთი მიდის ბუღალტერიაში, სადაც მას ხელფასი ენიშნება, მეორე სისტემის მონაცემთა ბაზის ადმინისტრატორთან - კომპიუტერში შესატანად. კანცელარიაში ასევე ამზადებენ პირადობის მოწმობას.

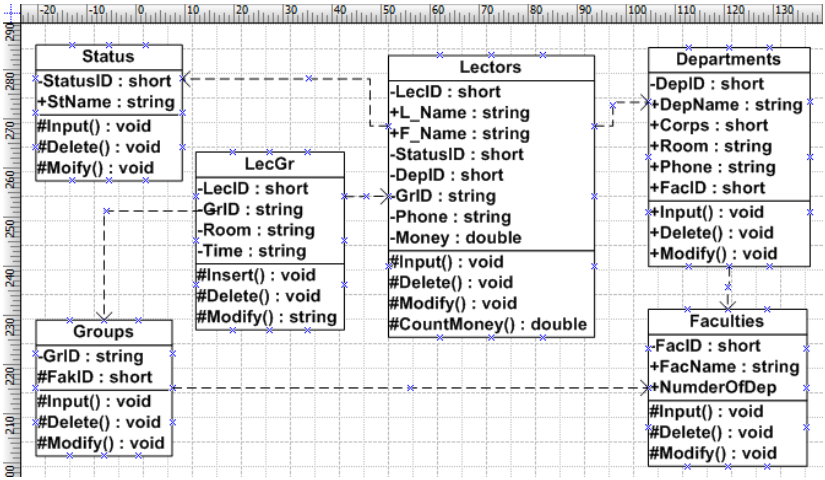
2.3. კლასების აღწერა და კლასთაშორის ასოციაციის დიაგრამა (Class-D, StateChart-D)

კლასი არის „დასახელების“, „კლასის მონაცემების“ და „კლასის მეთოდების“ ინკაფსულაცია. ჩვენი მართვის სფეროს შესაბამისი კლასები ასე უნდა გამოიყურებოდეს (ნახ.2.9):



ნახ.2.9. კლასები: Lectors და Students

კლასთაშორის კავშირების (**Class_Assotiation**) ასაგებად Ms Visio გამოიყენება StateChart დიაგრამა (ნახ.2.10).

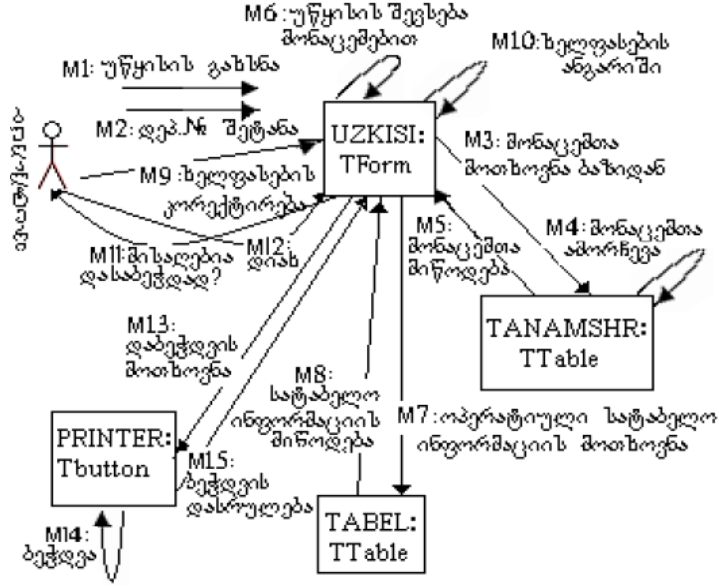


ნახ.2.10. კლასთაშორის კავშირები (StateChart-D)

2.4. ინტერფეისები და სცენარები (Sequence-D, Collaboration-D)

სცენარი არის რომელიმე როლის (Actor) კლასებსა და მის ობიექტებთან მუშაობის პროცესის თანამიმდევრობის აღწერა კონკრეტული ამოცანის (Action) გადასაჭრელად კომპიუტერზე. ამგვარად, ჯერ უნდა აიგოს სცენარი (მაგალითად, MsVisio-ში), თუ როგორ იმუშავებს მომხმარებელი კომპიუტერთან და შემდეგ მოხდეს მისი პროგრამული რეალიზაცია (მაგალითად, C# ენაზე).

სცენარის ასაგებად გამოიყენება მიმდევრობითობის (Sequence-D) და თანამოქმედების (Collaboration-D) დიაგრამები (ნახ.2.11–ა.ბ).

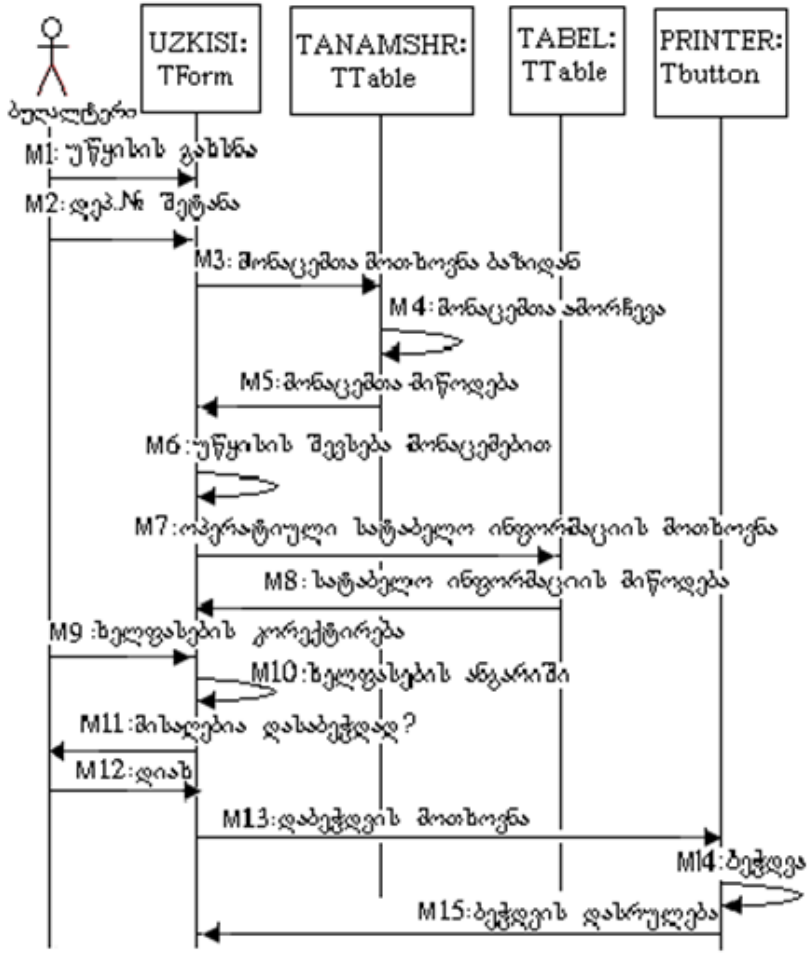


ნახ.2.11–ა. თანამოქმედების დიაგრამა (Collaboration–D)

თანამოქმედების დიაგრამაზე კლასებს შორის ურთიერთობა ასახულია შეტყობინებებისა და ინფორმაციული ნაკადების გაცვლის თვალსაზრისით ანუ რომელი კლასი რომელთან ურთიერთქმედებს და როგორ.

მიმდევრობითობის დიაგრამაზე კი შეტყობინებები (Messages) და ოპერაციები დალაგებულია მათი შესრულების მიმდევრობით დროში.

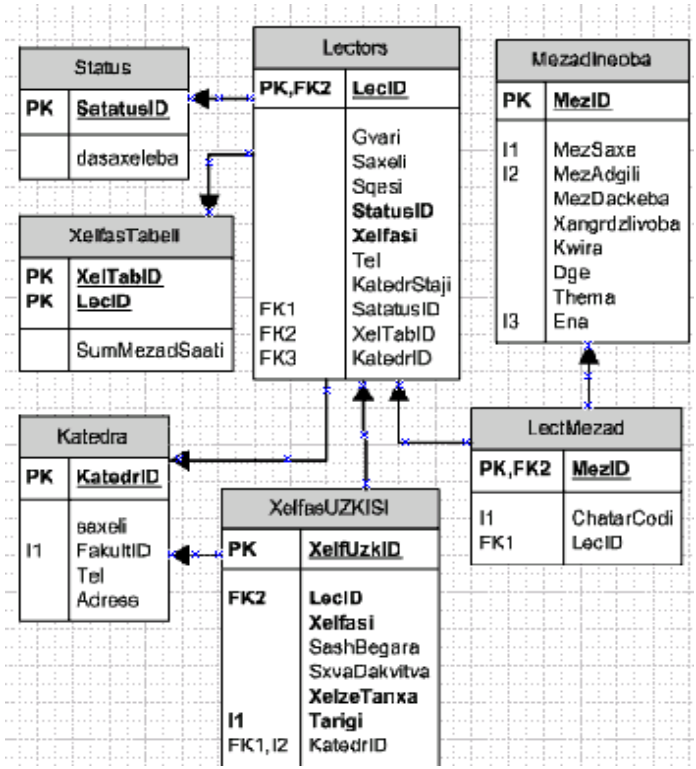
ამოცანა: „ხელფასის დარიცხვის უწყისის მომზადება და ბეჭდვა“



ნახ.2.11–ბ. მიმდევრობითობის დიაგრამა (Sequence–D)

2.5. საპრობლემო სფეროს ER მოდელი (Database-D)

საპრობლემო სფეროს კონცეპტუალური მოდელი აისახება არსთა დამოკიდებულების დიაგრამით (ER-D) ობიექტების, ატრიბუტებისა და ობიექტთაშორისი კავშირებით. დიაგრამას ვაგებთ ჩვენი კლასების საფუძველზე ხელფასის ამოცანისათვის MsVisio/Database ინსტრუმენტის გამოყენებით (ნახ.2.12).



ნახ.2.12. ER მოდელი

III თავი. მონაცემთა ბაზის აგება (Ms Access)

3.1. მონაცემთა ბაზის ცხრილების შექმნა (DB Tables)

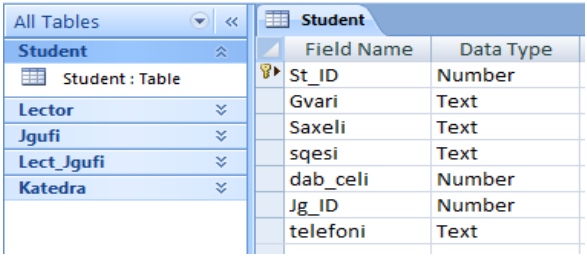
გამოყენებით პროგრამულ აპლიკაციას სჭირდება მონაცემთა ბაზა, რომელშიც შეინახავს ან რომლიდანაც ამოარჩევს მისთვის აუცილებელ ინფორმაციულ ფრაგმენტებს. შერჩეული გვაქვს Ms_Access მონაცემთა ბაზების მართვის სისტემა (რომელიც სტუდენტებმა ისწავლეს წინა სემესტრში), რომლის გამოყენებითაც უნდა შეიქმნას ბაზა და ცხრილები.

დავალება: აამუშავეთ Ms_Access პაკეტი და შექმენით თქვენი პროგრამული აპლიკაციისთვის საჭირო მონაცემთა ბაზა და ცხრილების სტრუქტურები.

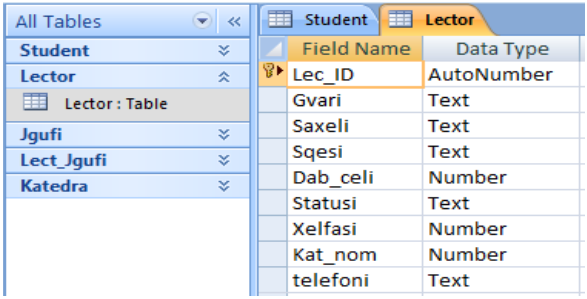
ჩვენ მაგალითისთვის ვიხილავთ უნივერსიტეტის მონაცემთა ბაზას (DB_AccessUni.accdb) ოთხი ცხრილით, რომლებშიც განთავსებულია ინფორმაცია სტუდენტების, ლექტორების, ჯგუფების და აკადემიური საგნების შესახებ.

ცხრილში ***სტუდენტი*** (Student) პირველადი გასაღებური ატრიბუტია St_ID ინდექსი, მისი მეორეული გასაღები - Jg_ID, რომლითაც უკავშირდება ცხრილს ***ჯგუფი*** (Jgufi), პირველადი ინდექსით Jg_ID. ესაა კავშირი 1:N, რომელიც ასახავს ბიზნესწესს (არსებულ კანონზომიერებას), რომ ერთი სტუდენტი შეიძლება იყოს მხოლოდ ერთ ჯგუფში და ერთ ჯგუფში შეიძლება იყოს რამდენიმე (N) სტუდენტი. ასევე, ***ლექტორი*** (Lector) ასწავლის რამდენიმე (N) ჯგუფს, მაგრამ ჯგუფსაც ჰყავს რამდენიმე (M) ლექტორი. ესაა M:N კავშირი. მისი რეალიზაცია არაა შესაძლებელი ***ლექტორის*** და ***ჯგუფის*** პირდაპირი კავშირით (განმეორებადი ველების პრობლემა !). ამისათვის შემოტანილია დამატებითი ცხრილი (რელაცია) ***ლექტორი_ჯგუფი*** (Lect_Jgufi). მასში შედგენილი

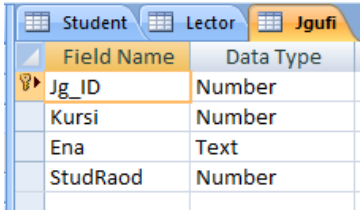
ინდექსი იქნება Lec_ID+Jg_ID, რომლებიც უკავშირდება ცალ-ცალკე *ლექტორს* და *ჯგუფს* (ნახ.3.1, ა–დ).



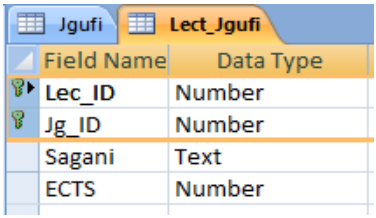
ნახ.3.1-ა. „სტუდენტი“ ცხრილის სტრუქტურა



ნახ.3.1-ბ. „ლექტორი“ცხრილის სტრუქტურა



ნახ.3.1-გ. „ჯგუფი“ ცხრილის სტრუქტურა



ნახ.3.1-დ. „ლექტორი_ჯგუფი“ ცხრილის სტრუქტურა

ასევე შესაძლებელია სხვა ცხრილების დამატება მონაცემთა ბაზაში, როგორიცაა მაგალითად, კათედრა, ფაკულტეტი, ხელფასის უწყისი, გამოცდა, სესიის შედეგები და ა.შ.

3.2. ბაზის ცხრილების შევსება მონაცემებით

მონაცემთა ბაზის აგების მომდევნო ეტაპზე საჭიროა ჩვენ მიერ შექმნილი ცხრილების შევსება ჩანაწერებით, რომლებიც ასახავს რეალურ (ან კვაზირეალურ) სიტუაციას.

დავალება. Ms_Access ბაზის ცხრილებში შეიტანეთ მონაცემები. სტრიქონების რაოდენობა უნდა იყოს საკმარისი ექსპერიმენტების ჩასატარებლად (მინიმუმ 5–20 სტრიქონი).

3.2, ა–დ ნახაზებზე ნაჩვენებია შევსებული ცხრილები.

Student | Lector | Jgufi | Lect_Jgufi

St_ID	Gvari	Saxeli	sqesi	dab_cel	Jg_ID	telefoni
1	ალავიძე	ალეკო	მამარ.	1990	108050	222-22-20
2	ბურდული	წინო	მდედრ.	1991	108050	137-33-33
3	ბურძგლა	დიტო	მამარ.	1989	108835	599-10-20-20
4	გაბედავა	ვახტანგ	მამარ.	1992	108835	333-67-89
5	გაბელია	წაზი	მდედრ.	1992	108935	222-45-67
6	დანელია	მიმოზა	მდედრ.	1990	108935	577-44-44-45
7	ხვიტია	ბუკუ	მამარ.	1992	108935	593-45-67-89
8	აკოფოვი	რობერტინო	მამარ.	1989	108051	297-11-11-11
9	დოლიძე	რიჩარდი	მამარ.	1990	108051	597-34-56-78
10	ზარანდია	მუშნი	მამარ.	1980	108836	597-12-23-34
11	თოფურია	ბლაბი	მდედრ.	1991	108936	577-10-10-10
12	კეკელია	კეკელა	მდედრ.	1992	108936	579-30-30-30
13	გალოგრე	ხვიჩა	მამარ.	1989	108937	270-44-44
14	დუნდუა	გოჩა	მამარ.	1991	108937	599-22-33-44
15	ვასაძე	სოკრატე	მამარ.	1985	108052	577-33-67-55
16	ჯალაღონია	მაცი	მამარ.	1992	108937	577-99-00-00

ნახ.3.2-ა. ცხრილი „სტუდენტი“

Lector | Jgufi | Lect_Jgufi

Lec_ID	Gvari	Saxeli	Sqesi	Dab_celi	Statusi	Xelfasi	Kat_nom	telefoni
6	ბარათელი	აწი	მდედრ.	1970	ასოც.პროფესორი	864	94	599-23-23-23
7	კუცია	თეა	მდედრ.	1987	ლაბორანტი	144	94	577-12-13-14
8	გაბედავა	ომიკო	მამრ.	1950	სრ.პროფესორი	1296	94	577-33-55-22
9	დვალი	დავითი	მამრ.	1950	სრ.პროფესორი	1296	86	599-99-90-90
10	ფიფია	კოჩი	მამრ.	1960	ასოც.პროფესორი	936	86	233-33-46
11	მეფარია	თვარისა	მდედრ.	1980	ას.პროფესორი	288	94	593-55-55-55
12	წინიძე	ლია	მდედრ.	1980	ასოც.პროფესორი	864	94	577-78-89-90
13	ოდიშარია	ოდიშა	მამრ.	1956	ას.პროფესორი	576	86	236-37-38
14	სამხარაძე	ვახშამა	მამრ.	1970	ასოც.პროფესორი	864	51	577-88-99-00

ნახ.3.2-ბ. ცხრილი „ლექტორი“

Student | Lector | Jgufi | Lect_Jgufi | Relatio

Jg_ID	Kursi	Ena	StudRaod
108050	2	ქართული	29
108051	2	ქართული	30
108059	2	რუსული	12
108835	4	ქართული	29
108836	4	ქართული	25
108935	3	ქართული	28
108936	3	ქართული	30
108937	3	ქართული	17
108940	3	ინგლისური	20

ნახ.3.2-გ. ცხრილი „ჯგუფი“

ჩანაწერის წინ „+” სიმბოლო ხსნის კავშირს მეორე, იერარქიულად დაქვემდებარებულ ცხრილთან (ნახ.3.2–ე).

Student | Lector | Jgufi | Lect_Jgufi | Katedr

Lec_ID	Gvari	Saxeli	Sqesi
6	ბარათელი	ანი	ქალი
7	კუცია	თეა	ქალი
8	გაბედავა	ომიკო	კაცი
9	დვალი	დავითი	კაცი
10	ფიფია	კოჩი	კაცი

Jg_ID	Sagani	Add
5	კომპიუტერის არქიტექტურა	
6	კომპიუტერის არქიტექტურა	
7	სერვერული ტექნოლოგიები	
8	სერვერული ტექნოლოგიები	
9	კომპიუტერის არქიტექტურა	

ნახ.3.2-დ. ცხრილი „ლექტორი_ჯგუფი“ საგნების მიხედვით

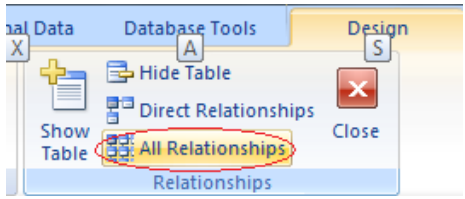
Student | Lector | Jgufi | Lect_Jgufi | Relatio

Lec_ID	Jg_ID	Sagani
8	5	კომპიუტერის არქიტექტურა
8	6	კომპიუტერის არქიტექტურა
8	7	სერვერული ტექნოლოგიები
8	8	სერვერული ტექნოლოგიები
8	9	კომპიუტერის არქიტექტურა
9	7	მათემატიკა
9	8	მათემატიკა
10	12	მოწაცემთა ბაზები
10	13	მოწაცემთა ბაზები

ნახ.3.2–ე. კავშირი ორ ცხრილს შორის

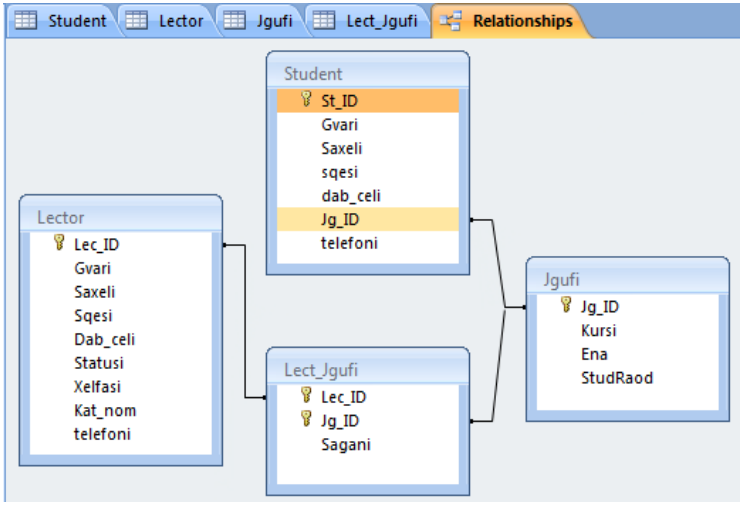
3.3.მონაცემთა ბაზის ცხრილთაშორის კავშირების აგება (Relationships)

მონაცემთა ბაზის ცხრილთაშორის კავშირების ასაგებად აქვე მენიუდან გამოვიყენოთ Database Tools -> Relationships (ნახ.3.3–ა).



