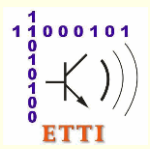


Universitatea POLITEHNICA din Bucuresti
Facultatea de Electronica, Telecomunicatii si Tehnologia Informatiei

Catedra de Telecomunicatii

ITNQ
2011

Laboratorul **SAIM** – sala **B123**

04/03/2012

POO – an II

TPI – an III

Inginerie Software in Comunicatii (ISC)

PSNIM
2012

2011-2012

Laborator 1

Dezvoltarea aplicaţiilor Java si ingineria inversa UML folosind mediul de dezvoltare integrat NetBeans IDE 6. Crearea diagramelor UML ale cazurilor de utilizare si de clase

Descrierea laboratorului

In aceasta lucrare de laborator vor fi acoperite urmatoarele probleme:

- **Dezvoltarea aplicaţiilor Java folosind facilitatile UML** din NetBeans 6
- **Ingineria inversă a aplicaţiilor Java folosind facilitatile UML** din NetBeans 6
- **Crearea diagramelor UML ale cazurilor de utilizare in NetBeans 6**
- **Crearea diagramelor UML de clase in NetBeans 6**
- **Teme de casa**

Atentie: **La inceputul laboratoarelor stergeti mai intai toate proiectele existente** (click dreapta pe nodul proiectului in fereastra ***Projects***, selectati ***Delete*** si **confirmati ca doriti sa fie sterse sursele – in cazul proiectelor Java**). **La finalul laboratoarelor stergeti proiectele create**.

1.1. Dezvoltarea aplicaţiilor Java folosind facilitatile UML din NetBeans

In continuare veti învăţa cum să folosiţi **caracteristicile UML ale IDE-ului NetBeans** pentru a crea o diagramă de clasa UML.

Apoi veti folosi generarea de cod si facilitatile de **inginerie inversa** pentru **a crea o mica aplicatie bancara**, pe care **o puteti testa** prin **executia unei clase de test.** Această aplicatie bancara este **mult simplificată**; un model real pentru o aplicatie de acest tip ar necesita mai multe detalii.

Scopul acestui material este de **introducere in facilitatile UML ale IDE-ului NetBeans**, nu de a invata conceptele UML sau limbajul de programare Java. Pentru a parcurge materialul care urmeaza, **IDE-ul NetBeans** trebuie să fie instalat pe sistemul dumneavoastra si ar trebui sa fiti familiarizati cu piesele de bază ale acestuia. Pentru a parcurge etapele care urmeaza **aveti nevoie de software-ul din următorul tabel**:

Software-ul	Versiunea obligatorie
NetBeans IDE	**versiunea 6.1** sau versiunea 6.0
Java Development Kit (JDK)	**versiunea 6** sau versiunea 5

Ar trebui de asemenea sa fiti familarizati cu **notiunile de baza ale limbajului de programare Java si UML (*Unified Modeling Language*)**.

1.1.1. Crearea proiectului Java al aplicatiei

În această secţiune, veti crea un nou proiect Java pentru aplicaţia pe care urmează să o dezvoltaţi în acest tutorial.

1. Din meniul principal, selectaţi ***File*> *New Project*** şi apoi faceţi următoarele în ***New Project Wizard***:

   Sub ***Categories***, selectaţi ***Java***.

   Sub ***Projects***, selectati ***Java Application***

   Click ***Next***.

2. In campul *Project Name* completati **JavaPrj**.

3. Pentru campul ***Project Location***, click ***Browse***, si navigaţi la orice director de pe computer (in laborator alegeti drive-ul **D:** si directorul **\isw**, si creati sau selectati un **subdirector cu numarul grupei**, apoi creati un **subdirector cu nume diferit de cele existente**, de exemplu: **D:\isw\441E\Proiect1**). Click ***Open*** si completati **MyPrj**.

4. Debifati optiunile ***Set as Main Project*** si ***Create Main Class***.

5. Click ***Finish***. O casuta de progres apare. Cand proiectul **JavaPrj** este creat, acesta apare in fereastra ***Projects***.

1.1.2. Crearea proiectului UML si a diagramei de clase

În această secţiune, veti crea proiectul UML si diagrama de clase pentru aplicatie.

Un **proiect UML** este un mecanism prin care puteti stoca şi gestiona o colecţie de fisiere pentru un model UML.

Un **model UML** conţine toate diagramele modelului respectiv, elementele asociate lor şi metadatele legate de modelul UML.

1. Pentru a crea un proiect UML, selectaţi ***File*> *New Project*** şi apoi faceţi următoarele:

   Sub ***Categories***, selectati ***UML***.

   Sub ***Projects***, selectati ***Java-Platform Model***.

   Faceti ***Click*** pe ***Next***.

   Se va deschide caseta de dialog ***New Java-Platform Model***.

2. In campul ***Project Name*** completati **UMLPrj.** De mentionat ca dupa ce completati numele proiectului, IDE-ul sugereaza automat acest nume si pentru campul ***Project Folder***.

3. Verificaţi că locatia proiectul este **MyPrj**.

4. Faceti ***Click*** pe Finish. IDE-ul creeaza proiectului UML si apare caseta de dialog ***Create New Diagram.***

5. Din lista ***Diagram type*** alegeti ***Class Diagram***.

6. In campul ***Diagram Name*** completati **BankClassDiagram**.

7. Lasati **UMLPrj** in campul ***Namespace*** si click ***Finish***. IDE-ul realizeaza următoarele:

Adaugă nodul proiectului **UMLPrj** in fereastra ***Projects***.

Creeaza un nod **BankClassDiagram** sub nodul ***Model***

Afişează noua diagramă în editorul de diagrame (diagrama este goala în acest moment)

Deschide (***Modeling***) ***Palette***

1.1.3. Adăugarea şi definirea elementelor claselor

Acum, adăugaţi şi definiti elementele clasei care vor constitui aplicaţia dumneavoastra. Veti folosi pictograma ***Class*** din ***Palette*** pentru a crea elementele.

1. Din sectiunea ***Basic*** din ***Palette***, selectati pictograma ***Class*** si faceti click in editorul de diagrama. Aceasta actiune va plasa un element al clasei in editorul de diagrama.

2. Deselectaţi pictograma ***Class*** făcând click dreapta oriunde în editor.

3. Selectaţi elementul nou adaugat, introduceti **BankAccount** şi apăsaţi ***Enter***. IDE-ul face următoarele:

Denumeste elementul **BankAccount**

Creează o metoda publica, **BankAccount()**

Afişează proprietăţile clasei **BankAccount** în fereastra ***Properties***

4. Avand elementul **BankAccount** selectat in editor, faceti click dreapta pe ***Attributes*** si alegeti ***Insert Attribute*** din meniul pop-up.

O linie editor se deschide şi afişează următoarele informaţii:

```
visibility type name[ranges]=initialValue{name=value}
```

5. Scrieti **balance** şi apăsaţi ***Enter***. Un atribut privat numit **balance** de tip **int** apare in clasa **BankAccount**. Următoarele operaţii sunt create pe clasa:

```
public int getBalance()
public void setBalance(int val)
```

6. Cu elementele clasei **BankAccount** selectate în editorul de diagrama, faceţi click dreapta pe cuvântul ***Operations*** şi alegeţi ***Insert Operation*** de la meniul pop-up.

O linie editor se deschide şi afişează următoarele informaţii:

```
visibility returnType name(parameter) {properties...}
```

7. Scrieti **withdraw**, deplasati cursorul (folosind *forward arrow* de la tastatura) în paranteze, introduceti **int amount** şi apăsaţi ***Enter***. IDE-ul adaugă noua metoda in clasa element, după cum urmează:

```
public void withdraw(int amount)
```

1.1.4. Adăugarea altor elemente la diagramă

În această secţiune utilizaţi mai multe pictograme UML din ***Palette*** pentru a adăuga interfeţe, pachetele, atribute, precum şi operaţii pentru aplicatia dumneavoastră.

1. In secţiunea de bază a ***Palette***, selectaţi pictograma ***Package***

Notă: Dacă este necesar, defilaţi în jos pentru a vedea pictogramele suplimentare ***Modeling***.

2. Faceţi click in editorul de diagrama pentru a adăuga un pachet de elemente la diagrama de clasă.

3. Faceţi click dreapta oriunde în editorul de diagrama pentru a deselecta pictograma ***Package***. **Notă:** Pe măsură ce adăugaţi elemente de modelare la diagramă, aveţi posibilitatea să le selectaţi şi sa le glisaţi la noi locaţii în editorul de diagrama pentru a îmbunătăţi aspectul diagrama. Fiti atenti atunci când faceţi click dreapta, la fel ca în anumite poziţii, această acţiune deschide un meniu de tip pop-up pentru editorul de diagrama. Dacă se întâmplă acest lucru, trebuie doar să faceţi click din nou în spaţiul alb din editorul de diagramă.

4. Cu elementul *package* (pachet) selectat, completati **bankpack** şi apăsaţi ***Enter***.

5. In sectiunea ***Basic*** din ***Palette***, selectati pictograma ***Interface*** si dati click în editorul de diagrama.

6. Faceţi click dreapta oriunde în editorul de diagrama pentru a deselecta pictograma ***Interface***.

7. Cu elementul *interface* (interfata) selectat, completati **Bank** şi apăsaţi ***Enter***.

8. Adaugati o operatie deposit pentru interfata **Bank**. Veti adauga operatii la interfata în acelaşi mod in care au fost adăugate pentru clase (Pasul 6 din sectiunea precedenta). Definiţi operaţia, după cum urmează:

```
public void deposit(int amount)
```

9. In ***Palette***, selectati pictograma ***Class*** şi faceţi click in editorul din diagrama de doua ori.

10. Faceţi click dreapta oriunde în editorul de diagrama pentru a deselecta pictograma ***Class***. **Notă:** Dacă ati adaugat prea multe elemente de clasă, deselectati pictogramă class , apoi faceţi click dreapta pe elementul din clasa pe care doriţi să îl ştergeţi şi selectaţi ***Edit > Delete***.