ნახ.3.3–ა. ცხრილთაშორის კავშირების დიაგრამა

შედეგში უნდა მივიღოთ ისეთი სახის რელაციური კავშირების დიაგრამა, როგორიც მოცემულია 3.3–ბ ნახაზზე.



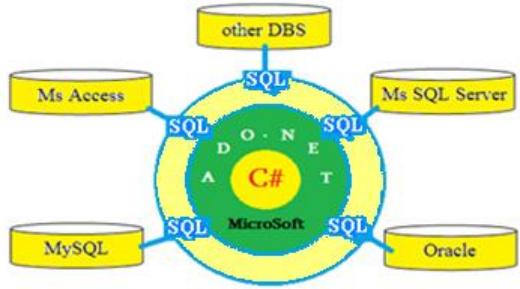
ნახ.3.3–ბ. ცხრილთაშორის კავშირების დიაგრამა

IV თავი. მომხმარებელთა ინტერფეისის დამუშავება

4.1. პროგრამული აპლიკაციების კავშირი მონაცემთა ბაზებთან (ADO.NET დრაივერი)

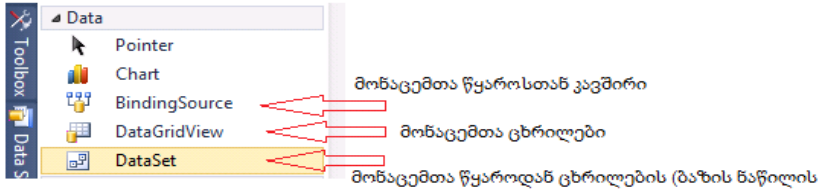
მომხმარებელთა პროგრამულ აპლიკაციებს (დანართებს) აუცილებლად ესაჭიროება ურთიერთქმედება მონაცემთა ცენტრალიზებულ ან ლოკალურ ბაზებთან, როდესაც ისინი მუშაობენ კლიენტის მანქანებზე.

Microsoft Visual Studio .NET Framework პლატფორმა მონაცემთა ბაზებთან სამუშაოდ იყენებს ADO.NET ტექნოლოგიას და SQL ენას (ნახ.4.1). პირველი არის დამაკავშირებელი დრაივერი C# (ან სხვა) ენასა და ბაზებს შორის, მეორე კი - მომხმარებლის საკონტაქტო ენაა ბაზებთან ე.წ. სტრუქტურირებულ მოთხოვნათა ენა [1,7]. აქ მას განვიხილავთ მოკლედ, გაცნობის დონეზე.



ნახ.4.1. C# <-> ADO.NET <-> SQL <–> DBS

C# ენა .NET გარემოში მონაცემთა ბაზებთან (DBS) სამუშაოდ გვთავაზობს შემდეგ კომპონენტებს (ნახ.4.2), რომელთა საფუძველია ADO.NET.



ნახ.4.2

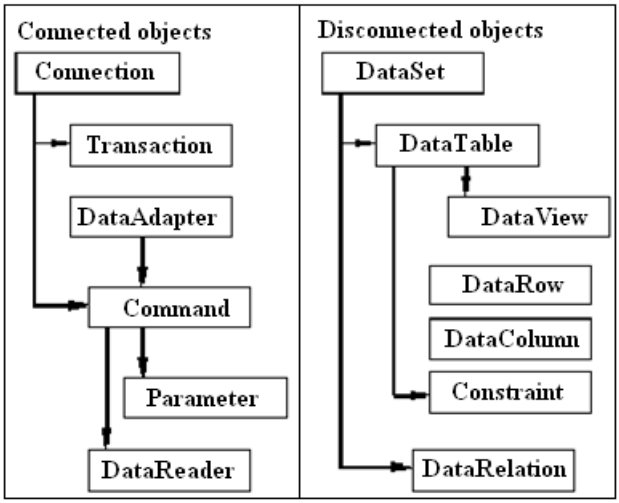
მონაცემებთან მიმართვის ტრადიციული ტექნოლოგიები, ჩვეულებრივ, ახორციელებდა მონაცემების წვდომას წყაროსთან მუდმივი მიერთების გზით. ასეთი მოდელის გამოყენებისას პროგრამული აპლიკაცია გახსნის მონაცემთა ბაზასთან მიერთებას და არ დახურავს მუშაობის დამთავრებამდე. დანართის სირთულის ზრდასთან ერთად იზრდება მონაცემთა ბაზის კლიენტების რაოდენობაც, რაც არაეფექტურს ხდის ბაზასთან მუდმივი მიერთების ტექნოლოგიას. მაგალითად, აპლიკაცია კარგად ემსახურება ორ მიერთებულ კლიენტს, 10-თან უკვე უჭირს მუშაობა და 100-თან საერთოდ ვერ ფუნქციონირებს.

ADO.NET სისტემაში ეს პრობლემები წყდება მონაცემებთან მიმართვის ისეთი მოდელის გამოყენებით, როგორიცაა **გამოყოფილი** მონაცემები. ასეთი მოდელის შემთხვევაში მონაცემთა წყაროსთან მიერთება გახსნილია მხოლოდ გარკვეული პროცედურების შესასრულებლად. მაგალითად, თუ აპლიკაციას დასჭირდა მონაცემები ბაზიდან, იგი მიუერთდება მას ამ მონაცემების გადმოტვირთვამდე, შემდეგ კი მიერთება დაიხურება.

ასევე, როდესაც ხორციელდება მონაცემთა განახლება ბაზაში, მიერთება წყაროსთან განხორციელდება UPDATE ბრძანების შესრულების დამთავრებამდე, შემდეგ იგი დაიხურება. ამგვარად, მონაცემებთან მიერთების დროის (გახსნა-დახურვის პერიოდი) შემცირებით, ADO.NET უზრუნველყოფს სისტემური რესურსების ეკონომიურ გამოყენებას და მონაცემთა წვდომის ინფრასტრუქტურის მასშტაბირებას, მწარმოებლურობის შემცირების გარეშე.

4.2. ADO.NET დრაივერის არქიტექტურა

4.3 ნახაზზე ნაჩვენებია ADO.NET ობიექტური მოდელის შემადგენელი კლასები, რომელთა დანიშნულებასაც მოკლედ შევეხებით ამ პარაგრაფში. სისტემის ობიექტური მოდელი ორი ნაწილისგან შედგება: მარცხენა - მიერთებადი ობიექტები (Connected Objects) და მარჯვენა - განცალკევებადი ობიექტები (Disconnected Objects).



ნახ.4.3

მონაცემებთან მიმართვა ADO.NET-ში ხორციელდება ორი კომპონენტით:

- მონაცემთა ერთობლიობით (DataSet ობიექტით), რომელშიც მონაცემები ინახება ლოკალურ კომპიუტერში;
- მონაცემთა მიმწოდებლით (DataProvider პროვაიდერით), რომელიც ასრულებს შუამავლის ფუნქციას პროგრამასა და მონაცემთა ბაზას შორის.

ობიექტი DataSet. ესაა მონაცემთა წარმოდგენა კომპიუტერის მეხსიერებაში მონაცემთა წყაროსგან

იზოლირებულად. ეს ობიექტი შეიძლება განვიხილოთ, როგორც მონაცემთა ბაზის ფრაგმენტის ლოკალური ასლი (კოპიო).

DataSet-ში მონაცემთა ჩატვირთვა შესაძლებელია ნებისმიერი დასაშვები წყაროდან, მაგალითად, Ms Access, SQL Server ბაზებიდან ან XML ფაილიდან. დასაშვებია მეხსიერებაში ამ მონაცემებით მანიპულირება, აგრეთვე მათი განახლება მთავარი წყაროსაგან დამოუკიდებლად.

ობიექტი DataSet შედგება DataTable ობიექტთა ერთობლიობისგან (ის შეიძლება ცარიელიც იყოს ანუ არ შეიცავდეს არც ერთ DataTable-ს).

ყოველი DataTable ობიექტი კომპიუტერის მეხსიერებაში ასახავს ერთ ცხრილს. მისი სტრუქტურა შეიცავს ორ ერთობლიობას: DataColumns, რომელშიც თავსდება ცხრილის სვეტები და ცხრილის შეზღუდვათა ერთობლიობა. ეს ორი ერთობლიობა ქმნის ცხრილის სქემას.

DataTable ობიექტი შეიცავს აგრეთვე DataRows ერთობლიობას, რომელშიც ინახება DataSet ობიექტის მონაცემები.

გარდა ამისა, DataSet ობიექტი შეიცავს DataRelations ერთობლიობას, რომელიც უზრუნველყოფს კავშირის შექმნას სხვადასხვა ცხრილის სტრიქონებს შორის. DataRelations შეიცავს DataRelation ობიექტთა ერთობლიობას, რომლებიც განსაზღვრავს ცხრილთაშორის კავშირებს (მაგალითად, 1:M კავშირის სარეალიზაციოდ).

დაბოლოს, DataSet ობიექტი შეიცავს ExtendedProperties ერთობლიობას, რომელშიც შეინახება დამატებითი მონაცემები.

მონაცემთა პროვაიდერი. ესაა ურთიერთდაკავშირებულ კომპონენტთა ერთობლიობა, რომელიც უზრუნველყოფს ეფექტურ მაღალმწარმოებლურ კავშირს მონაცემთა ბაზასთან.

.NET Framework-ს აქვს ორი პროვაიდერი: SQL Server .NET Data Provider, რომელიც შექმნილია SQL Server 7.0 ან უფრო მაღალ ვერსიებთან სამუშაოდ და OleDb .NET Data Provider სხვა ტიპის მონაცემთა ბაზებთან დასაკავშირებლად.

მონაცემთა ნებისმიერი პროვაიდერი შედგება მსგავსი უნივერსალური კლასების კომპონენტებისგან:

- Connection, რომელიც უზრუნველყოფს მონაცემთა ბაზასთან მიერთებას;

- Command, რომელიც გამოიყენება მონაცემთა წყაროს სამართავად. იგი გამოიყენებს ბრძანებებს, რომლებიც არ აბრუნებს მონაცემებს, მაგალითად, INSERT, UPDATE და DELETE ან ბრძანებებს, რომლებიც აბრუნებს DataReader ობიექტს (მაგალითად, SELECT);

- DataReader გამოიყენება მხოლოდ ჩანაწერთა ერთობლიობის წასაკითხად მიერთებული მონაცემთა წყაროდან;

- DataAdapter შეავსებს გამოყოფილ DataSet ან DataTable ობიექტს და განაახლებს მათ შედგენილობას.

მონაცემებთან მიმართვა ხორციელდება შემდეგნაირად: ობიექტი Connection აყენებს დანართის (აპლიკაციის) მონაცემთა ბაზასთან მიერთებას, რომელიც პირდაპირ მისაწვდომია Command და DataAdapter ობიექტებისთვის. Command ობიექტი უზრუნველყოფს ბრძანებათა შესრულებას უშუალოდ მონაცემთა ბაზაში. თუ შესასრულებელი ბრძანება აბრუნებს რამდენიმე მნიშვნელობას, მაშინ Command ხსნის მათთან მიმართვას DataReader ობიექტის საშუალებით. მიღებული შედეგები შესაძლებელია დამუშავდეს უშუალოდ დანართის კოდით ან DataSet ობიექტით, რომელიც შეივსება DataAdapter ობიექტის დახმარებით. მონაცემთა ბაზის განახლებისთვის ასევე გამოიყენება Command და DataAdapter ობიექტები.

ობიექტი Connection გვთავაზობს მიერთებას მონაცემთა ბაზასთან. Visual Studio .NET-ს აქვს Connection-ის ორი კლასი: SqlConnection (MsSQL_Server-თან შესაერთებლად) და OleDbConnection (სხვა ტიპის მონაცემთა ბაზებთან დასაკავშირებლად). მონაცემთა ბაზასთან კავშირის არხის გასახსნელი აუცილებელი მონაცემები ინახება Connection ობიექტის ConnectionString თვისებაში. ეს ობიექტი ინახავს აგრეთვე მეთოდებს, რომლებიც საჭიროა მონაცემთა დასამუშავებლად ტრანზაქციების გამოყენებით.

Command ობიექტს აქვს ორი კლასი: SqlCommand და OleDbCommand. იგი უზრუნველყოფს ბრძანებათა გამოყენებას

მონაცემთა ბაზაზე, რომელთანაც დამყარებულია კავშირი (მიერთება). აქ შეიძლება გამოყენებულ იქნეს შენახვადი პროცედურები (Stored Procedures), SQL-ენის ბრძანებები, აგრეთვე ოპერატორები მთლიანი ცხრილების მისაღებად. Command ობიექტს აქვს სამი მეთოდი:

- Execute Non Query: იყენებს ბრძანებებს, რომლებიც არ აბრუნებს მონაცემებს, მაგალითად, INSERT, UPDATE და DELETE;
- Execute Scalar: იყენებს მოთხოვნებს მონაცემთა ბაზისადმი, რომლებიც აბრუნებს მხოლოდ ერთ მნიშვნელობას;
- Execute Reader: აბრუნებს საშედეგო ერთობლიობას DataReader ობიექტის საშუალებით.

ობიექტი DataReader გვთავაზობს ნაკადს მონაცემთა ბაზის ჩანაწერების ერთობლიობით, ოღონდ მხოლოდ ერთი მიმართულებით წასაკითხად. მონაცემთა პროვაიდერის სხვა კომპონენტებისგან განსხვავებით DataReader-ის ეგზემპლარების შექმნა პირდაპირ არაა დასაშვები. მისი მიღება შეიძლება Command ობიექტის ExecuteReader მეთოდებით:

- SqlCommand.ExecuteReader მეთოდი აბრუნებს SqlDataReader ობიექტს;
- OleDbCommand.ExecuteReader მეთოდი კი - OleDbDataReader ობიექტს.

თუ DataReader ობიექტის შემცველი მონაცემების ჩაწერა დისკზე არაა საჭირო, მაშინ ეს სტრიქონები შეიძლება პირდაპირ გადაეგზავნოს დანართს. ვინაიდან დროის ნებისმიერ მომენტში მეხსიერებაში იმყოფება მხოლოდ ერთი სტრიქონი, DataReader ობიექტის გამოყენება თითქმის არ ამცირებს სისტემის მწარმოებლურობას, ოღონდ მოითხოვს მონოპოლიურ მიმართვას გახსნილ Connection ობიექტზე DataReader ობიექტის სასიცოცხლო დროის განმავლობაში.

ობიექტი DataAdapter არის ADO.NET-ის ძირითადი კლასი, რომელიც უზრუნველყოფს გამოყოფილ მონაცემებთან მიმართვას. არსებითად, იგი ასრულებს შუამავლის ფუნქციებს

მონაცემთა ბაზისა და DataSet ობიექტის ურთიერთ-ქმედებისთვის.

Fill მეთოდის გამოძახებისას DataAdapter ობიექტი მონაცემებით შეავსებს DataTable-ს ან DataSet-ს მონაცემთა ბაზიდან. მონაცემების დამუშავების შემდეგ, რომლებიც ჩატვირთულია მეხსიერებაში, შესაძლებელია მოდიფიცირებული ჩანაწერების მოთავსება მონაცემთა ბაზაში, DataAdapter ობიექტის Update მეთოდის გამოძახებით. DataAdapter-ს აქვს ოთხი თვისება, რომლებიც მონაცემთა ბაზის ბრძანებებია:

- SelectCommand შეიცავს ტექსტს ან ბრძანების ობიექტს, რომელიც ახორციელებს მონაცემთა ბაზიდან ამორჩევას (მაგალითად, მეთოდი Fill);
- InsertCommand შეიცავს ტექსტს ან ბრძანების ობიექტს, რომელიც ახორციელებს სტრიქონის ჩასმას ცხრილში;
- DeleteCommand შეიცავს ტექსტს ან ბრძანების ობიექტს, რომელიც ახორციელებს სტრიქონის წაშლას ცხრილიდან;
- UpdateCommand შეიცავს ტექსტს ან ბრძანების ობიექტს, რომელიც ახორციელებს მნიშვნელობათა განახლებას მონაცემთა ბაზაში.

Update მეთოდის გამოძახებისას ყველა შეცვლილი მონაცემი კოპირდება DataSet ობიექტიდან მონაცემთა ბაზაში, შესაბამისი ბრძანებების - InsertCommand, DeleteCommand ან UpdateCommand გამოყენებით.

4.3. ინტერფეისის კავშირი მონაცემთა ბაზასთან

Visual Studio .NET სისტემას აქვს სტანდარტული ოსტატი პროგრამებისა და დიზაინერების სიმრავლე, რომელთა საშუალებითაც ადვილად და ეფექტურად ხორციელდება მონაცემებთან წვდომის არქიტექტურა აპლიკაციების დამუშავების პროცესში. ამასთან, ADO.NET ობიექტური მოდელის ყველა შესაძლებლობა მისაწვდომია პროგრამულად, რაც უზრუნველყოფს არასტანდარტული ფუნქციების

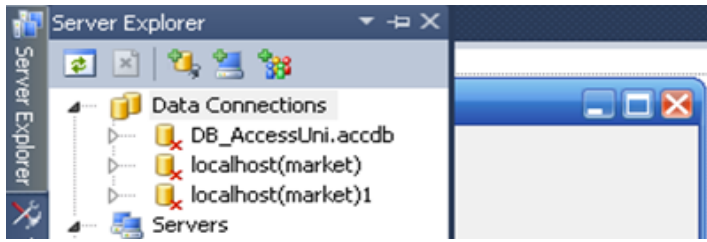
რეალიზაციის ან დანართების აგების შესაძლებლობას, რომელიც მომხმარებლის მოთხოვნილებებზეა ორიენტირებული.

აქ გავეცნობით, როგორ დავუკავშირდეთ მონაცემთა ბაზას ADO.NET-ის გამოყენებით, როგორ ამოვიღოთ საჭირო მონაცემები და გადავცეთ პროგრამულ აპლიკაციას. ეს საკითხები შეიძლება შესრულდეს Visual Studio .NET-ის გრაფიკული ინსტრუმენტებით და პროგრამულად.

C# პროგრამულ აპლიკაციაში არსებობს მონაცემთა ბაზასთან მიერთების რამდენიმე ხერხი. ამის განხორციელება ყველაზე მარტივია Visual Studio .NET-ის გრაფიკული ინსტრუმენტით. მონაცემთა წყაროსთან (DataSource) მიერთებისა და მართვისათვის გამოიყენება ფანჯარა Server Explorer.

ძირითადი ამოცანა, რომელსაც აქ განვიხილავთ, არის ADO.NET პროგრამული პაკეტის გამოყენებით მომხმარებელთა სამუშაო ინტერფეისის დამუშავების სადემონსტრაციო მაგალითის აგება. ამასთანავე, მონაცემთა ბაზების სახით უნდა გამოვიყენოთ Ms_Access პაკეტით აგებული ცხრილები.

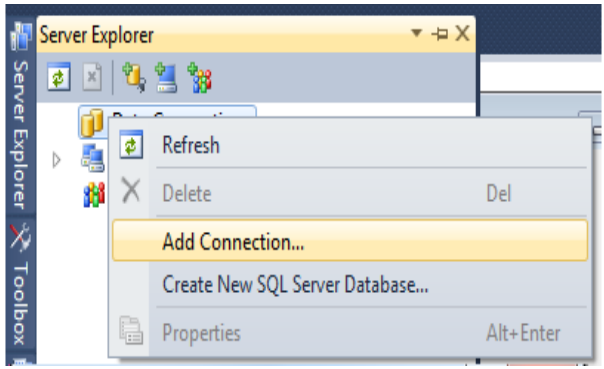
.NET სამუშაო გარემოს ჩატვირთვის შემდეგ საჭიროა Server Explorer-ის გახსნა და ბაზებთან კავშირის შემოწმება (მაგალითად, ნახ.4.4).