11. Numiti elementele claselor **Checking** si **AccountTest** si redimensionati daca e nevoie.

1.1.5. Identificarea relatiilor (asocierilor) între elemente

În această secţiune, veti utiliza pictograme UML din ***Palette*** pentru a identifica asocierile între elementele clasei.

1. Din sectiunea ***Basic*** din ***Palette***, selectati pictograma ***Implementation*** şi faceţi click in interiorul elementelor clasei **BankAccount**.

2. Faceţi click în elementul interfaţa **Bank** şi faceţi click dreapta oriunde în editorul de diagrama pentru a deselecta pictograma ***Implementation***. O relatie de implementare apare intre clasa si elementul de interfata. O relatie de implementare indica o relatie intre o clasa si o interfata.

3. In sectiunea ***Basic*** din ***Palette*** selectati pictograma ***Generalization***.

4. Faceţi click in elementul clasa **Checking** (subclasa), apoi faceţi click în elementul clasa BankAccount (superclasa). Apare caseta de dialog ***The Select Methods to Redefine***.

5. Selectati metoda **withdraw** si faceti click pe **OK**. IDE-ul realizeaza următoarele:

Închide caseta de dialog

Adaugă metoda **withdraw** la clasa **Checking**

Adauga relatia ***Generalization*** intre cele doua clase asociate.

O legatura de generalizare arata legatura intre o subclasa si o superclasa. Subclasele sunt particularizari ale claselor, adica ele pot mosteni caracteristicile (atribute si operatii) de la superclasa.

6. Faceti click dreapta, în orice spaţiu alb din editorul de diagrama pentru a deselecta pictograma ***Generalization***.

7. Din sectiunea ***Basic*** din ***Palette***, selectati pictograma ***Nested Link*** și faceti click pe elementul **BankAccount** și apoi pe elementul *package* **bankpack**.

8. Folositi pictograma ***Nested Link*** asa cum a fost descris in pasul anterior si conectati elementele **Checking**, **AccountTest**, si **Bank** la pachetul **bankpack**.

9. Deselectaţi ***Nested Link***. O legatura *nested* indica felul cum sunt organizate elementele in pachete. In acest caz, organizati toate elemenele claselor intr-un pachet **bankpack**.

10 Apăsaţi **Ctrl+S** oriunde în editorul de diagrama pentru a salva modificările făcute in model.

1.1.6. Generarea codului sursa Java

În această secţiune, folosind facilitatea UML-ului ***Generate Code*** veti genera codul sursa Java pentru modelul UML pe care l-aţi creat în secţiunile anterioare.

1. În fereastra ***Projects*** faceţi click dreapta pe nodul **UMLPrj** şi alegeţi ***Generate Code*** din meniul pop-up.Apare caseta de dialog ***Generate Code*** si specifica proiectul tinta .

2. Acceptati setarile implicite in caseta de dialog ***Generate Code***.

3. Faceţi click pe ***OK***. IDE-ul generează codul si fereastra de ieşire afişează progresul procesului de generare a codului.

1.1.7. Continuarea dezvoltarii folosind ingineria inversa (*reverse engineering*)

În această secţiune veţi continua cu dezvoltarea aplicatiei modificand codul sursa generat in editorul sursa şi veti folosi facilitatea de ingineria inversa pentru a vă actualiza modelul UML din aplicatia dumneavoastră.

1. In fereastra ***Projects***, expandati **UMLPrj** > ***Model*** > **bankpack**.

2. Faceţi clic dreapta pe nodul **BankAccount** si alegeti ***Navigate To Source*** din meniul de pop-up.

3. Adăugaţi următorul cod la metoda **deposit** in **Source Editor** (editorul sursei):

```
setBalance(getBalance() + amount);
```

4. In fereastra ***Projects*** sub **UMLPrj** > ***Model*** > ***bankpack*** faceţi click dreapta pe nodul **AccountTest** si alegeti ***Navigate To Source*** din meniul de pop-up.

5. Introduceti (sau *copy and paste*) codul următor in **Source Editor**:

```
public static void main(String[] args) {
    Checking myChecking = new Checking();
    myChecking.deposit(100);
    System.out.println("Checking Balance is: " +
            myChecking.getBalance() );
}
```

Acest cod creează un nou obiect **Checking** si specifica un depozit de **100$**, apoi afiseaza rezultatele.

6. Click dreapta in ***Source Editor*** şi alegeti ***Format Code***.

7. Apăsaţi **Ctrl+S** oriunde în ***Source Editor*** pentru a salva modificările făcute in fişierul sursă **AccountTest.java**.

8. Din nou click dreapta in ***Source Editor*** şi alegeţi ***Reverse Engineer*** din meniul pop-up. Apare caseta de dialog ***Reverse Engineer***.

9. Selectaţi ***Use Existing UML Project*** in caseta de dialog ***Reverse Engineer*** si alegeti **UMLPrj** ca proiect destinatie.

10. Faceţi click pe **OK** pentru a porni procesul de inginerie inversă.

11. Apare casuta de dialog ***Model Element Overwrite Authorization***, solicitand confirmarea suprascrierii elementului model **AccountTest**. Faceţi click pe **Yes / Yes to All**.

12. Faceţi click pe tab-ul **BankClassDiagram**. Observaţi că metoda main recent introdusa apare acum intre elementele clasei **AccountTest** în diagrama de clasa. Folosind caracteristica de inginerie inversa, modificarile facute in codul sursa al proiectului Java se reflecta in proiectul corespunzator modelului UML.

1.1.8. Testarea aplicatiei

Acum creati şi rulati proiectul dumneavoastra.

1. În fereastra ***Projects*** faceţi click dreapta pe nodul **JavaPrj** si alegeti ***Build*** din meniul de pop-up.

2. În fereastra ***Projects*** faceţi click dreapta pe nodul **AccountTest** si alegeti ***Run File*** din meniul de pop-up.

IDE-ul execută aplicatia si afiseaza urmatoarele iesiri in fereastra de iesire:

```
Checking Balance is: 100
```

Acum ati terminat aplicatia.

1.1.9. Sumar

Pana acum aţi proiectat o diagramă de clasă pentru o simplă aplicatie bancara. Aţi învăţat cum să efectuaţi următoarele activităţi:

Sa **creaţi un proiect UML.**

Sa **utilizaţi pictogramele UML** din ***Palette*** pentru a crea clase, interfeţe, pachete (incapsulari), atribute (campuri), şi operaţii (metode).

Sa **puneti clasele in relatie** cu ajutorul asocierilor UML.

Sa **vizualizati elementele** pe care le-ati creat in editorul de diagrame in proiectul UML asa cum sunt reprezentate in fereastra ***Projects***.

Sa **generaţi codul sursă** pentru elementele create în editorul de diagrame din proiectul UML şi sa **vizualizati sursele generate** în editorul surselor.

Să **folosiţi codul generat** şi **facilitatea de ingineria inversa (*reverse engineering*)** pentru a merge înainte şi înapoi (***round-trip engineering***) între modelarea şi dezvoltarea codului în editorul surselor.

Să **compilaţi şi să executaţi clase** din editorul surselor.

1.2. Ingineria inversă a aplicaţiilor Java folosind facilitatile UML NetBeans

În acest material invatati **cum se inverseaza codul sursă al unei aplicaţii Java existente** într-un proiect UML. În proiectul UML toate clasele si obiectele aplicatiei sunt reprezentate în fereastra ***Project*** de sub nodul proiect UML, în timp ce codul original Java rămâne în starea sa iniţială.

1.2.1. Crearea proiectelor demonstrative Java si UML

In aceasta sectiune veti folosi asistentul (*wizard*-ul) ***New Project*** pentru a crea proiectul demonstrativ Java si proiectul demonstrativ UML oferite de **IDE-ul NetBaens**.

1. Din meniul principal al IDE-ului, alegeţi ***File*** > ***New Project***. Apare asistentul ***New Project***.

2. În pagina ***Choose Project***, în panoul ***Categories*** expandaţi nodul ***Samples*** si selectati nodul ***UML***. Panoul ***Project*** este actualizat cu proiectele UML disponibile.

3. În panoul ***Projects***, selectaţi **UML Bank App Sample** şi faceţi click pe ***Next***.

4. În pagina ***Name and Location***, lasati valoarea implicită **UMLBankAppSample** pentru nume proiectului Java.

5. Pentru campul ***Project Location***, click pe ***Browse*** pentru a naviga la folderul in care doriti sa salvati fisierele pentru proiectele demonstrative. De mentionat faptul ca atunci cand schimbati valoarea campului ***Projects Location***, IDE-ul completeaza automat valorile din campurile ***Java Proejct Folder*** si ***UML Project Folder***.

6. Lăsaţi valoarea implicită **UMLBankAppSample-Model** pentru numele proiectului UML.

7. Faceti click pe ***Finish***. Apare caseta de dialog ***Opening Projects***. In fereastra ***Project*** apar proiectele: **UMLBankAppSample** si **UMLBankAppSample-Model**.

1.2.2. Explorarea proiectului UML în fereastra Projects

Aceasta sectiune va ghideaza prin diferitele parti ale aplicatiei **UMLBankAppSample** prin explorarea diferitelor facilitati ale ferestei ***Projects*** care cuprinde elementele aplicatiei.