ნახ.4.4

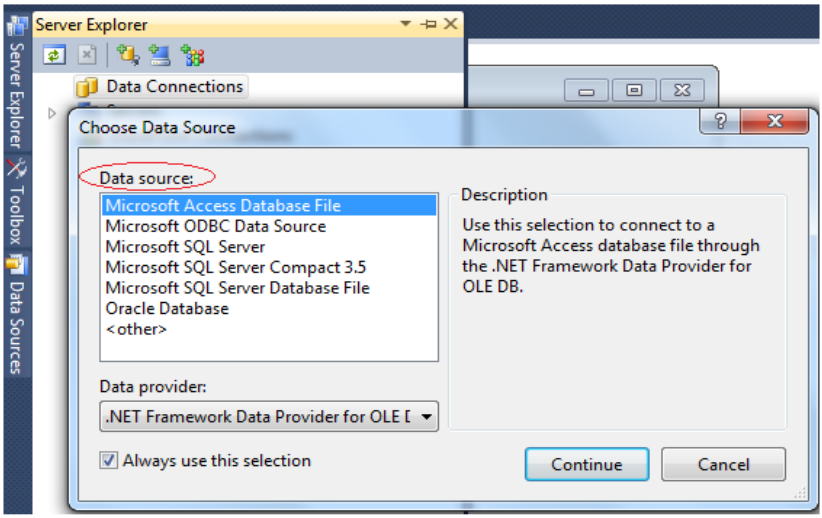
ახლა განვიხილოთ MsAccess ბაზასთან მუშაობის საკითხები.

სისტემის მენიუდან View | Server Explorer-ით გამოვიტანოთ ფანჯარა (ნახ.4.5) და ავირჩიოთ Add Connection.



ნახ.4.5

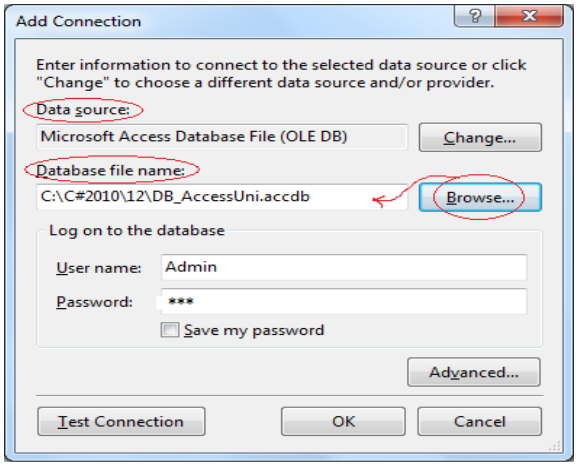
4.6 ნახაზზე Data Source ველში ავირჩიოთ სტრიქონი Microsoft Access Database File და Continue.



ნახ.4.6

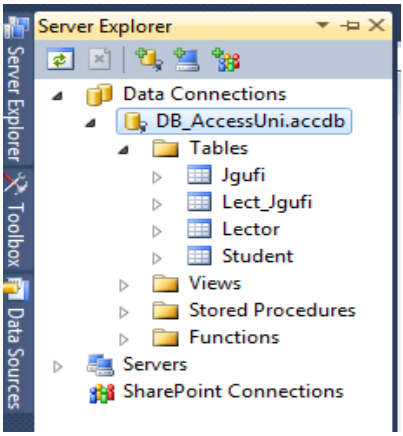
მივიღებთ 4.7 ფანჯარას, რომელშიც Browse ღილაკით გამოვიძახებთ კატალოგის მართვის დიალოგის ფანჯარას და

მივუთითებთ ჩვენ მიერ წინასწარ მომზადებულ Ms Access-ის ბაზის ფაილს, როგორც ეს Database file name ველშია ჩაწერილი.



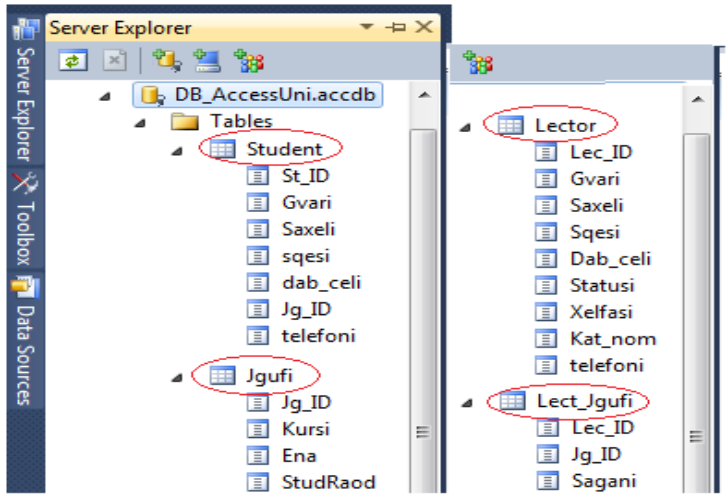
ნახ.4.7

საჭიროების შემთხვევაში მიეთითება User name და Password. შემდეგ Server Explorer-ში გამოჩნდება 4.8 ნახაზზე მოცემული სურათი.



ნახ.4.8

როგორც ვხედავთ, Data Connection-ში უნივერსიტეტის მონაცემთა ბაზის ფაილი **DB_AccessUni.accdb** გამოჩნდა, რომელიც შედგება ოთხი ცხრილისგან: Jgufi, Lect_Jgufi, Lector და Student. ცხრილები შედგება ველებისგან, რომელთაგან ერთ-ერთი გასაღებურია (ინდექსი): Lec_ID, St_ID, Jg_ID და ერთიც შედგენილი გასაღებია ორი ატრიბუტით: Lec_ID+Jg_ID (ნახ.4.9).



ნახ.4.9

ამგვარად, მონაცემთა ბაზა DB_AccessUni.acce მზადაა.

4.4. აპლიკაციის C# კოდიდან Ms Access ბაზის წვდომა და მოთხოვნების დამუშავება

ახლა განვიხილოთ C# პროგრამიდან მონაცემთა ბაზასთან წვდომის საკითხი. ეს პროცესი ოთხი ბიჯისგან შედგება:

- მონაცემთა ბაზასთან მიერთება (რაც ზემოთ განვიხილეთ Server Explorer->Data Connection-ში);
- SQL ბრძანების გადაცემა მონაცემთა ბაზაზე;
- SQL ბრძანების შეფასება (და შესრულება);
- მონაცემთა ბაზასთან კავშირის დახურვა.

SQL ბრძანება სტრუქტურირებული მოთხოვნების ენაზე დაწერილი სკრიპტია, რომელიც გასაგებია მონაცემთა ბაზების მართვის სისტემისთვის და ასრულებს მას. ძირითადად, არსებობს ორი ტიპის მოთხოვნა:

- Select : მონაცემთა ამორჩევის SQL ბრძანება;
- insert, delete, update: მონაცემთა ბაზაში ცვლილებების განსახორციელებელი SQL ბრძანებები.

C# პროგრამულ პროექტს მონაცემთა ბაზასთან სამუშაოდ სჭირდება სახელსივრცე OleDb, რომელიც using.System.Data.OleDB ბრძანებითაა რეალიზებული.

SQL ბრძანებები Ms_Access ბაზაში გადაიგზავნება OleDb სახელსივრცის OleDBCommand კლასის ობიექტით. ამ კლასის ორი მნიშვნელოვანი თვისებაა: Connection (დავალება ბაზასთან დასაკავშირებლად, საითაც გაიგზავნება SQL მოთხოვნა) და CommandText (თვით SQL ბრძანების ტექსტი).

მოთხოვნის ტიპებზე დამოკიდებულებით (არჩევითი, ცვლილებების) OleDBCommand კლასს გააჩნია შემდეგი მეთოდები:

ExecuteReader() - აქვს select მოთხოვნის გაგზავნის და შედეგების მიღების ფუნქცია;

ExecuteNonQuery() - ემსახურება ცვლილების (insert, delete, update) აქციის ჩატარებას და რიცხვის მიღებას, რომელიც გვიჩვენებს, მონაცემთა ბაზის რამდენ ჩანაწერს შეეხო ეს პროცედურა.

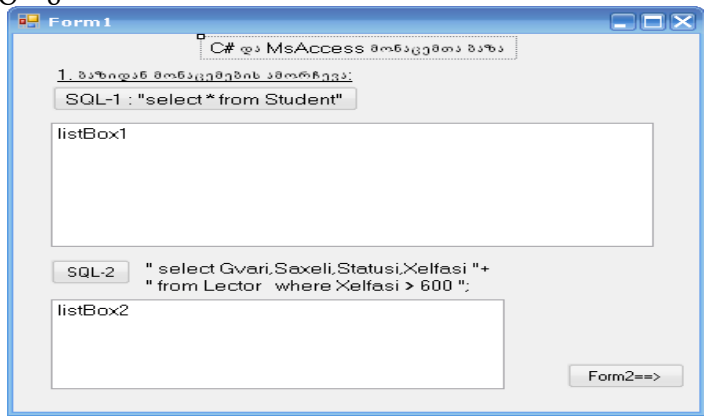
მონაცემთა ბაზიდან select მოთხოვნით ამორჩეული ჩანაწერები (ველებით და მნიშვნელობებით) ინახება OleDb სახელსივრცის OleDBReader კლასის ობიექტში. განვიხილოთ კონკრეტული მაგალითი.

ამოცანა-4.1. DB_AccessUni.acce უნივერსიტეტის მონაცემთა ბაზაში:

1. „ვნახოთ ყველა სტუდენტის ყველა ატრიბუტის მნიშვნელობა (ანუ მთლიანი ინფორმაცია)“;
2. „ვიპოვოთ იმ ლექტორთა გვარი, სახელი, სტატუსი და ხელფასი, რომელთა ხელფასი 600 ლარზე მეტია“.

4.10-ა ნახაზზე 1-ელ მოთხოვნას შეესაბამება SQL-1, ხოლო მეორეს - SQL-2 "select" კონსტრუქცია. ისინი ორ ღილაკზეა მიმაგრებული. საილუსტრაციო მაგალითში შედეგები გამოიტანება listBox1 და listBox2 ველებში.

C# პროგრამის კოდი ამ ორი ღილაკისთვის მოცემულია 4.1 ლისტინგში.



ნახ.4.10

```
// ლისტინგი_4.1 – C# + Ms Access -------
private void button1_Click(object sender, EventArgs e)
{
  OleDbConnection con = new OleDbConnection();
  OleDbCommand cmd = new OleDbCommand();
  OleDbDataReader reader;

con.ConnectionString =
     "Provider=Microsoft.ACE.OLEDB.12.0;" +
     "Data Source=C:\\C#2010\\WinADO\\DB_AccessUni.accdb";

cmd.Connection = con;
cmd.CommandText = "select * from Student";
try
 {
   con.Open();
   reader = cmd.ExecuteReader();
   listBox1.Items.Clear();
```

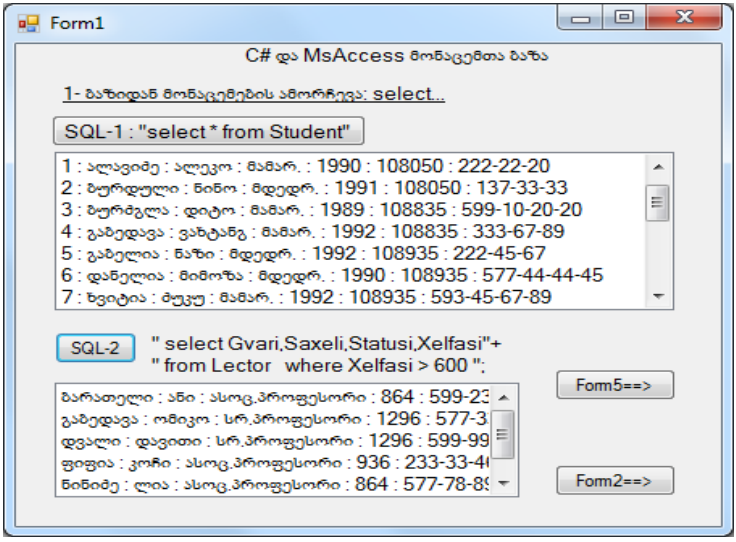
```
    while (reader.Read())
     {
      listBox1.Items.Add(reader["St_ID"] + " : " +
                  reader["Gvari"] + " : " +
                  reader["Saxeli"] + " : " +
                  reader["sqesi"] + " : " +
                  reader["dab_celi"] + " : " +
                  reader["Jg_ID"] + " : " +
                  reader["telefoni"]);
     }
     reader.Close();
     con.Close();
    }
   catch (Exception ex)
    {
     MessageBox.Show(ex.Message);
    }
  }
// end button1_click ----------------

private void button2_Click(object sender, EventArgs e)
 {
   OleDbConnection con = new OleDbConnection();
   OleDbCommand cmd = new OleDbCommand();
   OleDbDataReader reader;

 con.ConnectionString =
      "Provider=Microsoft.ACE.OLEDB.12.0;" +
      "Data Source=C:\\C#2010\\WinADO\\DB_AccessUni.accdb";
 cmd.Connection = con;
 cmd.CommandText =
    "select  Gvari,Saxeli,Statusi,Xelfasi " +
    "from Lector " +
    "where Xelfasi > 600 ";
 try
  {
   con.Open();
   reader = cmd.ExecuteReader();
   listBox2.Items.Clear();
   while (reader.Read())
    {
     listBox2.Items.Add( reader["Gvari"] + " : " +
```

```
                    reader["Saxeli"] + " : " +
                    reader["Statusi"] + " : " +
                    reader["Xelfasi"]);
      }
     reader.Close();
     con.Close();
    }
   catch (Exception ex)
     {
       MessageBox.Show(ex.Message);
     }
  }
// end button2_click ----------------
```

შენიშვნა. პროგრამის თავში using-სტრიქონებში უნდა ჩაემატოს:

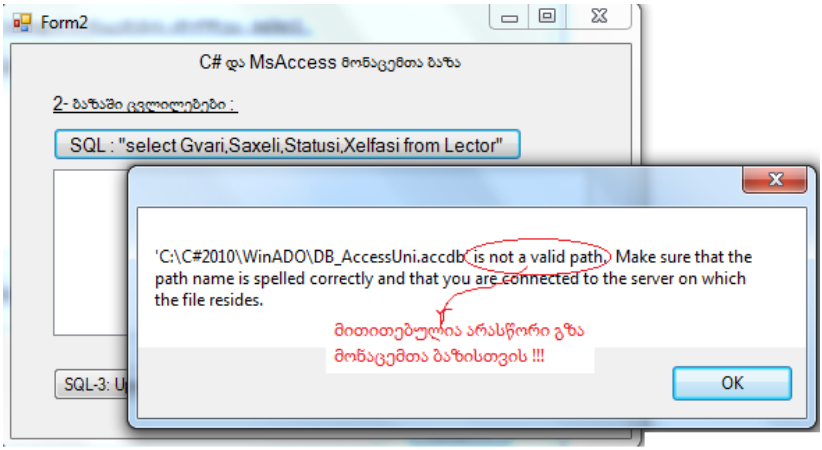
```
using System.Data.OleDb;
```

4.11 ნახაზზე ნაჩვენებია აღნიშნული მოთხოვნების შესრულების შედეგები MsAccess ბაზის Student და Lector ცხრილებიდან. პირველში „ * ” ნიშნავს „ყველა“ ველს, ხოლო მეორეში აიღება მხოლოდ „გვარი, სახელი, სტატუსი და ხელფასი“.



ნახ.4.11

პროგრამაში **try…catch** ბლოკით ხდება მონაცემთა ბაზასთან წვდომის პროცესში შესაძლო შეცდომების აღმოჩენა, რაც აადვილებს პროგრამისტის მუშაობას. მაგალითად, ინფორმაციის მიღება, რომ მონაცემთა ბაზის ფაილი არაა მითითებულ patch-კატალოგში (ნახ.4.12) ან, რომ SQL მოთხოვნის სინტაქსში შეცდომაა და ა.შ.



ნახ.4.12

Open() მეთოდით ხდება პროგრამის კავშირის გახსნა ბაზასთან. შემდეგ ExecuteReader() მეთოდით მოთხოვნა გადაეგზავნება ბაზას. შედეგები ბრუნდება OleDbReader კლასით, რაც ხორციელდება reader მიმთითებლით (მაჩვენებლით).

ვინაიდან წინასწარ არაა ცნობილი, რამდენი სტრიქონი იქნება შედეგში, გამოიყენება ListBox, რომელიც წინასწარ სუფთავდება.

Read() მეთოდი ბაზიდან გვაწვდის ერთ ჩანაწერს (სტრიქონს) და ამავდროულად სპეცმაჩვენებლით მიუთითებს მომდევნო ჩანაწერზე. თუ ჩანაწერი ბოლოა, მაშინ მაჩვენებლის მნიშვნელობა ხდება false. ეს მართვა ხორციელდება while ციკლით try ბლოკში.

ჩანაწერის შიგნით ველების მნიშვნელობები შეესაბამება მათ ნომრებს ან დასახელებებს. შესაძლებელია ასევე ყველა ველის გამოტანა („ * “ -ით), რომლებიც სპეცგამყოფითაა (მაგ., „ : “) დაცილებული.

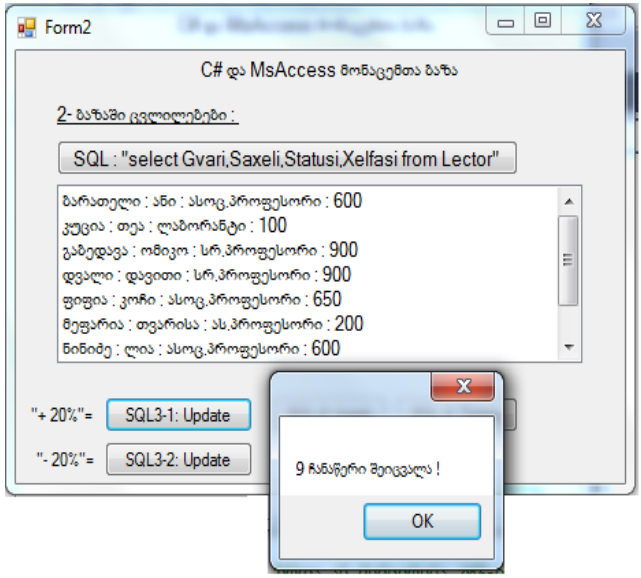
ბოლოს, Reader ობიექტი და კავშირი უნდა დაიხუროს Close() მეთოდით.

ამოცანა-4.2. მონაცემთა ბაზაში „უნივერსიტეტი“ DB_AccessUni.acce საჭიროა ცვლილებების განხორციელება insert(), delete() და update() მეთოდების გამოყენებით. Form2-ზე მოვათავსოთ შესაბამისი ელემენტები და განვახორციელოთ ტრანზაქციები:

ა) ყველა ლექტორის ხელფასი 20%-ით გაიზარდოს. (ამასთანავე შესაძლებელი უნდა იყოს საწყისი მონაცემების აღდგენა, ანუ შემცირდეს ხელფასები 20 %-ით);

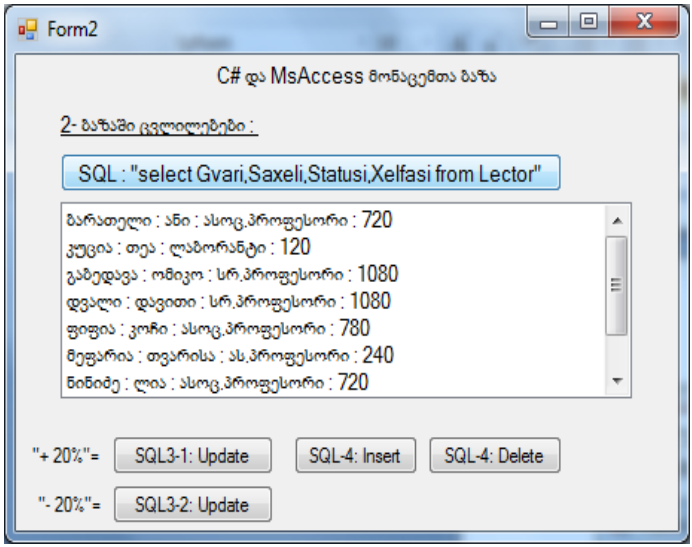
ბ) 108935 ჯგუფში დაემატოს ახალი სტუდენტი გვარით „ახალაძე“, სახელით „ნოუთბუკა“ და ა.შ.;

გ) ამოიშალოს ბაზიდან 108836 ჯგუფი, რომელმაც დაასრულა 4-წლიანი სწავლების კურსი.



ნახ.4.13

ლექტორთა ხელფასების შეცვლილი შედეგები მოცემულია 4.14 ნახაზზე.



ნახ.4.14

SQL3-2 ღილაკით შემცირდება ხელფასის რაოდენობა 20%-ით ანუ აღდგება პირველადი მონაცემები ბაზაში.