1. Din meniul principal, selectati ***Windows*** > ***Properties*** pentru a deschide fereastra Properties.

2. În fereastra ***Projects*** expandati nodul **UMLBankAppSample-Model** > ***Model*** > **bankpack**.

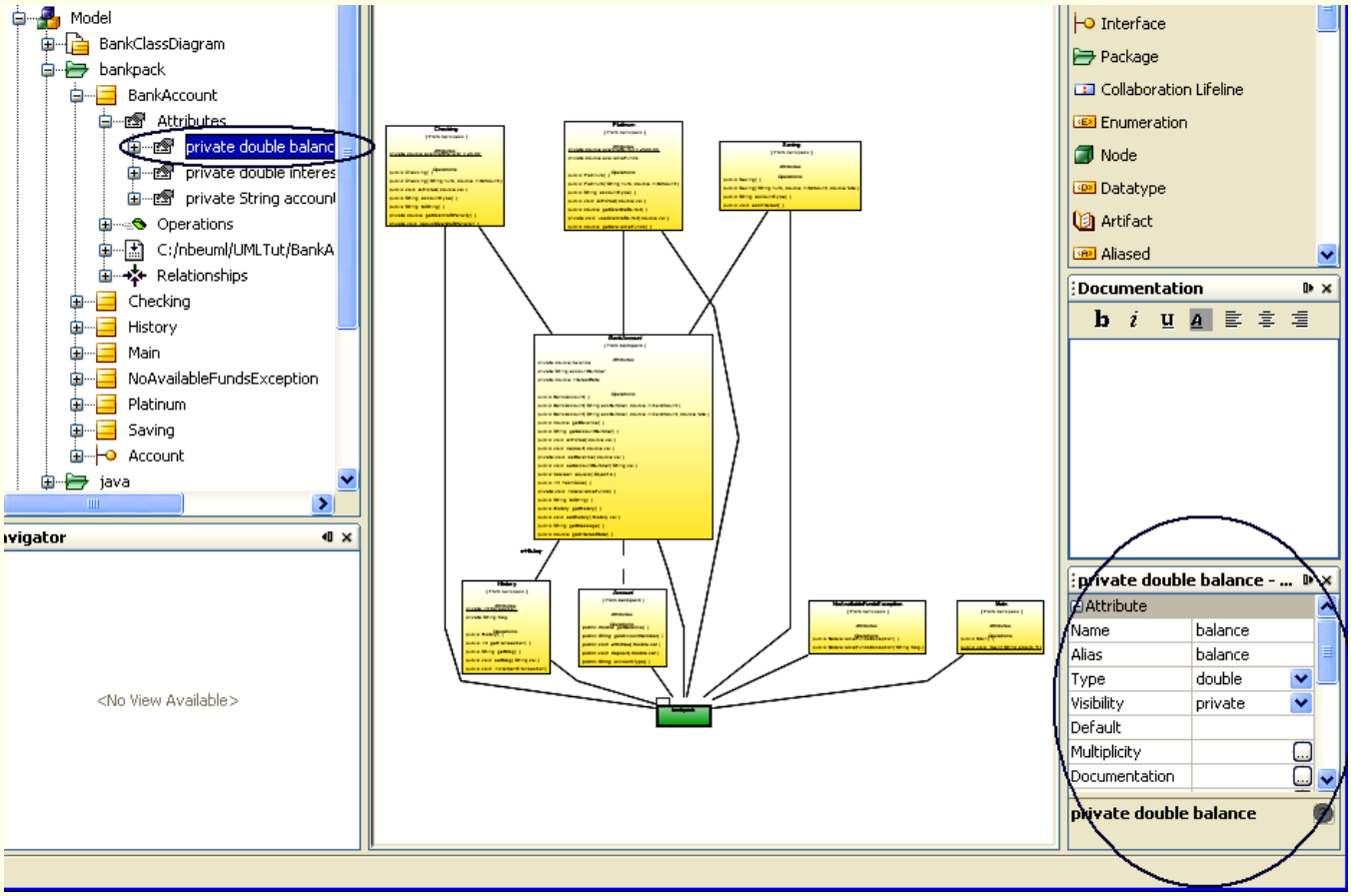
3. Expandaţi nodul **BankAccount** si nodul ***Attributes***. Toate atributele clasei **BankAccount** sunt in acest director.

Dacă există mai puţin de trei atribute, acestea apar sub nodul clasei fara un director ***Attributes***.

4. Selectati nodul ***Attribute*** si nu numele **private double balance**. Fereastra ***Properties*** prezinta numele atributului şi proprietăţile lui.

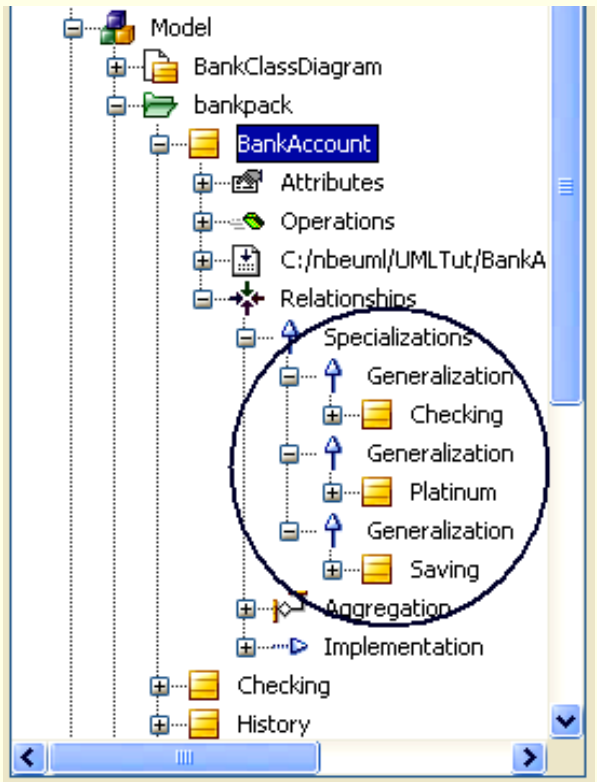
5. In fereastra ***Projects*** window, sub nodul clasei **BankAccount**, expandati nodul ***Operations***. Toate metodele clasei **BankAccount** apar sub acest nod.

6. Sub clasa **BankAccount** expandati nodul ***Relationships*** (relatii). Sunt afişate noduri pentru trei tipuri de relaţii: ***Specializations***, ***Aggregation***, si ***Implementation***.



7. Exindeti nodul ***Specializations*** , apoi expandati cele trei noduri ***Generalization***. Link-urile ***Generalization*** indica relatiile dintre clasa **BankAccount** si alte elemente (clasele **Checking**, **Platinum**, si **Saving**).

8. (Opţional) Înainte de a trece la secţiunea următoare, inchideti nodurile expandate sub clasa **BankAccount**.



1.2.3. Crearea altui proiect UML prin inginerie inversă

În prima parte a acestui material aţi încărcat proiectul Java numit **UMLBankAppSample** si proiectul UML numit **UMLBankAppSample-Model** care a fost anterior generat folosind facilitatea de inginerie inversa. În această secţiune veti face toti pasi pentru a crea un alt proiect UML, folosind facilitatea de inginerie inversa pornind de la acelaşi proiect Java **UMLBankAppSample** pe care l-aţi folosit în secţiunea anterioară. Apoi veti reveni la proiectul demonstrativ.

1. In fereastra ***Projects***, faceţi click dreapta pe nodul **UMLBankAppSample** si alegeti ***Reverse Engineer.*** Apare caseta de dialog ***Reverse Engineer***. Observaţi că fila ***Selected Nodes*** contine valori ne-editabile pentru **UMLBankAppSample**.

2. Lasati casutele din coloana ***Reverse Engineer*** selectate.

3. Selectaţi butonul ***Create New UML Project***.

4. In campul ***Project Name*** acceptati valoarea implicita **UMLBankAppSample-Model1**.

5. Pentru fila ***Project Location*** lăsaţi valoarea implicită, care este directorul in care ati salvat proiectul UML. Dacă doriţi să utilizaţi un alt folder, faceţi click pe ***Browse*** pentru a naviga la alt director si faceţi click pe ***Open***.

6. Faceti click pe ***OK***. Apare caseta de progres ***Opening***. In fereastra ***Projects*** apare proiectul **UMLBankAppSample-Model1**.

1.2.4. Generarea unei diagrame de clase

În această secţiune veti genera diagrame de clase pentru aplicatia **UMLBankAppSample** pe care ati importat-o în secţiunea anterioară.

1. In fereastra ***Projects***, sub nodul **UMLBankAppSample-Model** expandati nodul ***Model*** si expandati nodul **bankpack**.

2. Selectati directorul **bankpack** si toate elementele de sub directorul **bankpack** prin apasarea tastei ***Shift*** sau a tastei ***Ctrl*** atunci când faceti selecţia.

3. Faceţi click dreapta pe elementele selectate şi alegeţi ***Create Diagram From Selected Elements*** din meniul de pop-up. Apare asistentul ***Create New Diagram***.