შესაბამისი update - კოდი მოცემულია 4.2 ლისტინგში.

```
// ლისტინგი_4.2 --- Ms Acees  “update” ------
using System;
using System.Data.OleDb;
using System.Windows.Forms;

namespace WinADO
{
 public partial class Form2 : Form
 {
  public Form2() { InitializeComponent();  }
  private void button1_Click(object sender, EventArgs e)
   {
    OleDbConnection con = new OleDbConnection();
    OleDbCommand cmd = new OleDbCommand();
    int Raod;
    // ხელფასის ზრდა ან შემცირება 20%-ით ----
```

```
    con.ConnectionString =
           "Provider=Microsoft.ACE.OLEDB.12.0;" +
           "Data Source=C:\\C#2010\\12\\DB_AccessUni.accdb";
    cmd.Connection = con;
 if(ReferenceEquals(sender,button1))
     // op = "*"  or " /" -----
  cmd.CommandText = "update Lector set Xelfasi=Xelfasi * 1.2";
 else
  cmd.CommandText = "update Lector set Xelfasi=Xelfasi / 1.2";
 try
  {
     con.Open();
     Raod = cmd.ExecuteNonQuery();
      MessageBox.Show(Raod + " ჩანაწერი შეიცვალა !");
     con.Close();
  }
 catch (Exception ex)
    {
      MessageBox.Show(ex.Message);
    }
}

private void button4_Click(object sender, EventArgs e)
 {
   OleDbConnection con = new OleDbConnection();
  OleDbCommand cmd = new OleDbCommand();
  OleDbDataReader reader;

  con.ConnectionString =
            "Provider=Microsoft.ACE.OLEDB.12.0;" +
           "Data Source=C:\\C#2010\\12\\DB_AccessUni.accdb";

  cmd.Connection = con;
  cmd.CommandText = "select Gvari,Saxeli,Statusi,
                                        Xelfasi from Lector";
  try
   {
     con.Open();
     reader = cmd.ExecuteReader();
     listBox1.Items.Clear();
     while (reader.Read())
```

```
      {
       listBox1.Items.Add(reader["Gvari"] + " : " +
                          reader["Saxeli"] + " : " +
                          reader["Statusi"] + " : " +
                          reader["Xelfasi"]);
      }
    reader.Close();
      con.Close();
     }
  catch (Exception ex)
    {
      MessageBox.Show(ex.Message);
    }
```

კოდში update ტიპის ცვლილებისათვის DB_MsAccessUni.accedb ფაილში გამოყენებულია კონსტრუქცია:

“update Lector set Xelfasi=Xelfasi * 1.2"

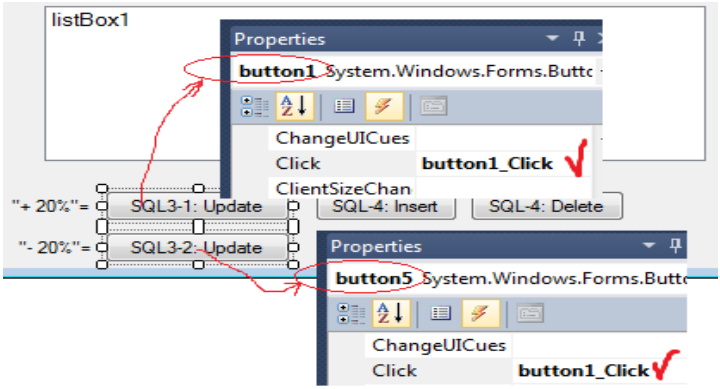
რაც ნიშნავს ცხრილის აქტუალიზაციას (update… set), Lector სახელით და ცხრილის ველის (ატრიბუტის) ცვლილების ალგორითმს (Xelfasi=Xelfasi * 1.2), რომელიც უნდა შესრულდეს ჩანაწერებზე;

try…catch ბლოკში იხსნება ბაზა (Open-ით) და გამოიძახება მეთოდი `ExecuteNonQuery()`, რომელიც დააბრუნებს ჩანაწერების რაოდენობას (Raod), რომელთაც ცვლილება შეეხო. ეს ინფორმაცია გამოიტანება MessageBox-ით.

მონაცემთა ბაზის Lector ცხრილში Xelfasi ველისათვის უნდა მოხდეს მნიშვნელობათა 20%-ით გაზრდა (button1: SQL3-1) ან შემცირება (button5: SQL3-2, ნახ.4.15).

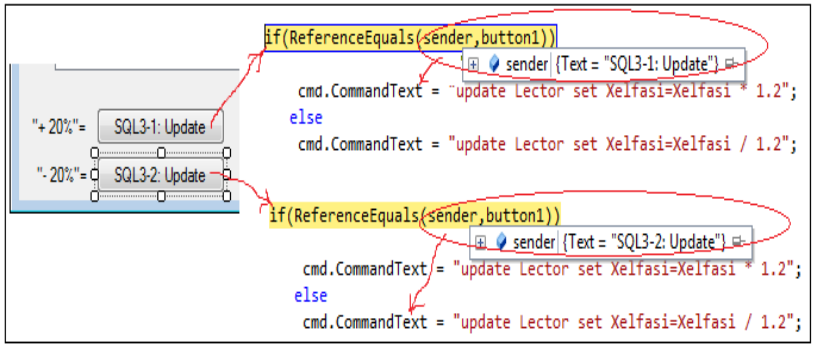
ამის განსახორციელებლად კოდში გამოყენებულია მეთოდი ReferenceEquals(sender object). ამ მეთოდით განისაზღვრება, აქვს თუ არა ორ ობიექტს ერთი მისამართი ანუ ინახება თუ არა ისინი მეხსიერების ერთი და იგივე უჯრედში. თუ „კი“, მაშინ „true“ და შესრულდება გამრავლება (ხელფასის მომატება), ხოლო თუ

სხვადასხვა ობიექტია, მაშინ - „false” და შესრულდება გაყოფა (ხელფასის შემცირება).



ნახ.4.15

კოდის ReferencialEquals(sender, button1) მეთოდში, იმის მიხედვით, თუ რომელი ბუტონი (SQL3-1 თუ SQL3-2) იქნება არჩეული, sender-ს მიენიჭება button1 ან button5 მიმთითებლის მნიშვნელობა (ნახ.4.16).



ნახ.4.16

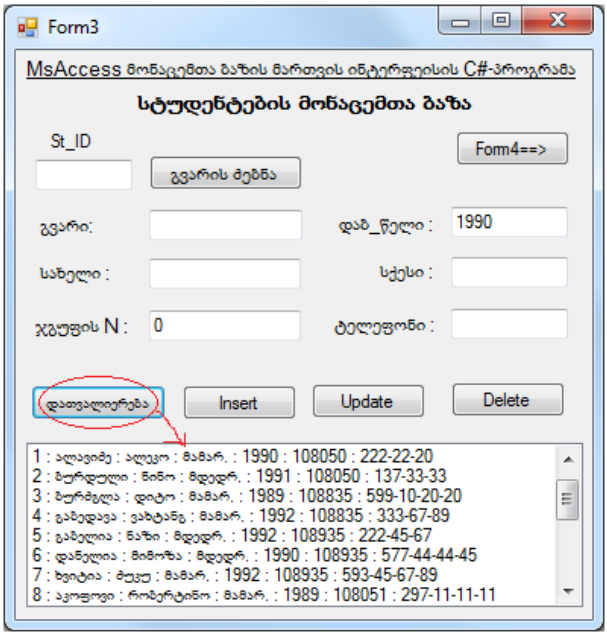
ამგვარად, როდესაც sender და button1 ერთი და იგივე მისამართზეა, მაშინ ხდება ხელფასის მომატება. დასასრულ, პროგრამის button4_Click ღილაკით listBox1-ში გამოიტანება

MsAccess ბაზის Lector ცხრილის განახლებული მონაცემები (მომატებული ან დაკლებული ხელფასებით).

ამოცანა-4.3. საჭიროა აიგოს კოდი **მომხმარებლის ინტერფეისი** სტუდენტთა რეგისტრაციისათვის, რომელიც Ms Access მონაცემთა ბაზასთან იმუშავებს, განახორციელებს ჩანაწერების ძებნის, ჩამატების, წაშლის და კორექტირების სერვისულ ფუნქციებს.

საწყისი ფორმა 4.17 ნახაზზეა ნაჩვენები. "დათვალიერება" ღილაკი კავშირშია ცხრილ **სტუდენტთან** და გამოაქვს ჩანაწერები textBox1-ში.

პროგრამის საწყისი ფრაგმენტი (ბაზასთან მიერთება, მონაცემთა ინიციალიზაცია) და ღილაკი "დათვალიერება" მოცემულია 4.3 ლისტინგში.



ნახ.4.17

```
// ლისტინგი_4.3  --- MsAccess ბაზის მონაცემების მართვა ----
using System;
using System.Collections;
using System.Data.OleDb;
using System.Drawing;
using System.Windows.Forms;
nam espace WinADO
{
 public partial class Form3 : Form
  {
    public Form3() {InitializeComponent(); }
    OleDbConnection con = new OleDbConnection();
    OleDbCommand cmd = new OleDbCommand();
    OleDbDataReader reader;
    ArrayList stNummer = new ArrayList();

 private void Form3_Load(object sender, EventArgs e)
  {
   con.ConnectionString =
      "Provider=Microsoft.ACE.OLEDB.12.0;" +
     "Data Source=C:\\C#2010\\12\\DB_AccessUni.accdb";
    cmd.Connection = con;
  }

 private void button1_Click(object sender, EventArgs e)
  {
    Datvaliereba();
  }
 private void Datvaliereba()
  {
   try
     { con.Open();
       cmd.CommandText = "select * from Student";
       Ekranze_gamotana();
     }
    catch (Exception ex)
      {
        MessageBox.Show(ex.Message);
      }
     con.Close();
     textBox1.Text=""; // St-ID
     textBox2.Text=""; // Gvari
```

```
      textBox3.Text=""; // Saxeli
      textBox4.Text=""; // sqesi
      textBox5.Text="1990"; // dab_celi
      textBox6.Text="0"; // Jg_ID
      textBox7.Text=" "; // telefoni
   }
  private void Ekranze_gamotana()
  {
    DateTime DabTarigi;
    reader = cmd.ExecuteReader();
     listBox1.Items.Clear();
     stNummer.Clear();
     while (reader.Read())
      {
       listBox1.Items.Add(
       reader["St_ID"] + " : " +
       reader["Gvari"] + " : " +
       reader["Saxeli"] + " : " +
       reader["sqesi"] + " : " +
       reader["dab_celi"] + " : " +
       reader["Jg_ID"] + " : " +
       reader["telefoni"]);
      }
     reader.Close();
    }

private void button2_Click(object sender, EventArgs e)
    {  // Insert ღილაკის მეთოდი -----
      int Raod;
      try
       {con.Open();
        cmd.CommandText="insert into Student "+
         "(St_ID,Gvari,Saxeli,sqesi,dab_celi,Jg_ID,telefoni) "+
        " values (" + textBox1.Text +",'"+
              textBox2.Text +"', '" +textBox3.Text + "', '"+
             textBox4.Text +"', " + textBox5.Text + ", "+
             textBox6.Text +", '" + textBox7.Text + "')";
         MessageBox.Show(cmd.CommandText);

         Raod=cmd.ExecuteNonQuery();
         if(Raod >0)
         MessageBox.Show("ერთი სტრიქონი ჩაიწერა !");
```

```
        }
       catch(Exception ex)
        {
         MessageBox.Show(ex.Message);
     MessageBox.Show("შეიტანეთ გვარი და სხვა მონაცემები...");
        }
       con.Close();
       Datvaliereba();
   }
  }
}
```

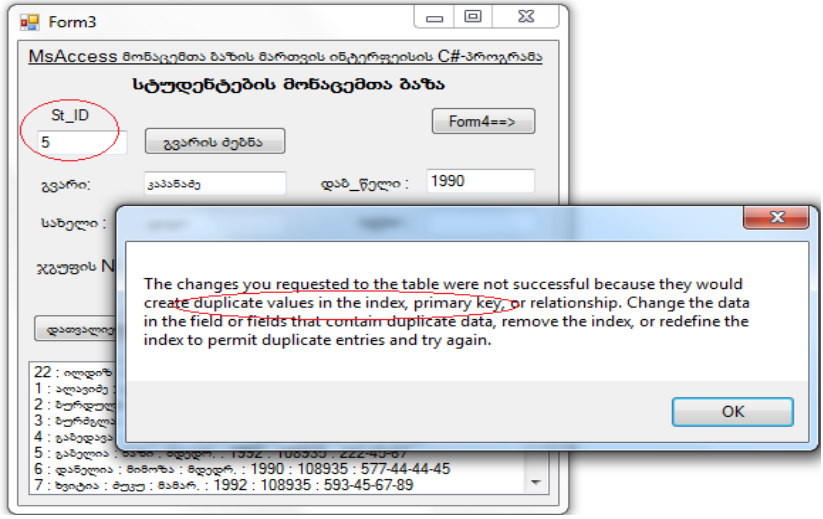
კოდის დასაწყისში Form3() კლასსა და Form3_Load მეთოდში გადმოტანილია ის ძირითადი კონსტრუქციები, რომლებიც საერთოა ფორმაზე მოთავსებული ელემენტებისათვის. ესაა con, cmd, stNummer ობიექტების შექმნა `OleDbConnection()`, `OleDbCommand()` და `ArrayList()` კლასების საფუძველზე. აგრეთვე, ბაზასთან მიერთება მისი პროვაიდერისა და კატალოგის გზის მითითებით. SQL მოთხოვნებისა და კავშირის მაჩვენებლებისთვის განისაზღვრა OleDbDataReader-ის reader.

პირველი რეალიზებული მეთოდი, რომლის ლისტინგიც ზემოთ განვიხილეთ, არის აგებული Form3-ის ღილაკის „დათვალიერება“ სტუდენტთა ცხრილისათვის. მისი გამოყენება შესაძლებელია მრავალჯერადად, ბაზის მონაცემთა ცვლილებების მონიტორინგის მიზნით.

აქ try…catch ბლოკში მოთავსებულია ბაზის გახსნა-დახურვის, select მოთხოვნის სტრიქონი, შეცდომების „დაჭერის“ და შეტყობინების გამოცემის მექანიზმი და ბოლოს ამორჩეული სტრიქონების გამოტანის მეთოდი `Ekranze_gamotana()`. შედეგები აისახება listBox1-ში.

მეორე რელიზებული მეთოდი ეკუთვნის Insert -ღილაკს ანუ შესაძლებელია Form3-ზე textBox-ებში ახალი სტუდენტის მონაცემების შეტანა და შემდეგ აღნიშნული ღილაკით (იხ. 4.3 ლისტინგში `button2_Click`) მისი მოთავსება Access ბაზაში.

თუ მომხმარებელმა შეცდომით შეიტანა რომელიმე მონაცემი, სისტემა პოულობს მას და გამოაქვს შესაბამისი შეტყობინება. მაგალითად, თუ შეტანილია სტუდენტის ინდექსი St_ID, რომელიც უკვე არსებობს, მაშინ მივიღებთ შეტყობინებას, რომელიც 4.18 ნახაზზეა ნაჩვენები. ბაზაში შეცდომიანი სტრიქონი არ ჩაიწერება.

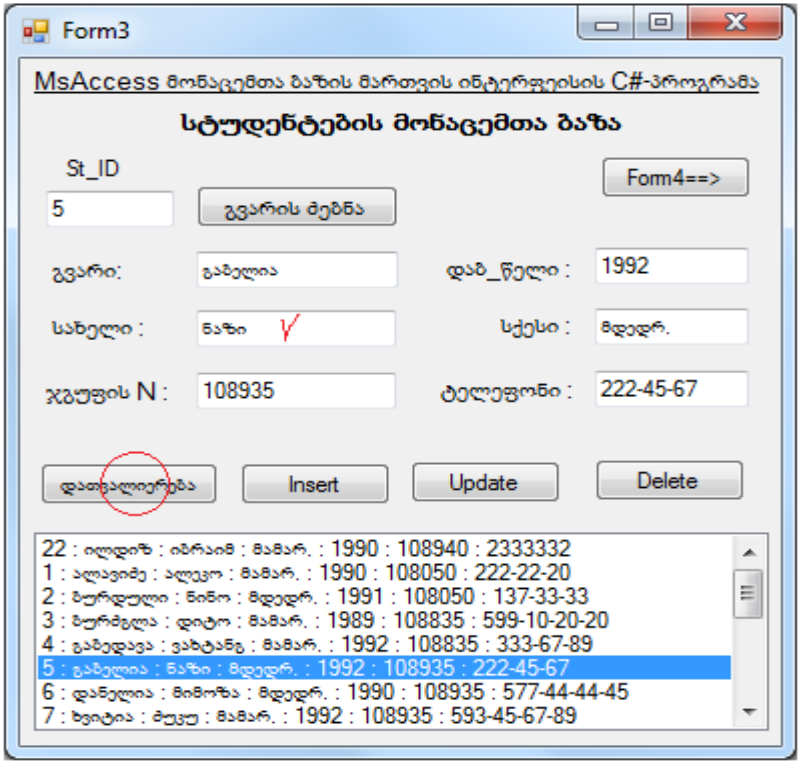


ნახ.4.18

ახლა განვიხილოთ ბაზაში ცვლილებების შეტანის (Update) და ჩანაწერების წაშლის (Delete) მეთოდების დაპროგრამების საკითხები.

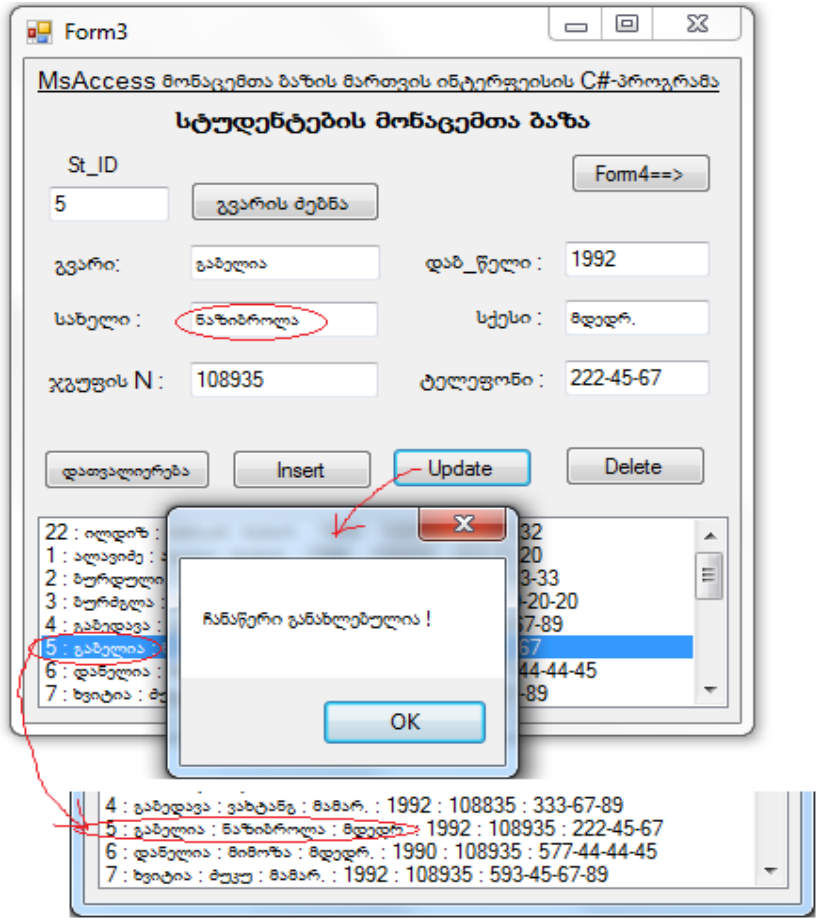
ამოცანა-4.4. Ms Access ბაზაში არსებული "სტუდენტების" ცხრილიდან Windows ფორმაზე ListBox-ში გამოვიტანოთ ჩანაწერები. ავირჩიოთ შესაცვლელი სტრიქონი, რომლის სვეტის მნიშვნელობები აისახება ფორმის შესაბამის textBox ველებში. აქ შევცვალოთ ამ ველების მნიშვნელობები. Update ღილაკით ცვლილებები უნდა მოთავსდეს ბაზაში. შედეგების შემოწმების სისწორე განხორციელდება სტრიქონების ხელახალი ასახვით ListBox-ში.

4.19 ნახაზზე მოცემულია ფორმის საწყისი მდგომარეობა. არჩეულია მე-5 სტრიქონი. textBox-ებში ჩაიწერა ამ სტრიქონის მონაცემები. საჭიროა შეიცვალოს **სახელი=”ნაზი”** ახლით.



ნახ.4.19

4.20 ნახაზზე ნაჩვენებია განახლების პროცესის ფრაგმენტები. ველის **სახელი**=”**ნაზიბროლა**” (შესაძლებელია სხვა ველების ცვლილებაც), შემდეგ Update ღილაკით გამოჩნდება შეტყობინება ”ჩანაწერი განახლებულია” (თუ ტრანზაქცია შესრულდა სწორად). თუ რამე შეცდომაა, გამოვა შესაბამისი შეტყობინება.



ნახ.4.20

4.4 ლისტინგის კოდი არის 4.3 ლისტინგის გაფართოება. აქ ორი მოვლენა პროგრამირდება. ერთი, listBox1-ში სტრიქონის არჩევა ცვლილების მიზნით, რომელსაც textBox-ებში გამოაქვს შესაბამისი მონაცემები. მეორე, Update ღილაკის დაჭერით შესრულებული პროცედურა (მონაცემთა ბაზაში შენახვა).

```
  //ლისტინგი_4.4 --- Update() ------
private void listBox1_SelectedIndexChanged(object sender,
                                                    EventArgs e)
{  // სტრიქონის არჩევა ----
 try
  {
    con.Open();
    cmd.CommandText = "select * from Student " +
        " where St_ID = " + stNummer[listBox1.SelectedIndex];

       reader = cmd.ExecuteReader();
       reader.Read();

       textBox1.Text="" + reader["St_ID"];
       textBox2.Text="" + reader["Gvari"];
       textBox3.Text="" + reader["Saxeli"];
       textBox4.Text="" + reader["sqesi"];
       textBox5.Text="" + reader["dab_celi"];
       textBox6.Text="" + reader["Jg_ID"];
       textBox7.Text="" + reader["telefoni"];

       reader.Close();
    }
  catch (Exception ex)
    {
      MessageBox.Show(ex.Message);
    }
  con.Close();
}
private void button3_Click(object sender, EventArgs e)
  {   // update -------
    int Raod;
    try
     {
       con.Open();
       cmd.CommandText = "update Student set " +
             "St_ID = " + textBox1.Text + ", "+
              "Gvari = '" + textBox2.Text + "', " +
              "Saxeli ='" + textBox3.Text + "', " +
              "sqesi = '" + textBox4.Text + "', " +
              "dab_celi = " + textBox5.Text + ", " +
              "Jg_ID = " + textBox6.Text + ", " +
```

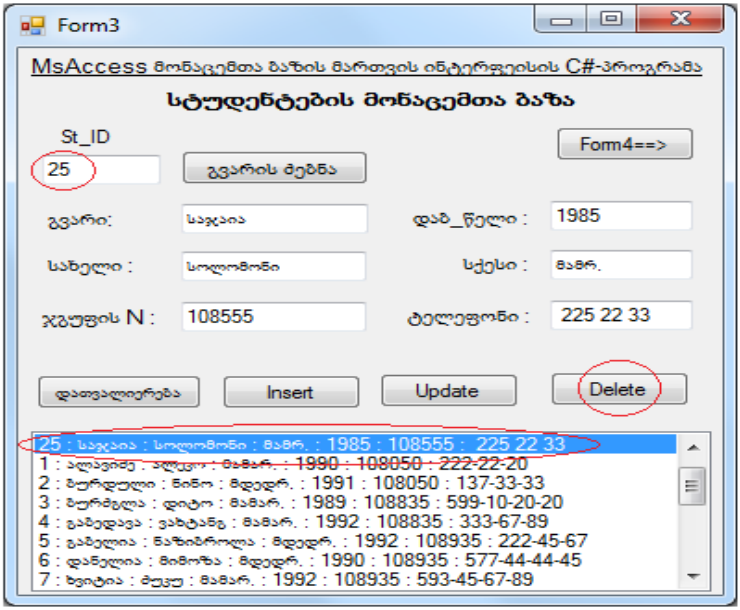
```
            "telefoni = '" + textBox7.Text + "' "+
      " where St_ID =" + stNummer[listBox1.SelectedIndex];
        MessageBox.Show(cmd.CommandText);
          Raod = cmd.ExecuteNonQuery();
          if (Raod > 0)
            MessageBox.Show("ჩანაწერი განახლებულია !");
      }
    catch(Exception ex)
      {
         MessageBox.Show(ex.Message);
         MessageBox.Show("შეასწორეთ ერთი ჩანაწერი " +
                   "აირჩიეთ რომელიმე გვარი," +
                   " ცალსახა სტუდ_ნომერი და " +
                   " სხვა მონაცემები !!!");
      }
    con.Close();
    Datvaliereba();
  }  // end Update
```

ამოცანა-4.5. განვიხილოთ Ms Access ბაზის ცხრილში "სტუდენტები" ჩანაწერის წაშლის პროცედურა, რომელიც Delete ღილაკზე იქნება მიბმული.

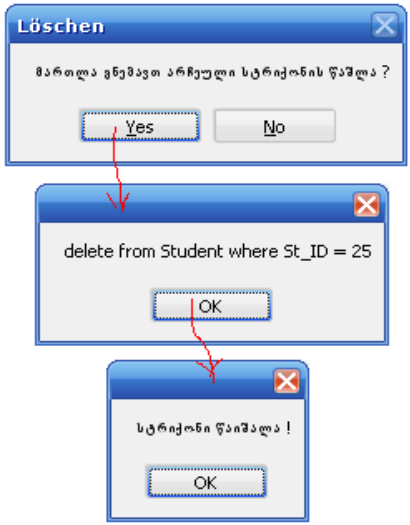
4.21 ნახაზზე ნაჩვენებია ფორმაზე მოთავსებული ცხრილის "სტუდენტები" ჩანაწერები, რომელთაგან, ერთ-ერთი, კერძოდ, 25-ე ჩანაწერი არჩეულია წასაშლელად.

Delete ღილაკის ამოქმედებით მიმდევრობით გამოჩნდება 4.22 ნახაზზე ნაჩვენები შეტყობინებები. თუ ყველაფერი ნორმალურადაა, ბაზიდან მოხდება 25-ე ჩანაწერის ამოშლა.

წაშლის პროგრამის ტექსტი მოცემულია 4.5 ლისტინგში. აქაც try…catch ბლოკის cmd.CommandText სტრიქონში ჩაწერილია წაშლის SQL ბრძანება, რომელიც შემდეგ `ExecuteNonQuery()` მეთოდით გადაეცემა შესასრულებლად. შედეგის ნახვა ხორციელდება პროგრამის ბოლოს, Datvaliereba() მეთოდით.



ნახ.4.21



ნახ.4.22

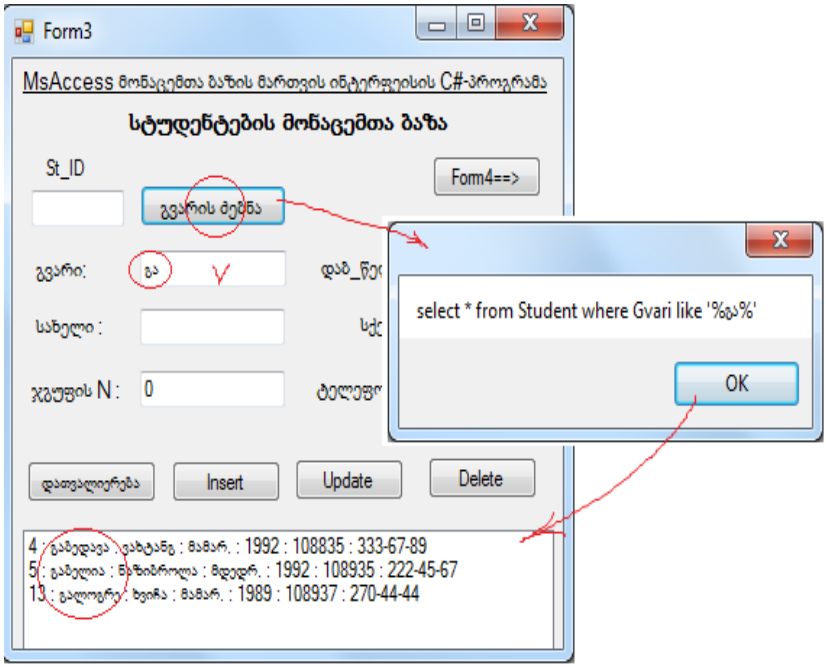
```
// ლისტინგი_4.5 ---  Delete -------
private void button4_Click(object sender, EventArgs e)
 { int Raod;
   if (textBox1.Text == "")
    {
      MessageBox.Show("აირჩიეთ ერთი სტრიქონი !");
      return;
    }
   if (MessageBox.Show("მართლა გნებავთ არჩეული" +
                  " სტრიქონის წაშლა ?", "Löschen",
            MessageBoxButtons.YesNo) == DialogResult.No)
     return;
   try
   {
    con.Open();
    cmd.CommandText = "delete from Student " +
          "where St_ID = " + stNummer[listBox1.SelectedIndex];
      MessageBox.Show(cmd.CommandText);
    Raod = cmd.ExecuteNonQuery();
    if (Raod > 0)
       MessageBox.Show("სტრიქონი წაიშალა !");
   }
  catch (Exception ex)
   { MessageBox.Show(ex.Message); }
  con.Close();
  Datvaliereba();
}  //end Delete -----------
```

ამოცანა-4.6. ავაგოთ პროგრამა, რომელიც გვარის ველში შეტანილი ერთი, ორი ან მეტი სიმბოლოთი იპოვის მრავალჩანაწერიანი ბაზის შესაბამის სტრიქონებს და გამოიტანს მათ listBox1-ში (ნახ.4.23).

ეს კოდი მოცემულია 4.6 ლისტინგში.

```
  // ლისტინგი_4.6 ---- გვარით ძებნა -----
private void button5_Click(object sender, EventArgs e)
 {  try
      {
        con.Open();
        cmd.CommandText = "select * from Student where" +
```

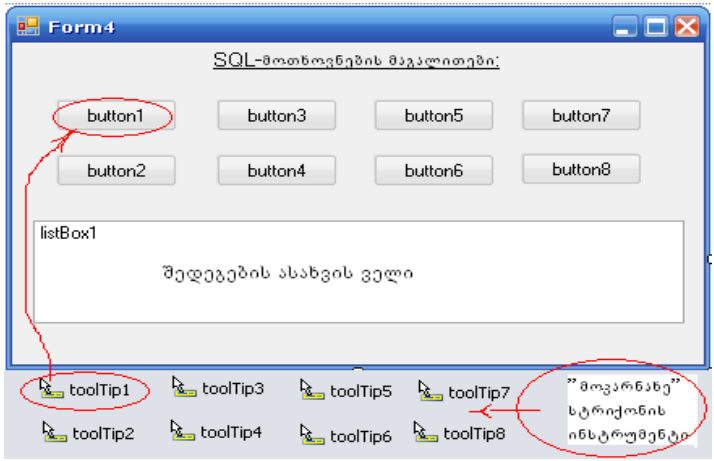
```
                " Gvari like '%" + textBox2.Text + "%'";
          MessageBox.Show(cmd.CommandText);
          Ekranze_gamotana();
        }
      catch (Exception ex)
        {
          MessageBox.Show(ex.Message);
        }
      con.Close();
  }
```



ნახ.4.23

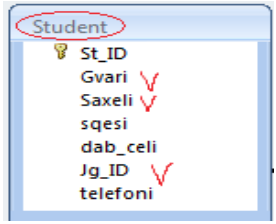
4.5. SQL მოთხოვნების დაპროგრამების საილუსტრაციო მაგალითები C#-ში MsAccess ბაზისთვის

ამოცანა-4.7. ავაგოთ C# კოდი - მომხმარებლის ინტერფეისი MsAccess მონაცემთა ბაზის რამდენიმე ცხრილთან ერთდროულად სამუშაოდ. გამოვიყენოთ DB_AccessUni ბაზის Table-ები: Student, Group, Lector და Lect_Jgufi (ნახ.3.1-3.3). საჭიროა მოთხოვნების წინასწარ დაპროექტება და მისი SQL ფორმატით წარმოდგენა. შემდეგ ამ სტრუქტურირებული მოთხოვნის ჩასმა C# კოდში (მიმაგრება button_Click... ღილაკებზე, ნახ.4.24)



ნახ.4.24

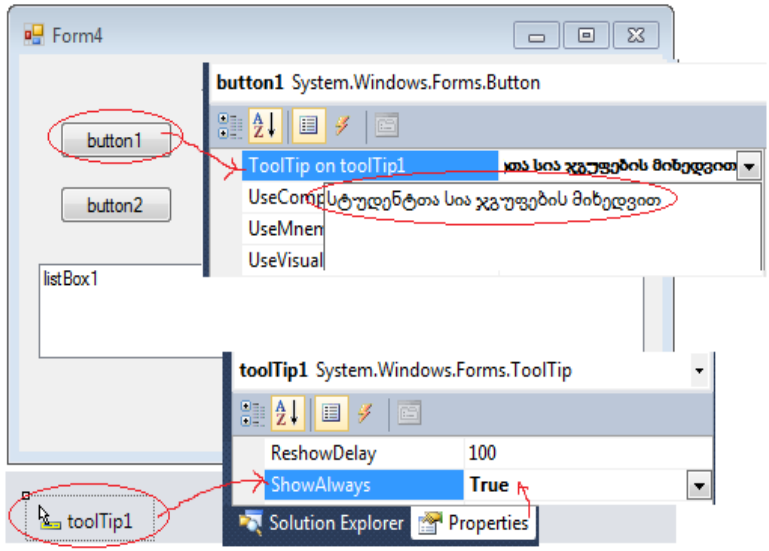
მოთხოვნა-1. „ეკრანზე გამოვიტანოთ სტუდენტთა სია: გვარი, სახელი, ჯგუფის_ნომერი, მოწესრიგებული ჯგუფის ნომრით და გვარით" (ნახ.4.25) .



ნახ.4.25

სასურველია მოთხოვნის ღილაკების „გასემანტიკურება“ (მაუსის კურსორის მიტანისას ღილაკზე გამოჩნდეს „მოკარნახე ტექსტი (hint მინიშნება), თუ რას აკეთებს ეს ღილაკი“).

ამისათვის ToolBox პანელიდან ფორმაზე გადმოვიტანოთ ToolTip1 არავიზუალური ელემენტი, რომელიც მოთავსდება ფორმის ქვემოთ (ნახ.4.26). მოვნიშნოთ იგი და Properties-ში დავაყენოთ თვისება ShowAllways = true.

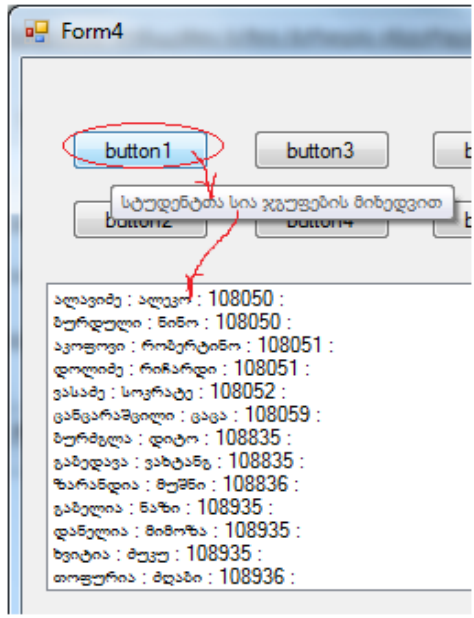


ნახ.4.26

შემდეგ მოვნიშნოთ ღილაკი button1 და Properties-ის თვისებაში ToolTip on toolTip1 ჩავწეროთ „მოკარნახე“ ტექსტი. ავამუშაოთ პროგრამა, მივიტანოთ მაუსის კურსორი button1 ღილაკთან. გამოჩნდება ტექსტი „სტუდენტთა სია ჯგუფების მიხედვით“. თუ არ გვინდა ეს ინფორმაცია, გადავალთ სხვაზე. თუ ავამოქმედებთ button1-ს, მივიღებთ შედეგს (ნახ.4.27).

button1 ღილაკის პროგრამის კოდის ფრაგმენტი, რომელშიც ასახულია SQL ტიპის მოთხოვნა, მოცემულია 4.7 ლისტინგში.

```
// ლისტინგი_4.7 --- button1 ----
private void button1_Click(object sender, EventArgs e)
 {
   Amorcheva("select * from Student order by Jg_ID, Gvari",
                                    "Gvari", "Saxeli", "Jg_ID");
   MessageBox.Show("select * from Student order by Jg_ID,
                                  'Gvari', 'Saxeli', 'Jg_ID' ");
 }
```



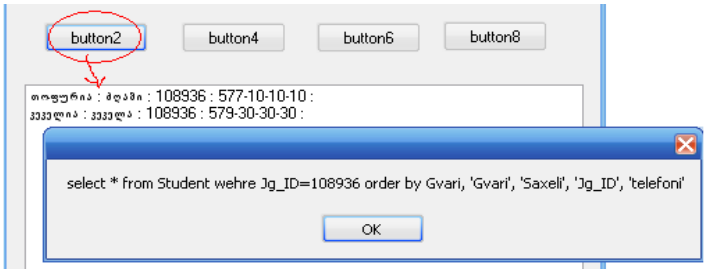
ნახ.4.27

მოთხოვნა-2. „ვიპოვოთ ის სტუდენტები, რომლებიც სწავლობენ 108936 ჯგუფში. გამოვიტანოთ მათი გვარები, სახელები, ჯგუფის ნომრები”.

```
// ლისტინგი_4.7 --- button2 ----
private void button2_Click(object sender, EventArgs e)
{
Amorcheva("select * from Student where Student.Jg_ID=108936 "+
```

```
  "order by Gvari", "Gvari", "Saxeli", "Jg_ID", "telefoni");
MessageBox.Show("select * from Student wehre Jg_ID=108936 "+
  "order by Gvari, 'Gvari', 'Saxeli', 'Jg_ID', 'telefoni' ");
}
```

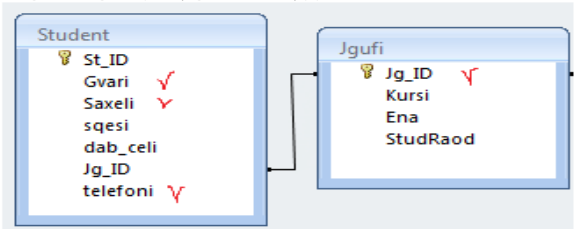
შედეგი მოცემულია 4.28 ნახაზზე.



ნახ.4.28

მოთხოვნა-3. „ვიპოვოთ იმ სტუდენტთა გვარები, სახელები, ჯგუფის ნომრები და ტელეფონები, რომლებიც სწავლობენ ინგლისურენოვან ჯგუფში”.

ამ შემთხვევაში საჭიროა ბაზიდან ორი ცხრილის გამოყენება: Student და Jgufi. 4.29 ნახაზიდან ჩანს, რომ ორი ეს ცხრილები კავშირშია ერთმანეთთან ატრიბუტით Jgufi.Jg_ID (პირველადი გასაღები ანუ უნიკალური ინდექსი) და Student.Jg_ID (მეორეული გასაღები, ანუ არაუნიკალური ინდექსი).

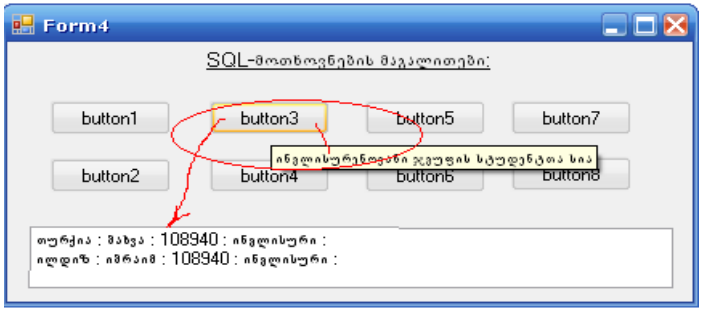


ნახ.4.29

მათი გათვალისწინებით SQL მოთხოვნას ექნება 4.8 ლისტინგზე მოცემული სახე.

```
// ლისტინგი_4.8 --- button3 ----
```

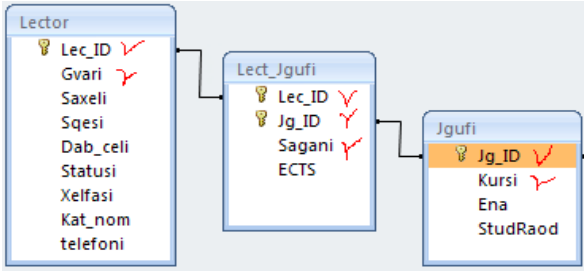
```
private void button3_Click(object sender, EventArgs e)
{
   Amorcheva("select * from Student,Jgufi " +
     "where Student.Jg_ID=Jgufi.Jg_ID and Ena='ინგლისური' " +
                  "order by Jgufi.Jg_ID, Gvari", "Gvari",
                           "Saxeli", "Student.Jg_ID", "Ena");
  MessageBox.Show("select * from Student,Jgufi "+
    "where Student.Jg_ID=Jgufi.Jg_ID and Ena='ინგლისური' " +
               "order by Gvari, 'Gvari', 'Saxeli',
                                 'Student.Jg_ID', 'Ena' ");
}
```



ნახ.4.30

მოთხოვნა-4. „რომელი კურსის რომელ ჯგუფებს და რა საგნებს ასწავლის ლექტორი გაბედავა ” ?

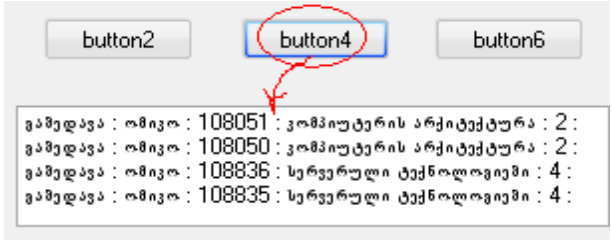
ბაზიდან უნდა გამოვიყენოთ სამი ცხრილი: Lector, Jgufi და Lect_Jgufi, რათა კონკრეტული ლექტორისთვის ვიპოვოთ მისი საგნები, ჯგუფები და კურსები (ნახ.4.31).



ნახ.4.31

კოდის ტექსტი მოცემულია 4.9 ლისტინგში, ხოლო შედეგები 4.32 ნახაზზე.