4. In lista ***Diagram Type*** selectati ***Class Diagram***.

5. Scrieti **BankClassDiagram** in campul ***Diagram Name***, lasati **UMLBankAppSample-Model** in campul ***Namespace*** si faceti click pe ***Finish***. IDE-ul realizeaza urmatoarele:

   Sub nodul ***Model*** creează un nod **BankClassDiagram**

   In editorul de diagrama se afiseaza noua diagrama

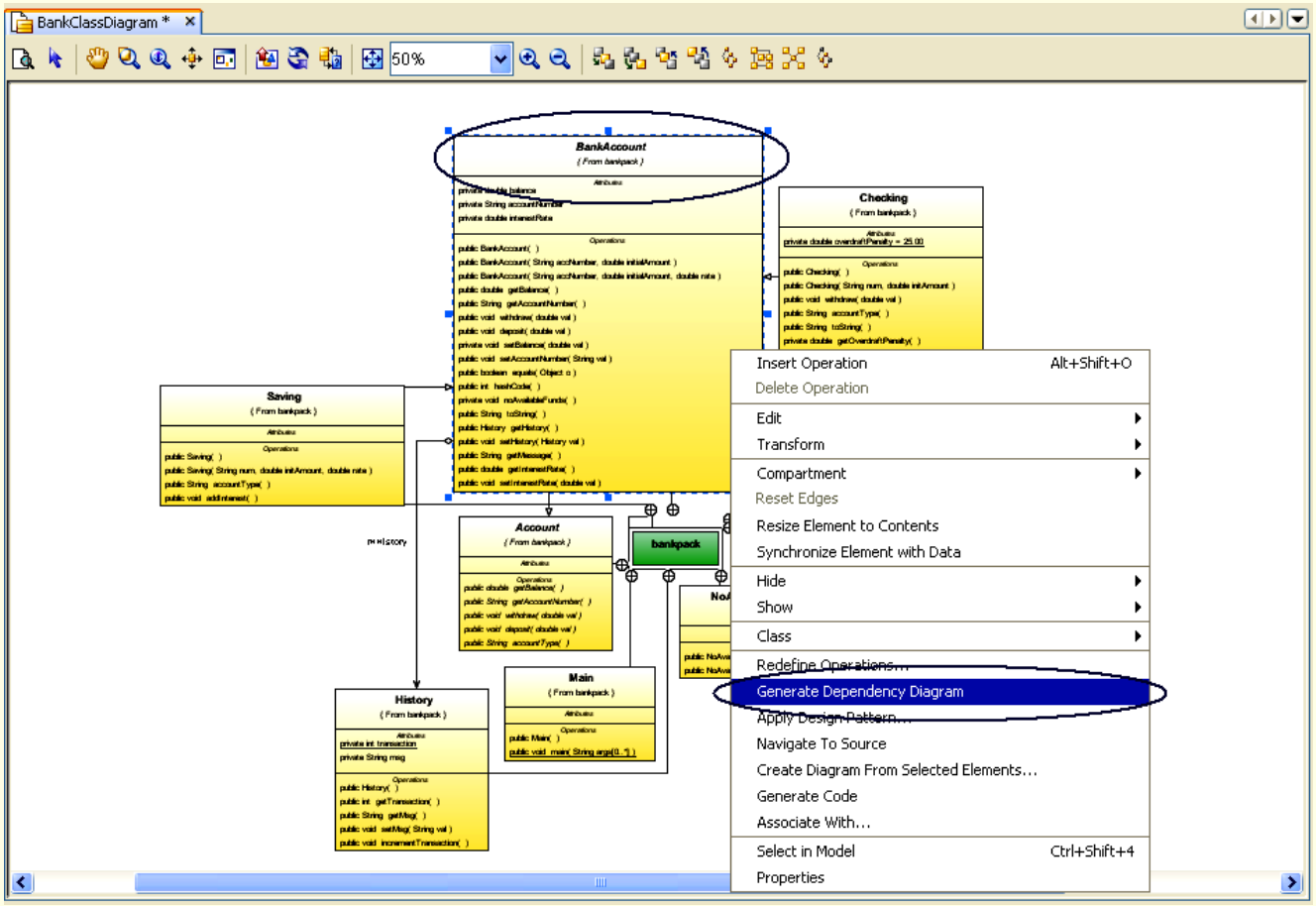
   Deschide ***Palette***

1.2.5. Generarea unei diagrame de dependente pentru o clasa data

Acum generati o diagrama de dependente pentru o una din clasele aplicatiei **BankApp**.

1. Faceti *double click* pe fila **BankClassDiagram** pentru a comuta la vizualizarea completa a figurii.

2. Faceti click dreapta pe clasa **BankAccount** in editorul de diagrame pentru a alege ***Generate Dependency Diagram*** din meniul de pop-up.



IDE-ul creeaza o diagrama de dependente si deschide fila **BankAccountDependencies** in editorul de diagrame pentru a vizualiza noua diagrama. Acum va concentrati pe fila din editorul de diagrame care arata diagrama **BankAccountDependencies**.

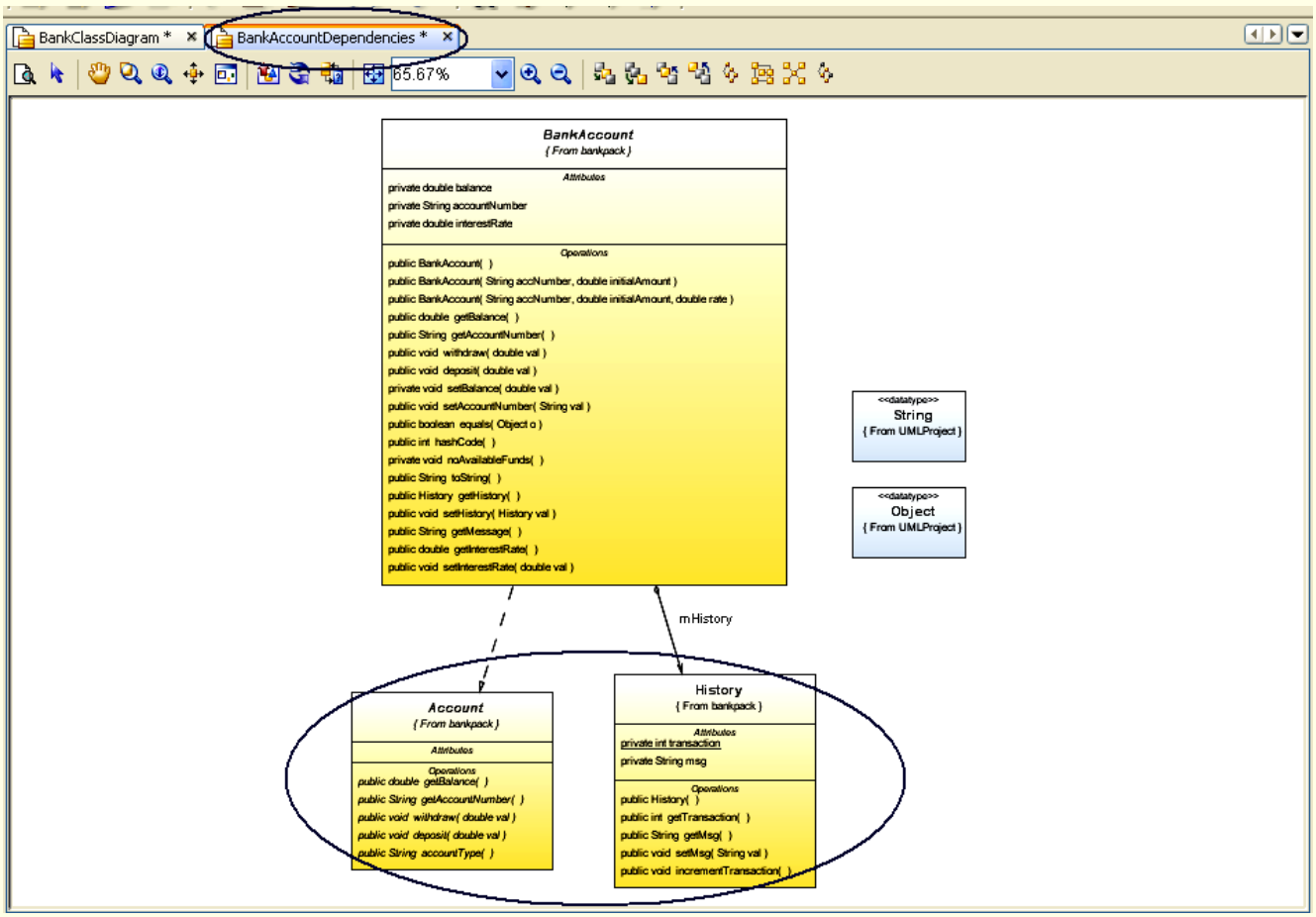


Diagrama **BankAccountDependencies** arata urmatoarele legaturile de dependenta:

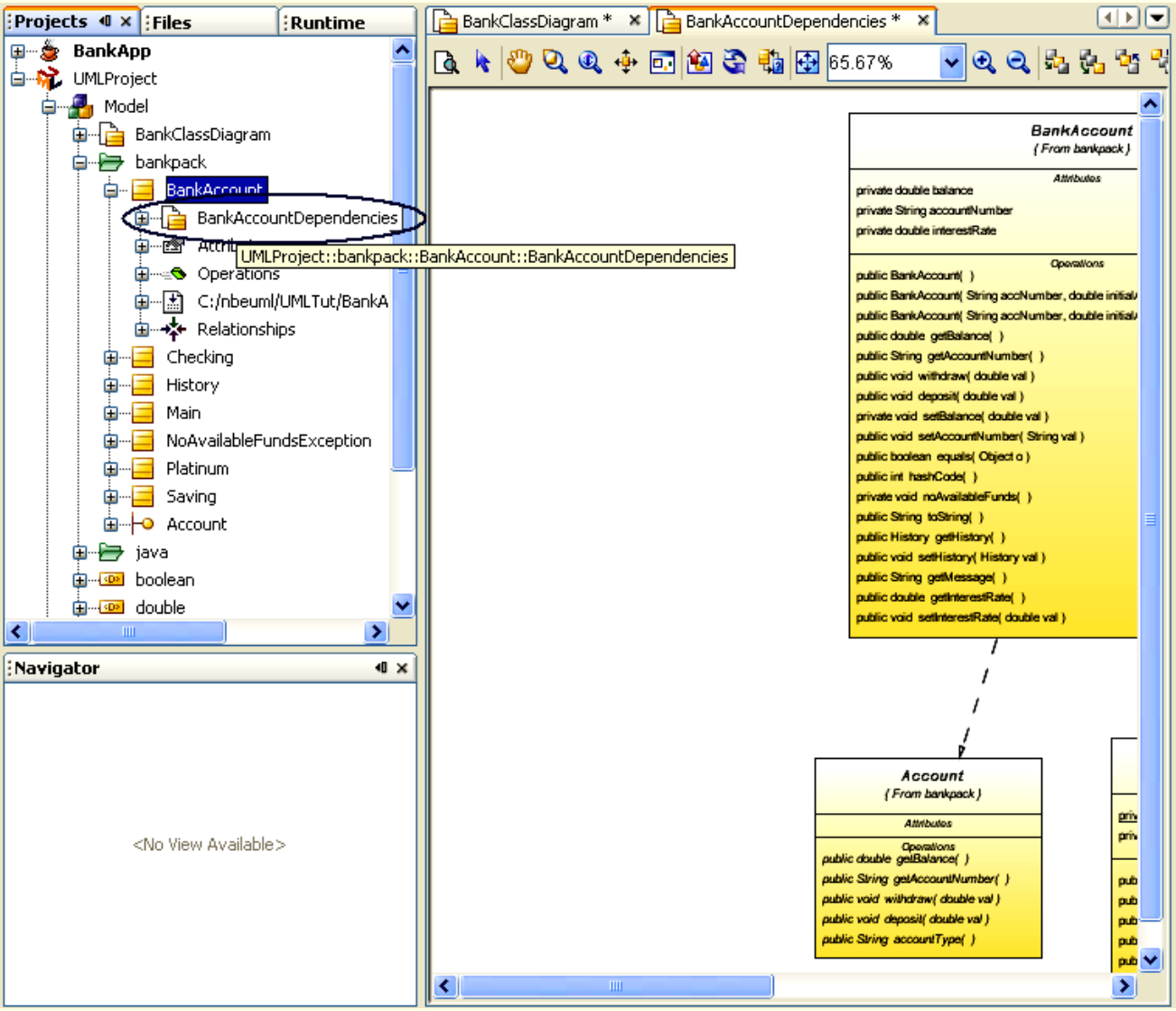
Relatia de implementare ***Implementation*** cu interfata **Account**

Relatia de agregare navigabila (***Navigable Aggregation***) catre clasa **History**

Observatie: Folosind aceasta optiune , ati creat o diagrama care arata toate dependentele pentru orice obiect dat.

3. Faceti double click pe fila **BankClassDiagram** pentru a comuta la vizualizarea multi-fereastra.

4. In ferestra ***Projects***, expandati nodul clasei **BankAccount** sub directorul ***UMLProject*** > ***Model*** > **bankpack**. Veti vedea ca un nod **BankAccountDependencies** a fost adaugat, reprezentand diagrama de dependente pe care ati creat-o.



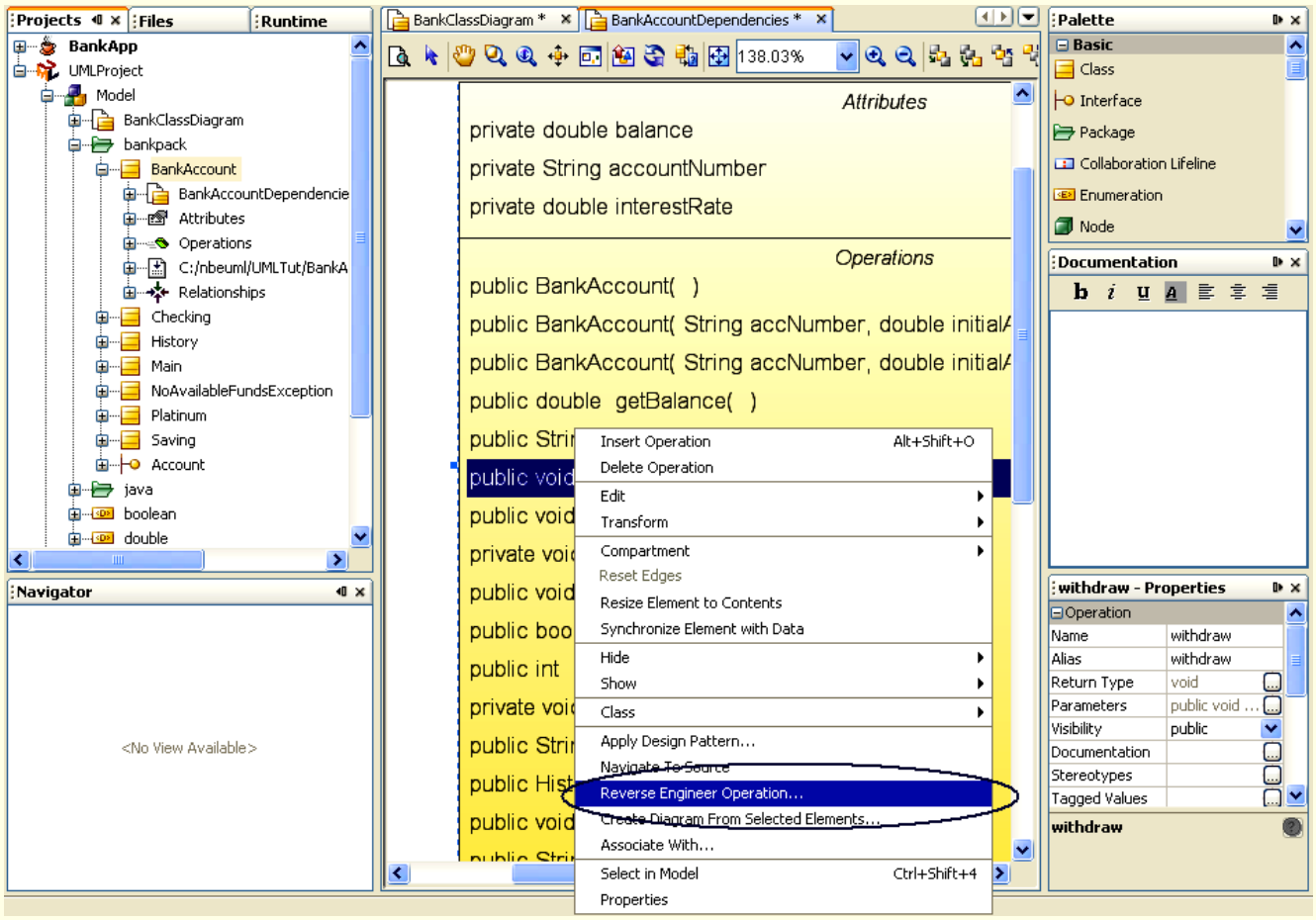
1.2.6. Generarea unei diagrame de secventa pentru o operatie

Acum generati o diagrama de secventa pentru operatia **withdraw** a aplicatiei. Veti observa cum este reprezentata ca diagrama si ca elemente in proiectul UML din fereastra ***Projects***

1. Apasati click pe tab-ul **BankAccountDependencies** din editorul de diagrame.

2. Selectati elementul **BankAccount** si ajustati zoom-ul pentru a citi usor etichetele operatiilor.

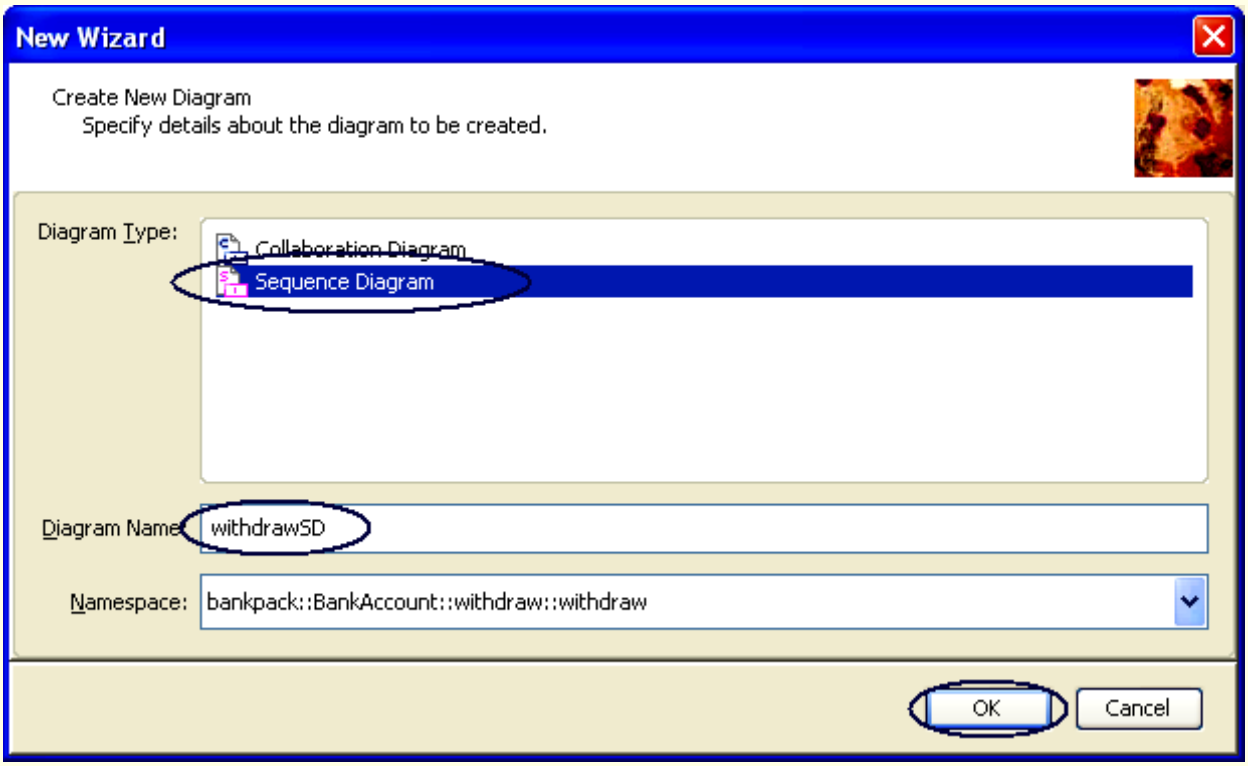
3. In editorul de diagrama, apasati click dreapta pe operatia **withdraw**, si alegeti operatia de inginerie inversa din meniul pop-up. Apare fereastra de dialog ***Create New Diagram***.