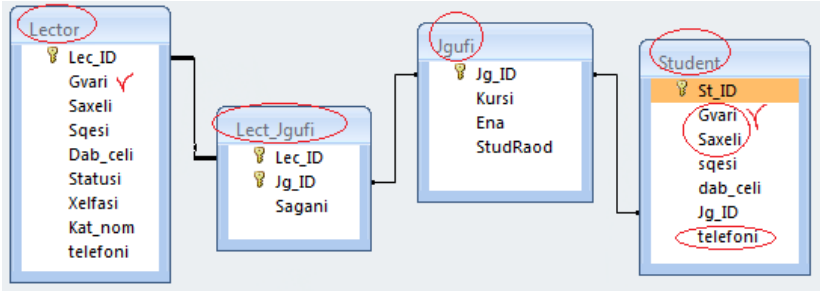
```
// ლისტინგი_4.9 --- button4 ----
private void button4_Click(object sender, EventArgs e)
{
  Amorcheva("SELECT * FROM Lector,Jgufi,Lect_Jgufi " +
            "where Jgufi.Jg_ID = Lect_Jgufi.Jg_ID and "+
               "Lector.Lec_ID = Lect_Jgufi.Lec_ID and "+
                             "Lector.Gvari='გაბედავა' " +
            "order by Lect_Jgufi.Sagani ", "Gvari", "Saxeli",
                    "Jgufi.Jg_ID", "Sagani","kursi");

  MessageBox.Show("SELECT * FROM Lector,Jgufi,Lect_Jgufi " +
                  "where Jgufi.Jg_ID = Lect_Jgufi.Jg_ID and "+
                      "Lector.Lec_ID = Lect_Jgufi.Lec_ID and
                         Lector.Gvari='გაბედავა' " +
                   "order by Lect_Jgufi.Sagani , 'Gvari',
                 'Saxeli', 'Jgufi.Jg_ID', 'Sagani', 'kursi'");
}
```



ნახ.4.32

მოთხოვნა-5. „რომელ სტუდენტებს ასწავლის ლექტორი გაბედავა ? საჭიროა გვარი, სახელი და ჯგუფის ნომერი”.

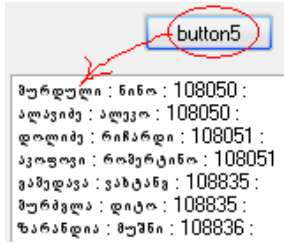
მონაცემთა ბაზიდან ამჯერად დაგვჭირდება ოთხივე ცხრილი (ნახ.4.33). კონკრეტული ლექტორიდან (მარცხენა ნაპირა ცხრილი) უნდა მივიდეთ კონკრეტულ სტუდენტამდე (მარჯვენა ნაპირა ცხრილი). შესაბამისი SQL მოთხოვნის ტექსტი მოცემულია 4.10 ლისტინგში.



ნახ.4.33

```
// ლისტინგი_4.10 --- button5 ----
private void button5_Click(object sender, EventArgs e)
{
 Amorcheva("SELECT * FROM Lector,Jgufi,Lect_Jgufi,Student " +
            "where Jgufi.Jg_ID = Lect_Jgufi.Jg_ID and "+
            "Lector.Lec_ID = Lect_Jgufi.Lec_ID and " +
    "Student.Jg_ID=Jgufi.Jg_ID and Lector.Gvari='გაბედავა' " +
             "order by Jgufi.Jg_ID ", "Student.Gvari",
                        "Student.Saxeli", "Student.Jg_ID");
  MessageBox.Show("SELECT * FROM
                        Lector,Jgufi,Lect_Jgufi,Student " +
                  "where Jgufi.Jg_ID = Lect_Jgufi.Jg_ID and
                    Lector.Lec_ID = Lect_Jgufi.Lec_ID and " +
    "Student.Jg_ID=Jgufi.Jg_ID and Lector.Gvari='გაბედავა' " +
                  "order by Jgufi.Jg_ID, 'Student.Gvari',
                       'Student.Saxeli', 'Student.Jg_ID' ");
}
```

შედეგი მოცემულია 4.34 ნახაზზე.

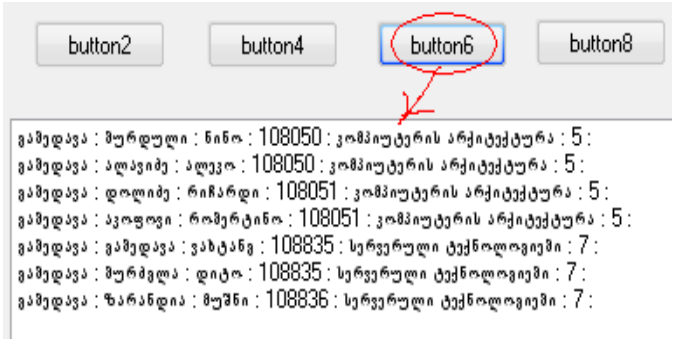


ნახ.4.34

მოთხოვნა_6: „ლექტორ გაბედავას რომელი ჯგუფის რომელ სტუდენტებთან აქვს ლექციები, რომელ საგნებში და რამდენ კრედიტიანია ეს საგნები ” ?

```
// ლისტინგი_4.11 --- button6 ----
private void button6_Click(object sender, EventArgs e)
{
Amorcheva("SELECT * FROM Lector,Lect_Jgufi, Jgufi,Student " +
          "where Lector.Lec_ID=Lect_Jgufi.Lec_ID and "+
                      " Jgufi.Jg_ID = Lect_Jgufi.Jg_ID and "+
    "Student.Jg_ID=Jgufi.Jg_ID and Lector.Gvari='გაბედავა' " +
          "order by Jgufi.Jg_ID ", "Lector.Gvari",
                    "Student.Gvari", "Student.Saxeli", +
                    "Jgufi.Jg_ID", "Sagani", "ECTS");
 MessageBox.Show("SELECT * FROM Lector,Lect_Jgufi, Jgufi, "+
                                                "Student " +
                "where Lector.Lec_ID=Lect_Jgufi.Lec_ID and "+
                      " Jgufi.Jg_ID = Lect_Jgufi.Jg_ID and "+
                       "Student.Jg_ID=Jgufi.Jg_ID and "+
                       "Lector.Gvari='გაბედავა' " +
                "order by Jgufi.Jg_ID, 'Lector.Gvari', "+
                          'Student.Gvari', 'Student.Saxeli',
                         'Jgufi.Jg_ID', 'Sagani', 'ECTS' ");
}
```

შედეგები გამოტანილია 4.35 ნახაზზე.



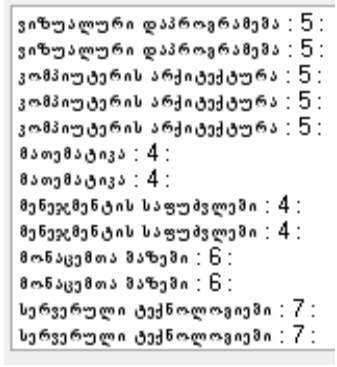
ნახ.4.35

მოთხოვნა-7. „რა საგნები ისწავლება და რამდენ კრედიტიანია ” ?

ეს ინფორმაცია მისაწვდომია ერთი ცხრილიდან Lect_Jgufi. თუ მოთხოვნა ასე დაიწერება:

```
Amorcheva("select Sagani, ECTS from Lect_Jgufi order by
                                    Sagani", "Sagani", "ECTS");
```

მაშინ მივიღებთ შემდეგ შედეგს (ნახ.4.36), რაც არაკორექტულია განმეორებადი სტრიქონების გამო:

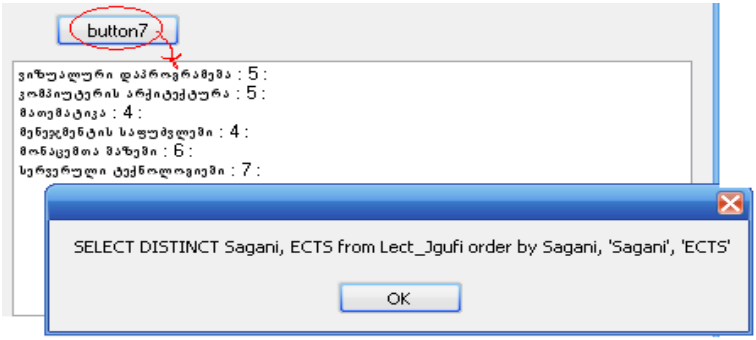


ნახ.4.36

საჭიროა SELECT DISTINCT… კონსტრუქციის გამოყენება, რაც გამორიცხავს განმეორებად სტრიქონებს რელაციურ ცხრილებში. ამგვარად, შესაძლებელია სტრიქონთა „სიმრავლის” მიღება. სწორი ტექსტი მოცემულია 4.12 ლისტინგში.

```
// ლისტინგი_4.12 --- button7 ----
private void button7_Click(object sender, EventArgs e)
{
  Amorcheva("select distinct Sagani, ECTS from Lect_Jgufi “+
            “ order by Sagani", "Sagani", "ECTS");
  MessageBox.Show("SELECT DISTINCT Sagani, ECTS “+
       from Lect_Jgufi order by Sagani, 'Sagani', 'ECTS' ");
 }
```

4.37 ნახაზზე ასახულია სწორი შედეგები.



ნახ.4.37

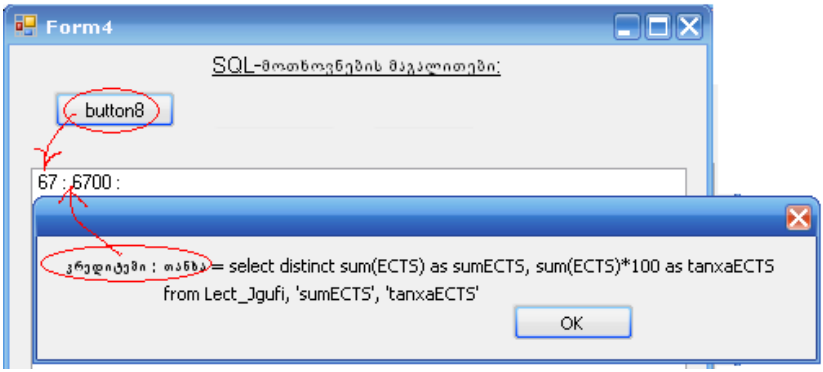
მოთხოვნა-8. „რამდენი კრედიტი შეიძინეს სტუდენტებმა დამატებით სემესტრში და რას უდრის ჯამური შემოსავალი თანხაში ” ?

Lect_Jgufi ცხრილში მოთავსებულია ოპერატიული ინფორმაცია მიმდინარე სემესტრის საგნების შესახებ ანუ აქედან უნდა მოხდეს ECTS-ატრიბუტის მნიშვნელობათა შეჯამება და მისი გამრალება ერთი კრედიტის ღირებულებაზე. სიმარტივისთვის დავუშვათ, რომ ის 100 ლარია.

4.13 ლისტინგში მოცემულია კოდის ტექსტი შესაბამისი SQL მოთხოვნით.

```
// ლისტინგი_4.13 --- button7 ----
private void button8_Click(object sender, EventArgs e)
{
  Amorcheva("select sum(ECTS) as sumECTS, “+
                  “sum(ECTS)*100 as tanxaECTS “+
            “from Lect_Jgufi ", "sumECTS", "tanxaECTS");
  MessageBox.Show(" კრედიტები : თანხა = select “+
        “sum(ECTS) as sumECTS, sum(ECTS)*100 as tanxaECTS "+
           "from Lect_Jgufi, 'sumECTS', 'tanxaECTS' ");
}
```

შედეგი მოცემულია 4.38 ნახაზზე.



ნახ.4.38

განხილული პროგრამის ყველა მოთხოვნის დასაწყისი და საერთო ნაწილი მოცემულია 4.14 ლისტინგში.

```
// ლისტინგი_4.14 --- Syetem+Contact with DBS+ try…catch ----
using System;
using System.Drawing;
using System.Windows.Forms;
using System.Data.OleDb;
namespace WinADO
{
 public partial class Form4 : Form
  {
   public Form4() { InitializeComponent(); }
   private void Amorcheva(string sqlbefehl,
                                    params string[] velebi)
   {
     OleDbConnection con = new OleDbConnection();
     OleDbCommand cmd = new OleDbCommand();
     OleDbDataReader reader;
     int i;
     string striqoni;
con.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;" +
        "Data Source=C:\\C#2010\\WinADO\\DB_AccessUni.accdb";
     cmd.Connection = con;
     cmd.CommandText = sqlbefehl;
     try
      {// აქ ხდება SQL მოთხოვნის წაკითხვა, ანალიზი ატრიბუტებით
       con.Open();
```

```
     reader = cmd.ExecuteReader();
     listBox1.Items.Clear();
     while (reader.Read())
      {
        striqoni = "";
        for (i = 0; i < velebi.Length; i++)
           striqoni += reader[velebi[i]] + " : ";

        listBox1.Items.Add(striqoni); // ეკრანზე გამოტანა ---
      }
     reader.Close();
     con.Close();
    }
  catch (Exception ex)
    {    // შეცდომის არსებობისას გამოსცემს შეტყობინებას --------
      MessageBox.Show(ex.Message);
    }
}

// აქ უერთდება button_Click  მეთოდები -----  // . . .
```

V თავი. Web აპლიკაციების აგება ASP.NET ტექნოლოგიით

5.1. შესავალი ASP.NET სისტემაში

ASP.NET (Active Server Pages – აქტიური სერვერული გვერდები) არის NET პლატფორმის ნაწილი და ტექნოლოგია, რომელიც დინამიკურად ქმნის დოკუმენტებს Web სერვერზე, როცა ისინი მოითხოვება HTTP-ს საშუალებით.

ASP.NET ტექნოლოგია ანალოგიურია PHP, ColdFusion და სხვა ტექნოლოგიების, მაგრამ მათ შორის მნიშვნელოვანი განსხვავებაცაა. ASP.NET, როგორც მისი დასახელება გვიჩვენებს, შეიქმნა სპეციალურად NET პლატფორმასთან სრული ინტეგრაციის მიზნით, რომლის ნაწილი ითვალისწინებს C# ენის მხარდაჭერას.

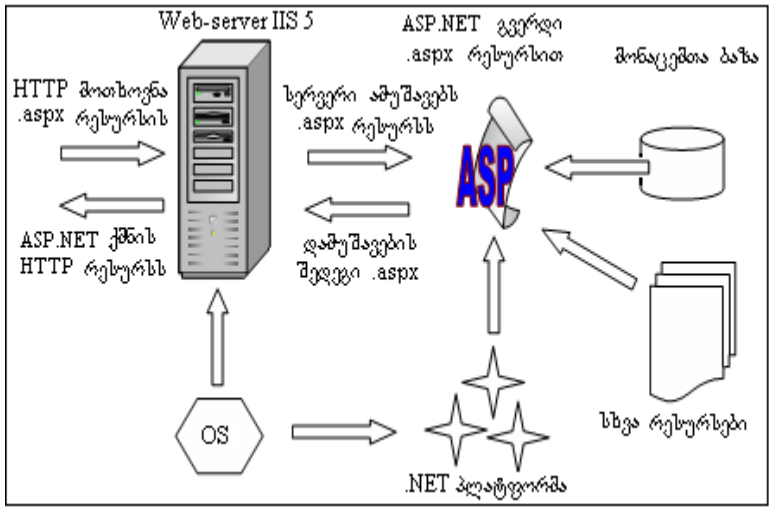
როგორც ცნობილია, Web გვერდების დასაპროგრამებლად გამოიყენება ისეთი სცენარების ენები, როგორიცაა VBScript ან JScript. ეს სკრიპტული ენები მუშაობდა, მაგრამ ხშირად გარკვეულ პრობლემებს უქმნიდა დაპროგრამების "ნამდვილი" ენების პროგრამისტებს სხვადასხვა ადმინისტრაციულ დავალებათა შესრულებისას, რაც საბოლოო ჯამში აისახება სისტემის მწარმოებლურობის დაქვეითებაში.

მაღალგანვითარებული ენების გამოყენების შემთხვევაში შესაძლებელია მუშაობის პროცესის უზრუნველყოფა სრული სერვერული ობიექტური მოდელით. ASP.NET ახორციელებს მიმართვას გვერდის ყველა მმართველ ელემენტთან, როგორც ზოგადად გარემოს ობიექტებთან. სერვერის მხარესაც კი ხორციელდება წვდომა .NET-ის ყველა საჭირო კლასთან.

გვერდების მართვის ელემენტები ფუნქციონალურია და ფაქტობრივად, შესაძლებელია ყველაფრის გაკეთება, რასაც Windows-ის ფორმის კლასებთან ვაკეთებდით, რაც უფრო მოქნილს ხდის სისტემას. ამის გამო, ASP.NET-ის გვერდებს, რომლებიც ქმნის HTML შედგენილობას, ხშირად უწოდებენ Web ფორმებს.

ASP.NET გამოიყენებს ინტერნეტის ინფორმაციულ სერვერს (IIS – Internet Information Server) HTTP მოთხოვნებზე

საპასუხო შინაარსის მისაწოდებლად. ASP.NET გვერდები მოთავსებულია ფაილებში .aspx გაფართოებით. მისი საბაზო არქიტექტურა მოცემულია 5.1 ნახაზზე.



ნახ.5.1. საბაზო არქიტექტურა

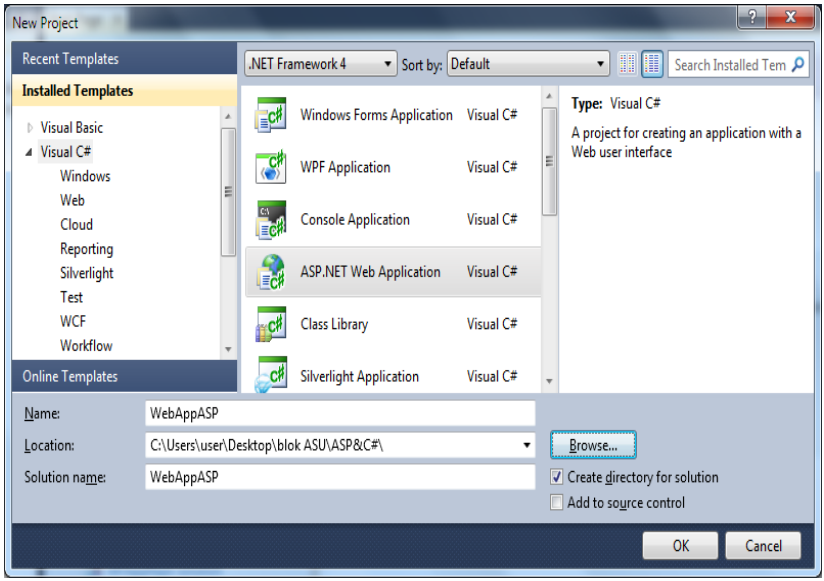
ASP.NET-ის დამუშავებისას მისაწვდომია .NET-ის ყველა კლასი, სპეციალური კომპონენტები, შექმნილი C# ან სხვა ენებზე, მონაცემთა ბაზები და ა.შ. ფაქტობრივად, სახეზეა ყველა ის შესაძლებლობა, რომელსაც იყენებს C# დანართის აგებისას. ე.ი. C#-ის გამოყენება ASP.NET-ში ეფექტურს ხდის დანართის შესრულებას.

Web აპლიკაციების აგება ASP.NET-ში ხდება კლასების კონსტრუირებით, რომლებიც ურთიერთმოქმედებს სხვა კლასებთან. ზოგი კლასი იწარმოება პლატფორმის საბაზო კლასებიდან, ზოგს შეუძლია ინტერფეისების რეალიზება ამ პლატფორმაში, ზოგიც ურთიერთქმედებს პლატფორმის საბაზო კლასებთან მათი მეთოდების გამოძახების გზით.

5.2. ASP.NET აპლიკაციის შექმნის ეტაპები

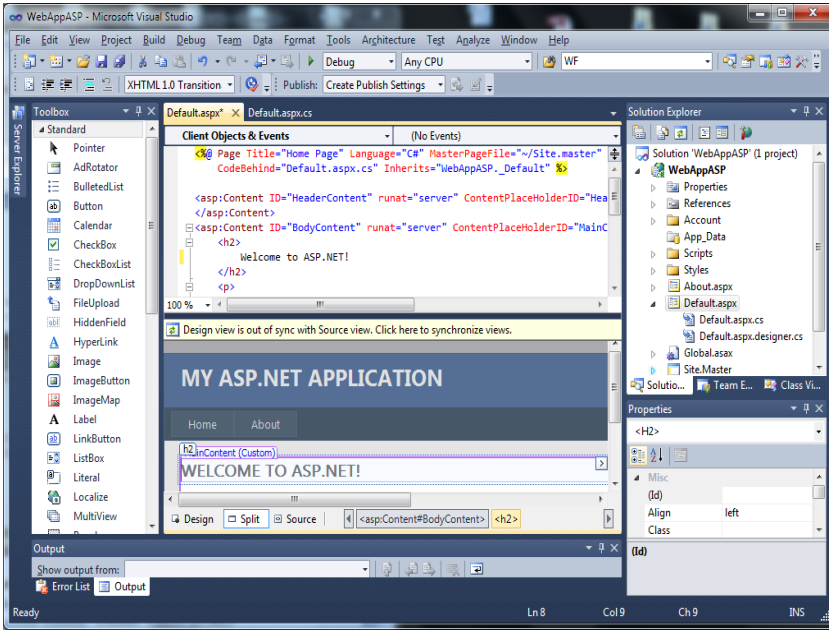
ASP.NET აპლიკაციის შესაქმნელად საჭიროა შემდეგი საფეხურების შესრულება:

- პროგრამების პანელიდან ავირჩიოთ:
  Start->Programs->Microsoft Visual Studio .NET 2010
- პროგრამის გაშვების შემდეგ აირჩიეთ მენიუს პუნქტი:
  File -> New -> Project.
- გაიხსნება ახალი პროექტების ტიპის არჩევის ფანჯარა (ნახ.5.2). ავირჩიოთ Visual C# Project და ASP.NET Web Application. Location ველში მივუთითოთ პროექტის შესანახი ადგილი. შევიტანოთ Web აპლიკაციის სახელი, მაგალითად, WebAppASP და OK.