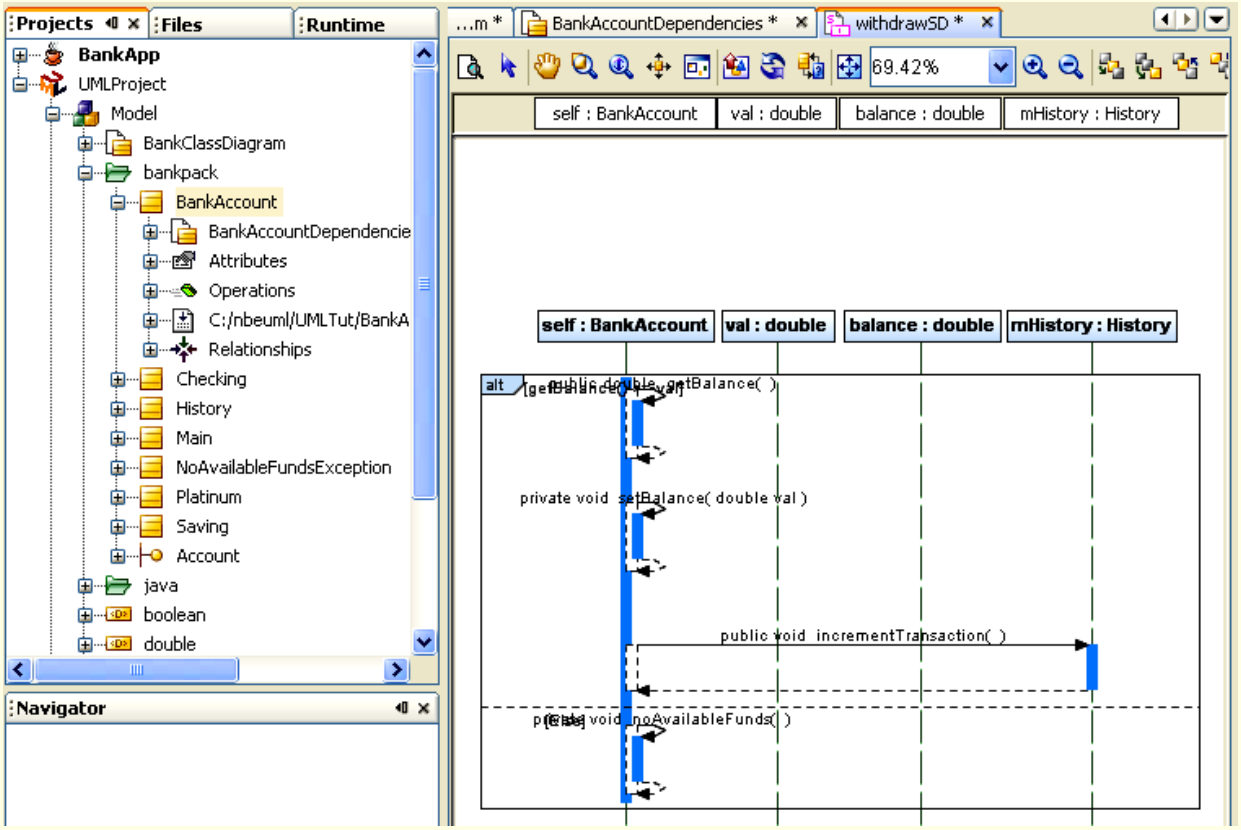
4. In lista ***Diagram Type***, selectati ***Sequence Diagram***.

5. In campul ***Diagram Name***, scrieti **withdrawSD**.

6. Acceptati valoarea standard in campul ***Namespace*** si dati click pe ***Finish***. O diagrama de secventa va aparea in editorul de diagrame.

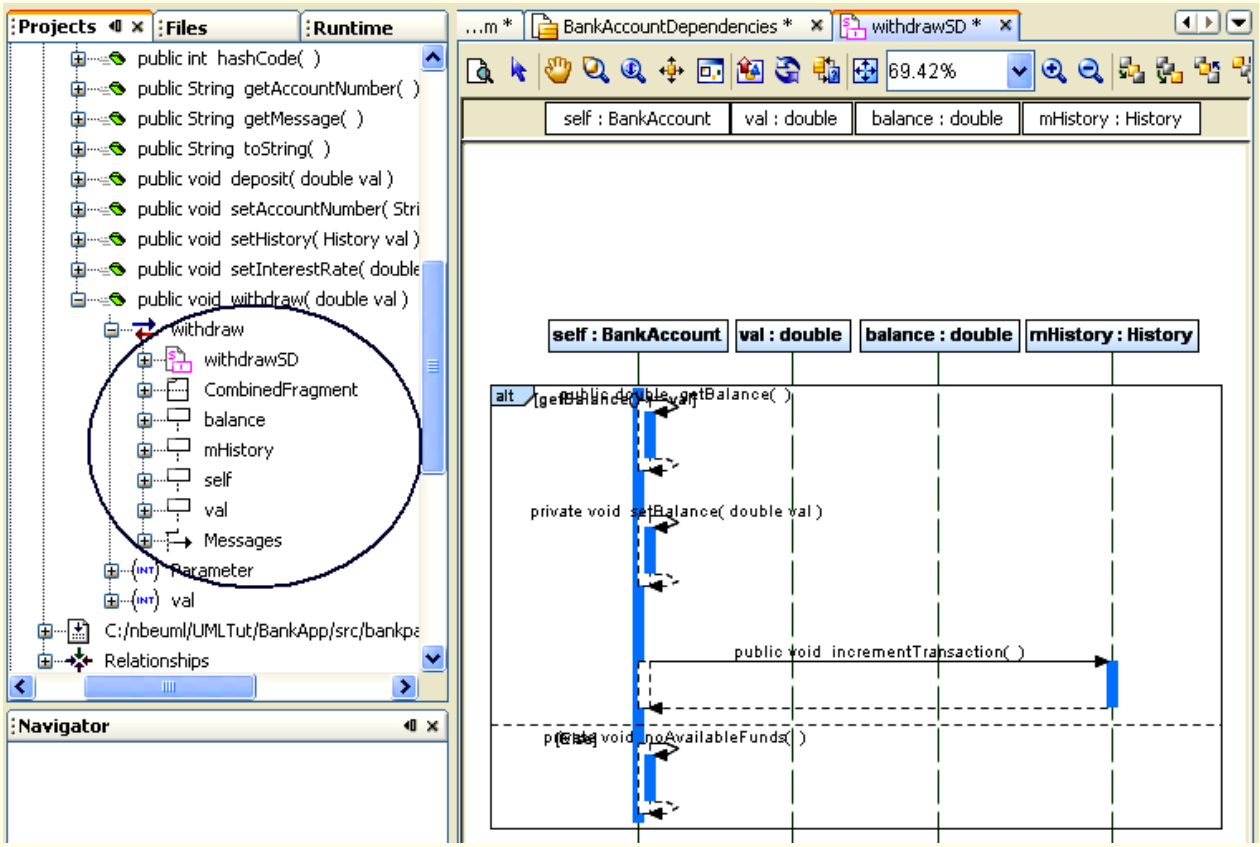


7. Expandati editorul de diagrama si manipulati nivelul de zoom astfel incat sa se poata examina cu usurinta noua diagrama de secventa. Diagrama de secventa arata fluxul de control, secventa comportamentelor, si concurenta proceselor si a activarilor.



8. In ferestra ***Projects***, sub nodul clasei **BankAccount**, expandati nodul ***Operations***, apoi expandati nodul operatiei **public void withdraw**. Vedeti nodul diagramei de secventa **withdraw** .

9. Expandati nodul **withdraw** pentru a vedea elementele diagramei de secventa reprezentate in editorul de diagrama.



1.2.7. Sumar

In acest material ati invatat sa efectuati inginerie inversa asupra unei aplicatii Java, prin importarea informatiei acesteia intr-un model UML. Ati invatat sa indepliniti urmatoarele sarcini:

Sa **generati o diagrama de clasa din aplicatia Java** importata

Sa **explorati caracteristicile ferestrei *Project*** folosita la reprezentarea elementelor aplicatiei.

Sa **folositi butoanele barei de instrumente *Diagram***.

Sa **folositi fereastra *Overview*** ca un instrument de vizualizare.

Sa **generati o diagrama de dependenta pentru una din clasele aplicatiei**.

Sa **generati o diagrama de secventa pentru o operatie**.

1.3. Crearea diagramelor UML ale cazurilor de utilizare in NetBeans 6

In continuare veti invata cum sa folositi **caracteristicile UML ale IDE-ului NetBeans** pentru a crea **o diagrama UML a cazurilor de utilizare** (*Use Case*) simpla. Folosind modelul diagramei Use Case, veti arata **relatia dintre actori** (entitatile externe) **si cazurile de utilizare in cadrul unei aplicatii**. Diagrama Use Case pe care o creati urmareste **diferitele functii si entitatile care interactioneaza cu acele functii in cadrul unei aplicatii** bancare teoretice.

O diagrama Use Case este folositoare atunci cand descrieti cerintele unui sistem in stadiile de analiza, proiectare (*design*), implementare si documentare. Scopul acestui tutorial este de a prezenta **diagramele UML ale cazurilor de utilizare in IDE-ul NetBeans**, fara a insista pe conceptele UML sau limbajul de programare Java.

1.3.1. Crearea proiectului UML si a diagramelor cazurilor de utilizare

În această secţiune, veti crea proiectul UML si diagrama cazurilor de utilizare pentru aplicatie.

1. Creati un nou director numit **UMLTutorial** pe drive-ul **D:** in directorul **\isw**, in **subdirectorul cu numarul grupei**, (de exemplu: **D:\isw\441E\UMLTutorial**).

2. Din meniul principal, selectaţi ***File*> *New Project*** şi apoi faceţi următoarele în ***New Project Wizard***:

   Sub ***Categories***, selectati ***UML***.

   Sub ***Projects***, selectati ***Java-Platform Model***.

   Faceti ***Click*** pe ***Next***.

   Se va deschide caseta de dialog ***New Java-Platform Model***.

3. In campul ***Project Name***, scrieti **UMLTutorialProject**.

Observati ca atunci cand scrieti ***Project Name***, IDE-ul va sugereaza automat acest nume.

4. Pentru campul ***Project Location***, faceti click pe ***Browse***.

5. In casuta de dialog a ***Select Project Location***, selectati **UMLTutorial**, care este directorul pe care l-ati creat in pasul 1.

6. Faceti click pe ***Open*** pentru a elimina casuta de dialog.

7. In pagina ***Name and Location***, click ***Finish***.

IDE-ul creaza proiectul UML si se deschide ***New Wizard*** si afiseaza casuta de dialog ***Create New Diagram***.

8. In lista ***Diagram Type***, selectati ***Use Case Diagram***.

9. In campul ***Diagram Name***, scrieti **UseCaseDiagram**.

10. Lasati **UMLTutorialProject** in campul ***Namespace*** si click ***Finish***. IDE-ul face urmatoarele:

    Creaza nodul **UseCaseDiagram** sub nodul ***Model***

    Afiseaza noua diagrama in editorul diagramei (diagrama este goala in acest moment)

    Deschide *(Modeling)* ***Palette***

1.3.2. Adăugarea şi etichetarea elementelor Use Case (caz de utilizare)

In aceasta sectiune veti adauga elementele ***Use Case*** (caz de utilizare) folosind ***Palette*** in IDE-ul NetBeans.

1. Din sectiunea ***Basic*** a ferestrei ***Palette***, selectati pictograma ***Use Case*** si faceti click o singura data in partea de sus stanga a editorului de diagrame.

Acesta actiune plaseaza un element ***Use Case*** in diagrama.

2. Deselectati pictograma facand click-drepta oriunde in editorul de diagrame sau apasand tasta ***ESC***.

3. Daca nu este deja selectat, selectati noul element adaugat facand click pe el o data.

4. Scrieti **Withdraw Money** si apasati ***Enter***.

Acesta eticheteaza elementul cu textul **Withdraw Money**.

5. Selectati pictograma ***Use Case*** din nou si plasati inca **7** elemente ***Use Case*** in diagrama. Plasati elementele in patru randuri continand cate doua elemente in fiecare rand.

6. Deselectati pictograma facand click-drepta oriunde in editorul de diagrame.

7. Selectati elementul ***Use Case*** aflat sub **Withdraw Money**.

8. Scrieti **Withdraw Cash from ATM** si apasati ***Enter***.

9. Etichetati elementele ***Use Case*** ramase dupa cum urmeaza:

   Deposit Money

   Process a Loan

   Apply for Loan

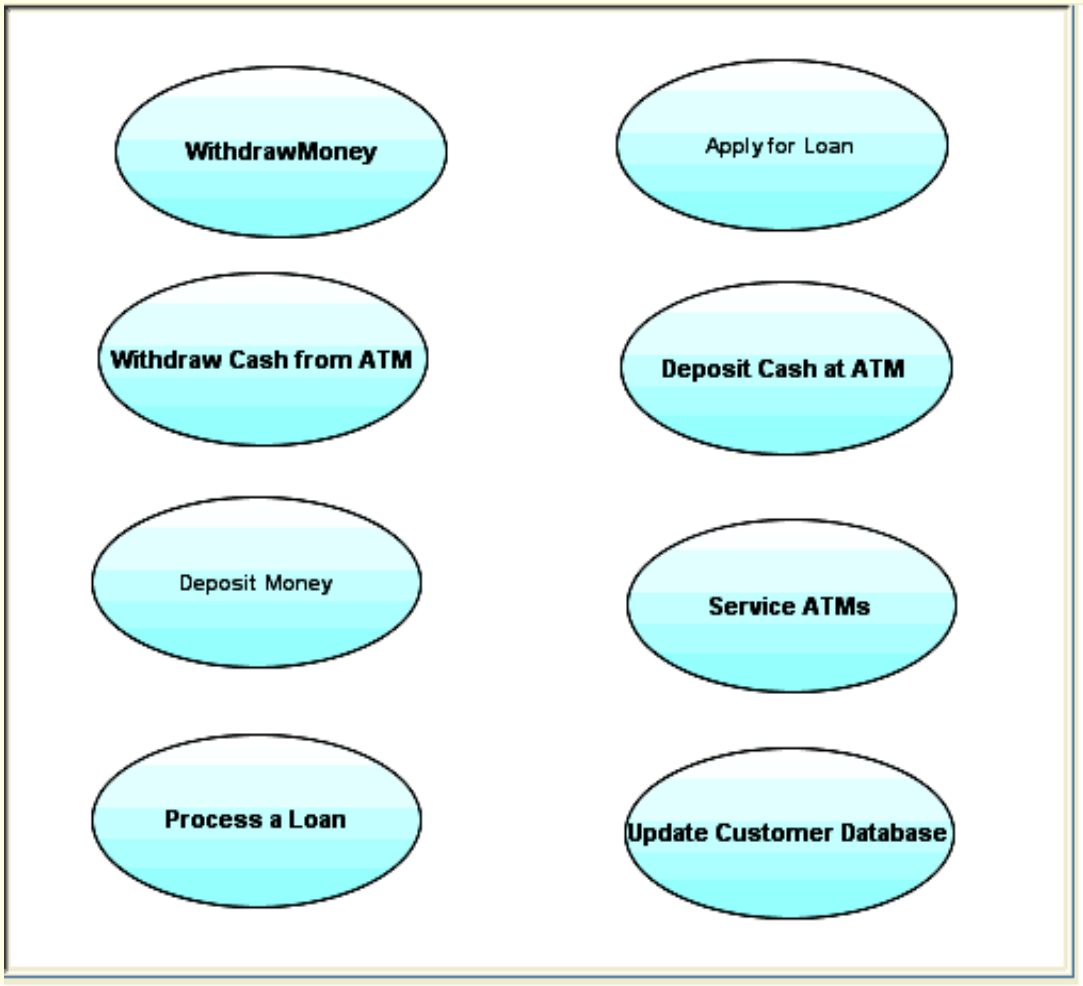
   Deposit Cash at ATM

   Service ATMs

   Update Customer Database

Observatie: Cand ati adaugat si etichetat elementele diagramei, le puteti redimensiona dupa nevoie facand click-dreapta pe element si selectand ***Resize to Element to Contents*** din pop-up menu.

Diagrama ar trebuie sa se asemene cu figura urmatoare.



1.3.3. Adăugarea şi etichetarea elementelor Actor

Acum veti adauga elementele cazurilor de utilizare folosind ***Palette*** in IDE-ul NetBeans.

1. Din sectiunea ***Basic*** a ferestrei ***Palette***, selectati pictograma ***Actor***.

2. Faceti click o data in dreapta elementului caz de utilizare **Apply for a Loan** pentru a plasa elementul ***Actor*** in diagrama. Un element ***Actor*** nedenumit este plasat in editorul de diagrame.

3. Faceti click pe ***ESC*** pentru a deselecta pictograma.

4. Selectati elementul ***Actor*** pe care tocmai l-ati plast in diagrama, scrieti **Customer** si apasati ***Enter***. Elementul ***Actor*** este astfel etichetat.

Observatie: Cand ati adaugat mai multe elemente diagramei, faceti click pe butonul ***Fit To Window*** din bara de instrumente (toolbar-ul) ***Diagram*** pentru a rearanja diagrama astfel incat sa puteti vedea intreaga diagrama in editorul de diagrame.