ნახ.5.2

- Visual Studio შექმნის აპლიკაციას და გახსნის Microsoft Visual C#.NET გარემოში (ნახ.5.3).



ნახ.5.3

შექმნილი პროექტი შეიცავს რამდენიმე ფაილს. მაგალითად:

- Global.asax ფაილი ემსახურება აპლიკაციის დონის მოვლენების დამუშავებას, როგორიცაა შეცდომების დაჭერა, ახალი სესიის შექმნა, სესიის დასრულება და სხვა;
- Web.config არის XML ფაილი, რომელიც შეიცავს აპლიკაციის კონფიგურაციის მონაცემებს: სესიის პარამეტრები, მონაცემთა ბაზასთან კავშირის პარამეტრები, მომხმარებლების ავტორიზაციისა და აუტენთიფიკაციის კონფიგურაციის პარამეტრები;
- Default.aspx და Default.aspx.cs ქმნის ერთ ვებ-გვერდს. Default.aspx ფაილი შეიცავს ვიზუალურ ელემენტებს. მისი სტანდარტული საწყისი XML კოდის ლისტინგი ასეთია:

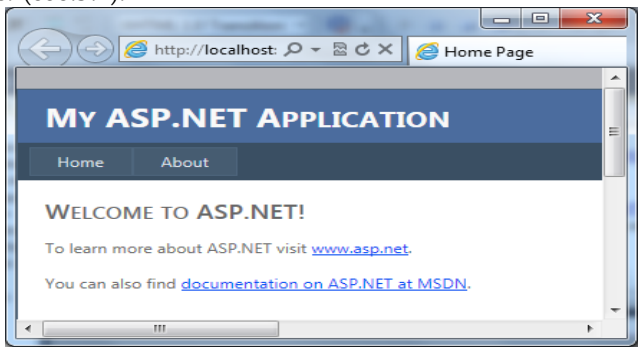
```
<!– ლისტინგი_5.1 ——— –>
<%@ Page Title="Home Page" Language="C#" MasterPageFile="~/Site.master"
AutoEventWireup="true"
```

```
CodeBehind="Default.aspx.cs" Inherits="WebAppASP._Default" %>
<asp:Content ID="HeaderContent" runat="server"
ContentPlaceHolderID="HeadContent">
</asp:Content>
<asp:Content ID="BodyContent" runat="server"
    ContentPlaceHolderID="MainContent">
   <h2> Welcome to ASP.NET! </h2>
   <p>
      To learn more about ASP.NET visit
     <a href="http://www.asp.net" title="ASP.NET
                    Website"> www.asp.net</a>.
   </p>
   <p>
      You can also find <a href="http://go.microsoft.com
                              /fwlink/?LinkID=152368&amp;clcid=0x409"
     title="MSDN ASP.NET Docs">documentation on ASP.NET at MSDN</a>.
   </p>
</asp:Content>
```

- Default.aspx.cs – შეიცავს Web ფორმის კლასის მოვლენების დამუშავების მეთოდებს და ბიზნესლოგიკას. საწყისი კოდი მოცემულია 5.2 ლისტინგში.

```
// ——— ლისტინგი_5.2 ————————
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
namespace WebAppASP
{
  public partial class _Default : System.Web.UI.Page
    {
     protected void Page_Load(object sender, EventArgs e)
       {
       }
    }
}
```

შექმნილი ვებ-გვერდის ნახვა შესაძლებელია Ctrl+F5 დაჭერით ან მენიუს პუნქტი Debug -> Start Without Debugging არჩევით (ნახ.5.4):

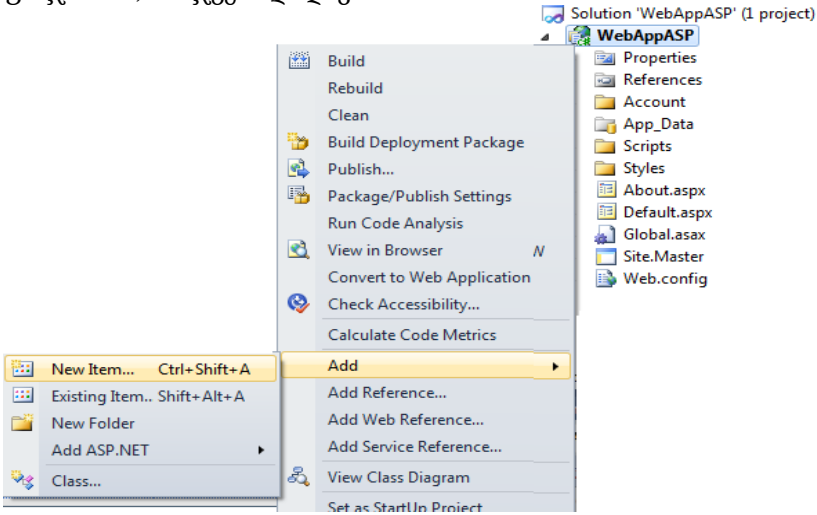


ნახ.5.3. შედეგი

5.3. პროექტში ახალი ვებ-გვერდის დამატება

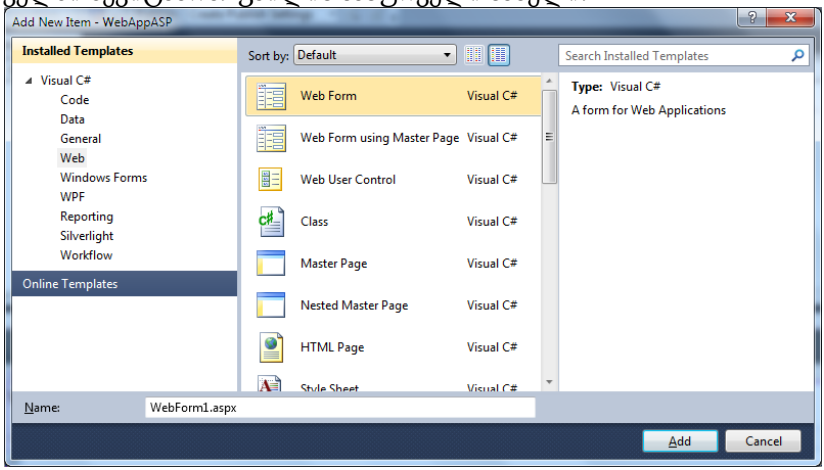
შექმნილ აპლიკაციას დავამატოთ ახალი ASPX ვებ-გვერდი.

ახალი ფაილის დამატება შესაძლებელია Solution Explorer ფანჯარაში, მარჯვენა ღილაკით: Add->Add New Item:



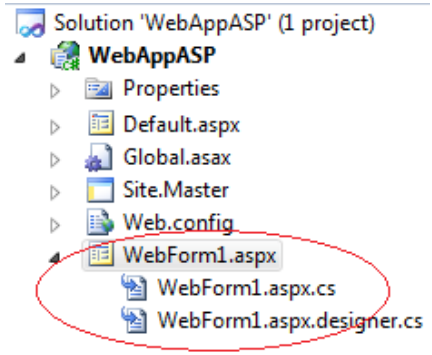
ნახ.5.4

გაიხსნება ახალი ფაილის არჩევის ფანჯარა (ნახ.5.5). ავირჩიოთ Web შაბლონების ფანჯარაში და Web Form. ხოლო Name ველში შევიტანოთ ფაილის სასურველი სახელი:



ნახ.5.5

Add ღილაკზე დაჭერით პროექტს დაემატება ახალი ფაილები: WebForm1.aspx და WebForm1.aspx.cs (ნახ.5.6). გვერდი ავტომატურად გაიხსნება დიზაინის რეჟიმში.

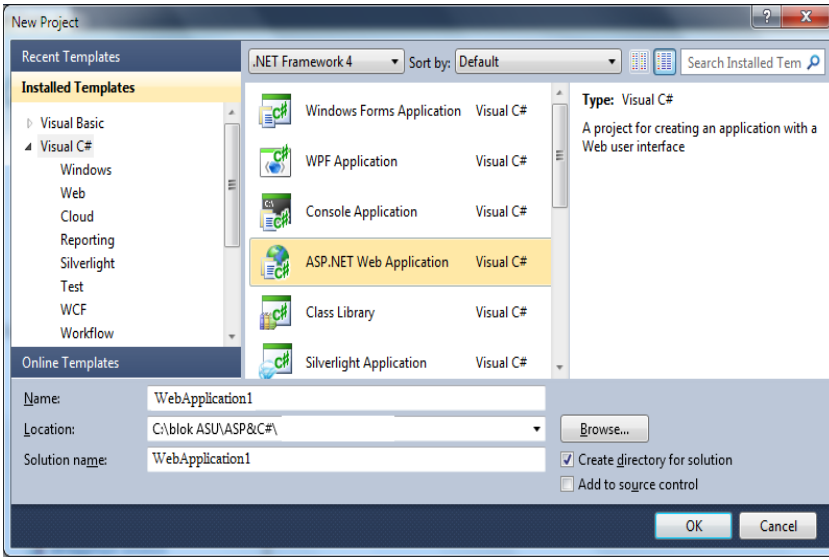


ნახ.5.6

5.4. ინტერაქტიული Web-გვერდის შექმნა

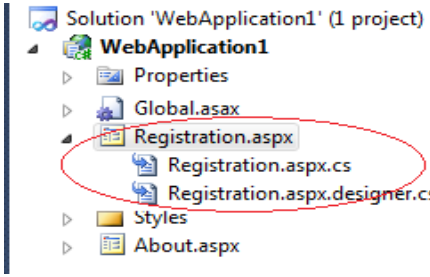
ამოცანა-5.1. ავაგოთ ახალი Web-გვერდი, რომელზეც მომხმარებელი შეიტანს საკუთარ მონაცემებს და გადააგზავნის სერვერზე.

შევქმნათ ახალი ASP.NET აპლიკაცია პროექტის სახელით WebApplication1 (ნახ.5.7).



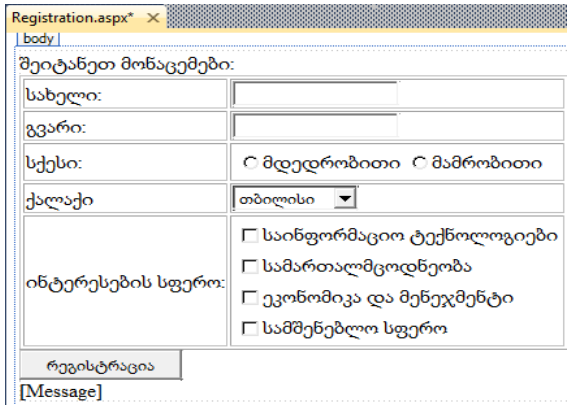
ნახ.5.7

დავამატოთ ფაილები: Registration.aspx და Registration.aspx.cs.



ნახ.5.8

Web-გვერდის სარეგისტრაციო ფორმის მაკეტი ნაჩვენებია 5.9 ნახაზზე.



ნახ.5.9

ფორმაზე მოთავსებულია სერვერული მართვის ელემენტები form, asp:TextBox, asp:DropDownList, asp:CheckBoxList, asp:Button, asp:Label და ა.შ., რომლებიც ასახულია Registration.aspx ფაილის 5.3 ლისტინგში. გახსენით Registration.aspx და შეიტანეთ შემდეგი კოდი:

```
<!— ლისტინგი_5.3 ———— –>
<%@ Page Language="C#" AutoEventWireup="true"
     CodeBehind="Registration.aspx.cs"
     Inherits="WebApplication1.Registration" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html >
 <head>
  <title>რეგისტრაციის ფორმა</title>
 </head>
 <body>
  <form method="post" runat="server" id="registration">
   შეიტანეთ მონაცემები:
   <table border="1">
```

```
  <tr>
   <td>სახელი:</td>
   <td>
 <asp:TextBox id="FirstName" runat="server"></asp:TextBox>
   </td>
  </tr>
  <tr>
   <td>გვარი:</td>
   <td>
 <asp:TextBox id="LastName" runat="server"></asp:TextBox>
   </td>
  </tr>
  <tr>
   <td>სქესი:</td>
    <td><asp:RadioButtonList id="Sex" runat="server"
             RepeatDirection="Horizontal">
        <asp:ListItem Value="მდედრობითი"></asp:ListItem>
        <asp:ListItem Value="მამრობითი"></asp:ListItem>
      </asp:RadioButtonList></td>
    </tr>
    <tr>
     <td>ქალაქი</td>
     <td><asp:DropDownList id="City" runat="server">
       <asp:ListItem Value="თბილისი"></asp:ListItem>
       <asp:ListItem Value="ქუთაისი"></asp:ListItem>
       <asp:ListItem Value="რუსთავი"></asp:ListItem>
       <asp:ListItem Value="გორი"></asp:ListItem>
       <asp:ListItem Value="ბათუმი"></asp:ListItem>
       <asp:ListItem Value="თელავი"></asp:ListItem>
      </asp:DropDownList></td>
    </tr>
    <tr>
      <td>ინტერესების სფერო:</td>
    <td>
     <asp:CheckBoxList id="Interests" runat="server">
 <asp:ListItem Value="საინფორმაციო ტექნოლოგიები">
 </asp:ListItem>
<asp:ListItem Value="სამართალმცოდნეობა"></asp:ListItem>
<asp:ListItem Value="ეკონომიკა და მენეჯმენტი"></asp:ListItem>
<asp:ListItem Value="სამშენებლო სფერო"></asp:ListItem>
```

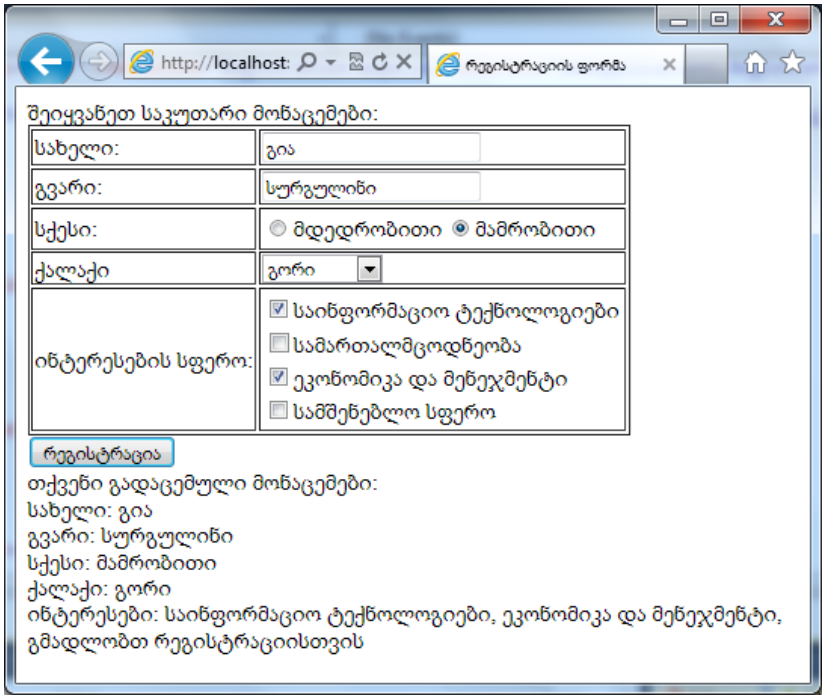
```
      </asp:CheckBoxList></td>
     </tr>
    </table>
   <asp:Button id="Register" runat="server" Text="რეგისტრაცია"
OnClick="Register_Click"></asp:Button>
    <br />
    <asp:Label id="Message" runat="server"></asp:Label>
   </form>
  </body>
</html>
```

Web-გვერდის ჩატვირთვის და მონაცემების შევსების შემდეგ „რეგისტრაცია“ Button-ის დაჭერისას გამოიძახება OnClick მოვლენაზე მიბმული მეთოდი Register_Click. ის აღიწერება C# კოდში, რომლის 5.4 ლისტინგი მოცემულია ქვემოთ. გავხსნათ Registration.aspx.cs ფაილი და შევიტანოთ შემდეგი კოდი:

```
// ── ლისტინგი_5.4 ──────────
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
namespace WebApplication1
{
  public partial class Registration : System.Web.UI.Page
    {
     protected void Page_Load(object sender, EventArgs e){ }
     protected void Register_Click(object sender, EventArgs e)
      {
       System.Text.StringBuilder sb = new
                                    System.Text.StringBuilder();
       sb.Append("თქვენი გადაცემული მონაცემები:<br>");
       sb.AppendFormat("სახელი: {0}<br>", FirstName.Text);
       sb.AppendFormat("გვარი: {0}<br>", LastName.Text);
       sb.AppendFormat("სქესი: {0}<br>", Sex.SelectedValue);
       sb.AppendFormat("ქალაქი: {0}<br>", City.SelectedValue);
       sb.Append("ინტერესები: ");

          foreach (ListItem item in Interests.Items)
```

```
            {
              if (item.Selected)
                 sb.AppendFormat("{0}, ", item.Value);
            }
            sb.Append("<br>გმადლობთ რეგისტრაციისთვის");
            Message.Text = sb.ToString();
        }
    }
}
```

Web-გვერდი და მისი შესრულების შედეგი მოცემულია 5.10 ნახაზზე.

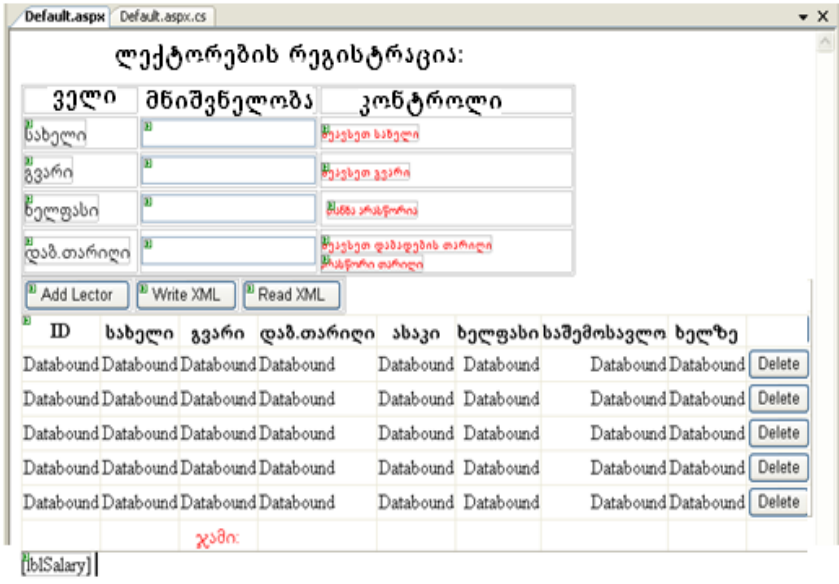


ნახ.5.10

5.5. DataSet / GridView ობიექტებთან მუშაობა და XML ფაილი

ინტერაქტიულ რეჟიმში მონაცემების შეტანისას ერთ-ერთი მნიშვნელოვანი საკითხია მათი კონტროლის პროცედურების დამუშავება, შეტანილ მონაცემთა ეკრანზე ასახვის საშუალებების GridView / DataSet გამოყენება. აგრეთვე მეტად მოსახერხებელია შედეგების XML ფაილში შენახვა და XML ფაილიდან მათი ამოღების პროცედურების შექმნა.

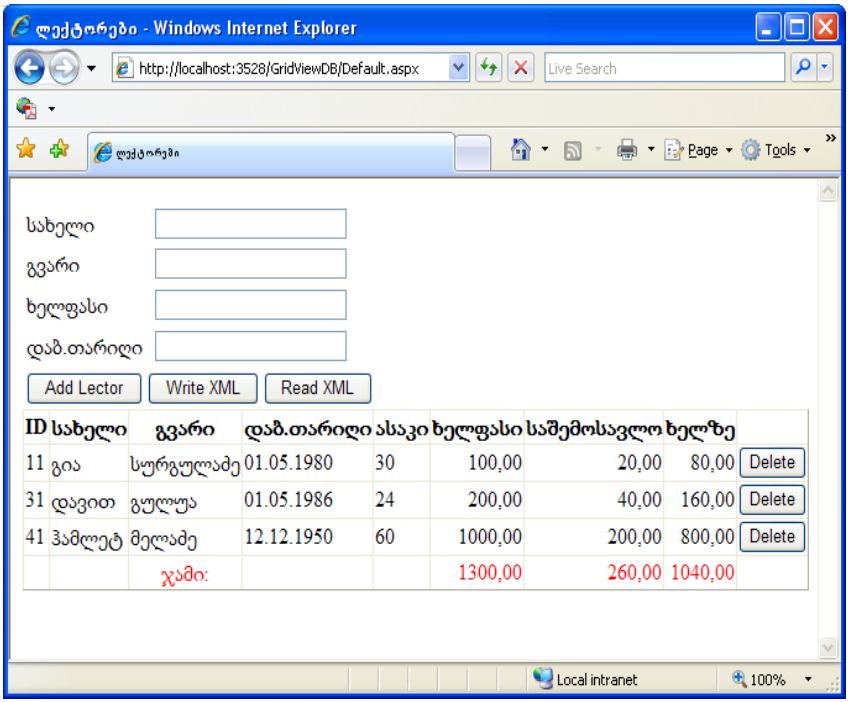
ამოცანა-5.2. Visual Studio .NET გარემოში ავაგოთ Web-პროექტი ლექტორთა სარეგისტრაციო მონაცემების შესატანად. განვახორციელოთ მონაცემთა ვიზუალური და ავტომატური კონტროლის საშუალებების გამოყენება. Add Lector-ღილაკით შეტანილი მონაცემები აისახოს GridView ცხრილში უნიკალური ID-ს მქონე სტრიქონის სახით, მომზადდეს სარეგისტრაციო ცხრილი ახალი ინფორმაციის შესატანად (ნახ.5.11). ავტომატური კონტროლისთვის გამოყენებულ იქნეს ToolBox->Validation;



ნახ.5.11

- GridView ცხრილში ავაგოთ სვეტები შესაბამისი დასახელებებით. გარდა სარეგისტრაციო მონაცემებისა, დავამატოთ გაანგარიშებადი ველებიც: ***ასაკი, საშემოსავლო_გადასახადი, ხელზე_ასაღები_თანხა***. ეს ველები განისაზღვრება C#-კოდში;
- GridView ცხრილში დავამატოთ ახალი სვეტი Delete-ფუნქციით, არასასურველი სტრიქონის ოპერატიულად წასაშლელად;
- GridView ცხრილის Fother სტრიქონში გამოვიტანოთ „ჯამი:“ ხელფასის, საშემოსავლოს და ***ხელზე_ასაღები_თანხის*** სვეტებისათვის ჯამური მნიშვნელობების გამოსატანად.

5.12 ნახაზზე ნაჩვენებია Web-გვერდის მაკეტი ბრაუზერში.



ნახ.5.12

ქვემოთ აღიწერება ღილაკები, რომლებიც ფორმაზეა განლაგებული. მათი დანიშნულებაა მონაცემთა დამატება, კონტროლი, შენახვა XML ფაილში, შემდეგ კი ამ ფაილიდან ამოღება. განვიხილოთ მათი პროგრამული კოდების ლისტინგები:

```
// ── ლისტინგი_5.5── Add_Lector ─────
 protected void Button1_Click(object sender, EventArgs e)
   {
      DataSet dsLectors = Session["MyDataSet"] as DataSet;
      DataTable dtLectors = dsLectors.Tables["Lectors"];

      DataRow newlector = dtLectors.NewRow();
      //newlector["ID"] = "1";
      newlector["FirstName"] = txtFirstName.Text;
      newlector["LastName"] = txtLastName.Text;
      if (!String.IsNullOrEmpty(txtSalary.Text))
           newlector["Salary"] = Decimal.Parse(txtSalary.Text);

      newlector["BirthDate"] = DateTime.Parse(txtBirthDate.Text);
      dtLectors.Rows.Add(newlector);
      object sumSalary = dtLectors.Compute("SUM(Salary)", "");
      object sumTax = dtLectors.Compute("SUM(Tax)", "");
      object sumNettoSalary = dtLectors.Compute("SUM(NettoSalary)", "");
      //lblSalary.Text = sumSalary.ToString();
     GridView1.Columns[5].FooterText = String.Format("{0:F2}", sumSalary);
     GridView1.Columns[6].FooterText = String.Format("{0:F2}",sumTax);
     GridView1.Columns[7].FooterText = String.Format("{0:F2}", sumNettoSalary);

      Session["MyDataSet"] = dsLectors;
      GridView1.DataSource = dsLectors;
      GridView1.DataBind();
   }
```

- დავამატოთ ორი ღილაკი: Write_XML (სტრიქონების შესანახად XML ფაილში) და Read_XML (სტრიქონების ამოსაღებად XML ფაილიდან).

```
// ── ლისტინგი_5.6 ──  Write_XML ------------------
protected void Button2_Click(object sender, EventArgs e)
   {
      DataSet ds = Session["MyDataSet"] as DataSet;
      ds.WriteXml(Request.PhysicalApplicationPath + "\\lectors.xml");
      //Response.Redirect("~/lectors.xml");
   }

// ── ლისტინგი_5.7 ──  Read_XML ---------------
protected void Button3_Click(object sender, EventArgs e)
   {
      DataSet ds = Session["MyDataSet"] as DataSet;
      ds.ReadXml(Request.PhysicalApplicationPath + "\\lectors.xml");
      DataTable dtLectors = ds.Tables["Lectors"];
      object sumSalary = dtLectors.Compute("SUM(Salary)", "");
      object sumTax = dtLectors.Compute("SUM(Tax)", "");
      object sumNettoSalary = dtLectors.Compute("SUM(NettoSalary)", "");
      //lblSalary.Text = sumSalary.ToString();
      GridView1.Columns[5].FooterText = String.Format("{0:F2}", sumSalary);
      GridView1.Columns[6].FooterText = String.Format("{0:F2}", sumTax);
     GridView1.Columns[7].FooterText = String.Format("{0:F2}", sumNettoSalary);
      GridView1.DataSource = ds;
      GridView1.DataBind();
   }
```

- GridView ცხრილთან სამუშაოდ გამოიყენება DataSet ობიექტი, რომელსაც C#-კოდში აქვს შემდეგი სახე:

```
// ── ლისტინგი_5.8 ── DataSet ──
private DataSet GetDataSet()
   {
      DataTable lectors = new DataTable("Lectors");
      //Add the DataColumn using all properties
      DataColumn id = new DataColumn("ID");
      id.DataType = typeof(int);
      id.Unique = true;
      id.AutoIncrement = true;
      id.AutoIncrementSeed = 1;
      id.AutoIncrementStep = 10;
      id.AllowDBNull = false;
```

```
id.Caption = "ID";
lectors.Columns.Add(id);

//Add the DataColumn using defaults
DataColumn firstName = new DataColumn("FirstName");
firstName.DataType = typeof(string);
firstName.MaxLength = 35;
firstName.AllowDBNull = false;
lectors.Columns.Add(firstName);

DataColumn lastName = new DataColumn("LastName");
lastName.DataType = typeof(string);
lastName.MaxLength = 50;
lastName.AllowDBNull = false;
lectors.Columns.Add(lastName);

DataColumn salary = new DataColumn("Salary", typeof(decimal));
salary.DefaultValue = 0.00m;
lectors.Columns.Add(salary);

DataColumn birthDate = new DataColumn("BirthDate",
                                                typeof(DateTime));
//birthDate.DefaultValue = DateTime.Now;
birthDate.AllowDBNull = true;
lectors.Columns.Add(birthDate);

DataColumn age = new DataColumn("Age", typeof(DateTime));
age.ColumnMapping = MappingType.Hidden;
age.Expression = "BirthDate";
lectors.Columns.Add(age);

DataColumn tax = new DataColumn("Tax", typeof(decimal));
tax.ColumnMapping = MappingType.Hidden;
tax.DataType = typeof(decimal);
tax.Expression = "salary*0.2";
lectors.Columns.Add(tax);

DataColumn netto = new DataColumn("NettoSalary", typeof(decimal));
netto.ColumnMapping = MappingType.Hidden;
netto.DataType = typeof(decimal);
netto.Expression = "salary - salary*0.2";
lectors.Columns.Add(netto);
```

```
        ////Derived column using expression
        //DataColumn lastNameFirstName = new DataColumn("LastName and
FirstName");
        //lastNameFirstName.DataType = typeof(string);
        //lastNameFirstName.MaxLength = 70;
        //lastNameFirstName.Expression = "lastName + ', ' + firstName";
        //employee.Columns.Add(lastNameFirstName);
        DataSet ds = new DataSet();
        ds.Tables.Add(lectors);
        return ds;
    }
```

- XML ფაილს, მასში სტრიქონების (ობიექტების) ჩაწერის შემდეგ ექნება ასეთი სახე:

```
<!—— ლისტინგი_5.9 — XML ჩანაწერების შენახვის სტრუქტურა –>
<?xml version="1.0" standalone="yes"?>
<NewDataSet>
  <Lectors>
    <ID>11</ID>
    <FirstName>გია</FirstName>
    <LastName>სურგულაძე</LastName>
    <Salary>100</Salary>
    <BirthDate>1980-05-01T00:00:00+04:00</BirthDate>
  </Lectors>
  <Lectors>
    <ID>31</ID>
    <FirstName>დავით</FirstName>
    <LastName>გულუა</LastName>
    <Salary>200</Salary>
    <BirthDate>1986-05-01T00:00:00+04:00</BirthDate>
  </Lectors>
  <Lectors>
    <ID>41</ID>
    <FirstName>გიორგი</FirstName>
    <LastName>სურგულაძე</LastName>
    <Salary>1900</Salary>
    <BirthDate>1980-12-30T00:00:00+04:00</BirthDate>
  </Lectors>
</NewDataSet>
```

VI თავი. ინსტრუქციები სისტემის საპილოტო ვერსიისა და საკურსო პროექტის გასაფორმებლად

6.1. აპლიკაციის საპილოტო ვერსიის ტესტირება და სადემონსტრაციოდ მომზადება

პროგრამული აპლიკაციის საპილოტო ვერსია არის ასაგები რეალური სისტემის სადემონსტრაციო, „სათამაშო“ ვარიანტი, რომელშიც კარგად ჩანს საპრობლემო სფეროს ობიექტის ავტომატიზაციის ამოცანების სანიმუშო მაგალითები. აქ ალგორითმულად და პროგრამულად რეალიზებულია დეველოპერების მიერ კომპიუტერული სისტემის მოდელი, თავისი მონაცემებით და მეთოდებით (ფუნქციებით).

ტესტირება მოიცავს ჩვენ მიერ აწყობილი სისტემის ცალკეული ამოცანების შესრულებაზე გაშვებას. საჭიროა მონაცემთა ბაზაში, ცხრილებში ინფორმაციის შეტანა-გამოტანისა და მონაცემთა კორექტირების პროცედურების შემოწმება, აგრეთვე მენიუს, ღილაკების და სხვა ვიზუალური კომპონენტების ფუნქციონირების სისწორის კონტროლი.

ბოლოს, უნდა დაიწეროს მოთხოვნები მონაცემთა ბაზიდან მონაცემთა ამოსაღებად და დასამუშავებლად. შემოწმდება, თუ რამდენად სწორად ასრულებს აგებული Windows ან Web აპლიკაციის პროგრამული კოდი წინასწარ გათვალისწინებულ დავალებებს.

6.2. საპრეზენტაციო ფაილის და საკურსო პროექტის დოკუმენტაციის მომზადება

საბოლოო შედეგების პრეზენტაციის მიზნით კომპიუტერისა და პროექტორის გამოსაყენებლად შეირჩევა აპლიკაციის პროექტის საილუსტრაციო მასალა: მიზანი, ამოცანები, გადაწყვეტა, შედეგები და რეკომენდაციები. საპრეზენტაციო სლაიდები მომზადდება Ms_PowerPoint ინსტრუმენტით.

პროექტის დოკუმენტაცია მზადდება MsWord ფაილის სახით, ნაბეჭდი A4 ფორმატით, Sylfaen ფონტით, 11p და 1.15 ინტერვალით, არეები 2 სმ, ძირითადი ტექსტი 25–40 გვერდი.

7. ლიტერატურა

1. სურგულაძე გ. ვიზუალური დაპროგრამება C#_2010 ენის ბაზაზე. სტუ. თბ., 2012. http://gtu.ge/books/GiaSurg_C_2010.pdf

2. სურგულაძე გ. დაპროგრამების მეთოდები და ინსტრუმენტები (UML, MsVisio, C++). სტუ. თბ., 2007. http://www.gtu.edu.ge/katedrebi/kat94/pdf/BC++B-22.pdf

3. გოგიჩაიშვილი გ., სუხიაშვილი თ. სისტემების ობიექტ–ორიენტირებული ანალიზი და დაპროექტება. სტუ. თბ., 2012. http://gtu.ge/books/suxi_gogichaishvili.pdf

4. თევდორაძე მ. საბუღალტრო აღრიცხვის მეთოდური სახელმძღვანელო. სტუ. თბ., 2005

5. სურგულაძე გ. მონაცემთა ბაზების სამაგიდო სისტემები. სტუ. თბ., 2004. www.gtu.edu.ge/katedrebi/kat94/pdf/DB_MsAccess.pdf

6. სურგულაძე გ. დაპროგრამების მეთოდები: საკურსო პროექტის მეთოდური სახელმძღვანელო (UML, MsVisio, C++Builder).. სტუ. თბ., 2007. www.gtu.edu.ge/katedrebi/kat94/pdf/Project-CPP-28.pdf

7. სურგულაძე გ., შონია ო., ყვავაძე ლ. მონაცემთა განაწილებული ბაზების მართვის სისტემები (MsSQL Server, Access, InterBase, JDBC, Oracle). სტუ, თბ., 2004

8. სურგულაძე გ., ბულია ი., თურქია ე. Web-აპლიკაციების დამუშავება მონაცემთა ბაზების საფუძველზე (ADO.NET, ASP.NET, C#). სახელმძღვ., სტუ, თბ., 2009.

9. სურგულაძე გ., ბულია ი., თურქია ე. Web-აპლიკაციების აგება ASP.NET & C# პაკეტებით .NET პლატფორმაზე. დამხმ.სახ., ლაბ.პრაქტიკუმი. სტუ, თბ., 2009. http://www.gtu.ge/books/GiaSurg_ASP_Lab.pdf

10. სურგულაძე გ., ბულია ი. კორპორაციულ Web აპლიკაციათა ინტეგრაცია და დაპროექტება. მონოგრ., სტუ. თბ., 2012. ელ–ვერსია: http://www.gtu.ge/books/GiaSurg_Book_2012.pdf

8. დანართები

8.1. დანართი N1

/საკურსო პროექტის სატიტულო გვერდი/

საქართველოს ტექნიკური უნივერსიტეტი

მართვის ავტომატიზებული სისტემების
დეპარტამენტი (№94)

საკურსო პროექტი

დისციპლინაში: **„პროგრამული აპლიკაციების დეველოპმენტის საფუძვლები“**

ჯგ.№: 108 . . .

სტუდენტი . . . . . .

თემა: “საკურსო პროექტის სათაური“

ხელმძღვანელი . . . . . .

ჩაბარების შედეგი . . . . . .
თარიღი . . . . .

თბილისი-2014

8.2. დანართი N2

/საკურსო პროექტის საწყისი მონაცემების ფორმა/

პროექტის საწყისი მონაცემები

(სტუდენტი მიიღებს პროფესორისგან)

1. საპრობლემო სფერო (კვლევის ობიექტი)

2. მართვის სფერო (ფუნქციური ამოცანა)

3. კლასები (დასახელება/მონაცემები/მეთოდები)

4. ობიექტების რაოდენობა (კლასებში)

5. ინტერფეისის დიზაინი (ფორმის სტრუქტურა)

6. საანგარიშო მონაცემები

პროფ. ხელმოწერა . . . . . .
თარიღი . . . . .

8.3. დანართი N3
საკურსო პროექტის ინდივიდუალური და ჯგუფური თემები

1. საპრობლემო სფერო: **ბიბლიოთეკა**
    1.1. ობიექტი: მკითხველთა რეგისტრაცია
    1.2. ობიექტი: წიგნების ანბანური კატალოგები
    1.3. ობიექტი: წიგნების თემატური კატალოგები
    1.4. ობიექტი: წიგნების გაცემა/დაბრუნება
    1.5. ობიექტი: კადრები/ხელფასები

2. საპრობლემო სფერო: **ფაკულტეტი**
    2.1. ობიექტი: სტუდენტები, ჯგუფები, კურსები
    2.2. ობიექტი: ფაკულტეტის კათედრები, სპეციალობები
    2.3. ობიექტი: ჯგუფები, საგნები, კრედიტები
    2.4. ობიექტი: სტუდენტები, საგნები, გამოცდები, შედეგები
    2.5. ობიექტი: ლექტორები, კათედრები, ჯგუფები

3. საპრობლემო სფერო: **სუპერმარკეტი**
    3.1. ობიექტი: პროდუქტი, ფირმა, ფასი, კატეგორია
    3.2. ობიექტი: კლიენტთა მომსახურება, სალარო/ჩეკი
    3.3. ობიექტი: ელექტრონული შეკვეთა ბინაზე მიტანით
    3.4. ობიექტი: დღიური/თვიური/წლიური ვაჭრობა
    3.5. ობიექტი: საწყობში პროდუქციის აღრიცხვა

4. საპრობლემო სფერო: **აფთიაქი**
    4.1. ობიექტი: მედიკამენტი, ქვეყანა, ფასი, კატეგორია
    4.2. ობიექტი: კლიენტთა მომსახურება, სალარო/ჩეკი
    4.3. ობიექტი: ელექტრონული შეკვეთა ბინაზე მიტანით
    4.4. ობიექტი: დღიური/თვიური/წლიური ეკონომიკური მაჩვენებლები
    4.5. ობიექტი: აფთიაქის საწყობი
    4.6. ან სხვ.

5. საპრობლემო სფერო: **საწარმოო ფირმა**

5.1. ობიექტი: პროდუქტი, ფასი, კატეგორია

5.2. ობიექტი: პროდუქტი, თვითღირებულება, ფასი, მოგება, რენტაბელობა

5.3. ობიექტი: პროდუქტი, ნედლეული, მიმწოდებელი, ნედლეულის_ფასი

5.4. ობიექტი: თვიური/წლიური შემოსავალი, ხარჯი, მოგება

5.5. ობიექტი: ნედლეულის და მზა პროდუქციის საწყობები

6. საპრობლემო სფერო: **კლინიკა**

6.1. ობიექტი: პაციენტი, დაავადება, მკურნალი_ექიმი

6.2. ობიექტი: ექიმი, ნოზოლოგიური_განყოფილება, ოთახი, ტელეფონი

6.3. ობიექტი: პაციენტი, დაავადება, მკურნალობის_ფასი

6.4. ობიექტი: თვიური/წლიური შემოსავალი, ხარჯი, მოგება

6.5. ობიექტი: ნოზოლოგიური_განყოფილება, საწოლების რაოდენობა, მკურნალობის_ვადა, მდგომარეობა

7. საპრობლემო სფერო: **მარკეტინგი**

3.1. ობიექტი: ავტომანქანების ბაზარი (ფირმა, მოდელი, სხვ.)

3.2. ობიექტი: ავტოპროფილაქტიკა, მომსახურების აღრიცხვა

3.3. ობიექტი: კომპიუტერების მაღაზია

3.4. ობიექტი: წიგნების მაღაზია სექციების მიხედვით

3.5. ობიექტი: პარფიუმერიის მაღაზია

8. საპრობლემო სფერო: **რესტორანი**

8.1. ობიექტი: მენიუ, კერძები, ფასები

8.2. ობიექტი: წინასწარი დაჯავშნის ამოცანა

8.3. ობიექტი: კლიენტთა მაგიდის მომსახურება

8.4. ობიექტი: შეკვეთების მიღება კერძების ბინაზე მიტანით

8.5. ობიექტი: თვიური/წლიური შემოსავალი, ხარჯი, მოგება

9. საკურსო თემების დაზუსტება ან სხვა სფეროებიდან შერჩევა ხდება პროფესორის და სტუდენტის კონსულტაციების პროცესში.

8.4. დანართი N4
საკურსო პროექტის რეპორტის სარჩევი

1. საპრობლემო სფეროს ობიექტის ბიზნესპროცესების შინაარსი და BPMN დიაგრამა. ამოცანის დასმა;

2. ასაგები კომპიუტერული სისტემის ფუნქციონალური და არაფუნქციონალური მოთხოვნები: როლების და მათი ქმედებების UML დიაგრამები (UseCase, Activity);

3. ელექტრონული დოკუმენტების ფორმები: საწყისი, ნორმატიული, ოპერატიული და გამომავალი;

4. სისტემის მონაცემთა ბაზის სტრუქტურა ER დიაგრამა (MsVisio Database);

5. სისტემის მონაცემთა ბაზა (MsAccess), ცხრილები (Tables) ჩანაწერებით;

6. მომხმარებელთა ინტერფეისები (სამუშაო ფორმები), MsVisual Studio.NET ინტეგრირებულ გარემოში (Windows Form ან ASP.NET). MsAccess ბაზის და C# -ენის საფუძველზე;

7. საპრობლემო სფეროს ფუნქციონალური ამოცანის (ჯგუფური პროექტის დროს – ამოცანების) გადაწყვეტის პროცესის ავტომატიზაცია (დაპროგრამება და ტესტირება);

8. მომხმარებელთა ინტერფეისებში MsAccess ბაზიდან მონაცემთა ძებნა და ამორჩევა, შესაბამისი C# კოდებით;

9. სისტემის დემოვერსია და საპრეზენტაციო სლაიდები.

გადაეცა წარმოებას 24.05.2013 წ. ხელმოწერილია დასაბეჭდად 27.05.2013 წ. ოფსეტური ქაღალდის ზომა 60X84 1/16. პირობითი ნაბეჭდი თაბახი 6. ტირაჟი 100 ეგზ.

საგამომცემლო სახლი ,,ტექნიკური უნივერსიტეტი''
თბილისი, მ. კოსტავას 77