5. Plasati inca **5** elemente ***Actor*** sub actorul **Customer** in editorul de diagrame.

6. Deselectati pictograma ***Actor*** facand click oriunde in editorul de diagrame.

7. Etichetati noile adugate elmente ***Actor*** dupa cum urmeaza:

Employees

Bank Teller

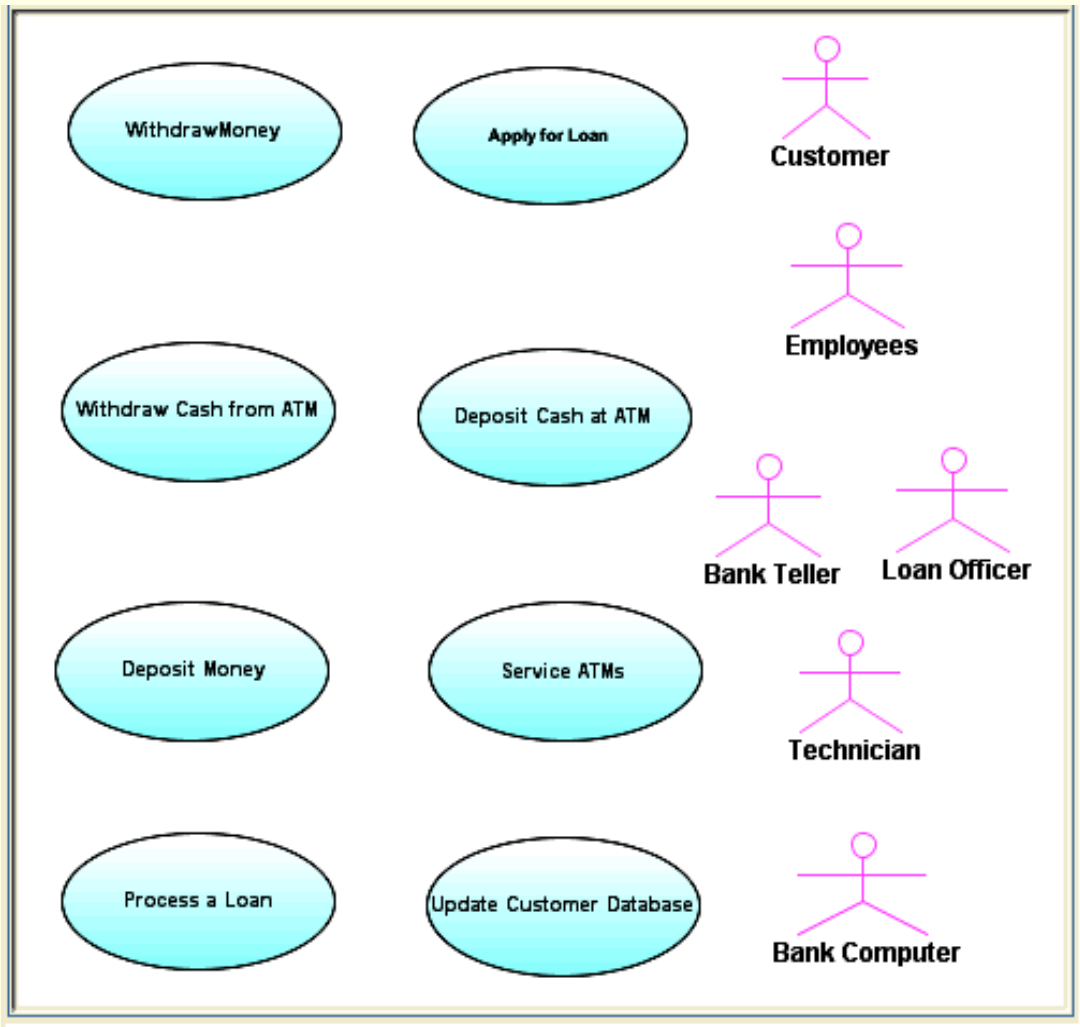
Loan Officer

Technician

Bank Computer

8. Click-dreapta pe tab-ul ***UseCaseDiagram*** si alegeti ***Save Document*** din pop-up menu.

Diagrama ar trebui sa se asemene cu figura urmatoare.



1.3.4. Legarea elementelor Actor intre ele

In aceasta sectiune veti lega elementele ***Actor*** intre ele folosind ***Generalization***.

1. Din sectiunea ***Basic*** a ferestrei ***Palette***, selectati pictograma ***Generalization***.

2. Faceti click pe elementul **Bank Teller**, apoi click pe elementul **Employees**.

Apare o legatura (*link*) intre cele doua elemente ***Actor***. Informatiile legate de relatia ***Generalization*** apar in fereastra ***Properties***.

3. Faceti click-drepta oriunde in editorul de diagrame pentru a deselecta pictograma ***Generalization***.

4. In editorul de diagrame, selectati legatura ***Generalization***.

5. In fereastra ***Properties***, faceti click pe butonul proprietatii ***Stereotypes***. Apare expertul ***Stereotypes***.

6. Faceti click pe ***Add***, apoi faceti click in campul gol ***Name*** si scrieti **implementation**.

7. Click pe ***OK***. :egatura este etichetata <<implementation>>.

8. Adaugati legaturi ***Generalization*** pentru urmatoarele:

Loan Officer to Employees

Technician to Employees

1.3.5. Legarea elementelor Actor de elementele Use Case

In aceasta sectiune veti lega elementele ***Actor*** intre ele folosind ***Association***.

1. Din sectiunea ***Basic*** a ferestrei ***Palette***, selectati pictograma ***Association***  .

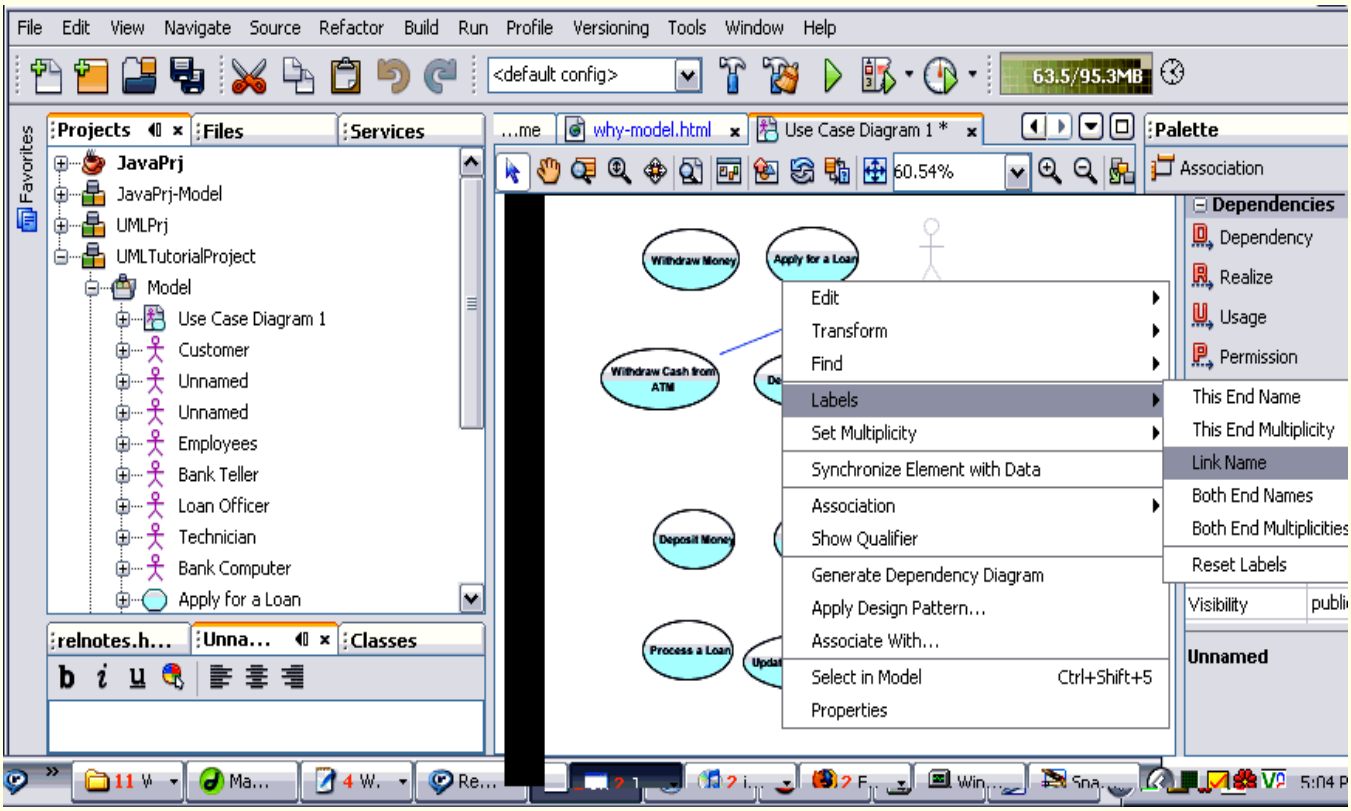
2. Faceti click pe elementul **Customer**, apoi click pe elementul **Withdraw Cash from ATM**.

Apare o legatura (*link*) intre ***Actor*** si ***Use Case***.

3. Faceti click-drepta oriunde in editorul de diagrame pentru a deselecta pictograma ***Association***.

4. Cu noua legatura ***Association*** inca selectata plasati cursorul in mijlocul liniei ce reprezinta legatura si faceti click dreapta

Nota: Cand legatura este selectata, aceasta devine albastra. Poate fi riscant sa tineti cursorul pe link. **Daca cursorul este pozitionat in spatiul alb** cand faceti click-dreapta, ati putea vedea meniul pop-up pentru editorul de diagrame mai degraba dacat meniul pop-up pentru legatura. **Daca se intampla asta incercati din nou, asigurandu-va ca legatura este albastra**, click-dreapta si ar trebui sa vedeti meniul pop-up corect pentru legatura, dupa cum se arata in figura urmatoare.



5. Alegeti ***Labels*** > ***Link Name*** din meniul pop-up. Legatura este etichetata cu textul **Unnamed**, care este evidentiat.

6. Scrieti textul **uses** si apasati ***Enter***. Legatura este etichetata acum cu textul **uses**.

7. Din sectiunea ***Basic*** a ferestrei ***Palette***, selectati pictograma ***Association*** si adaugati inca **7** legaturi, conectand ***Actori*** si ***Use Cases*** facand click pe elementul ***Actor*** intai, si apoi facand click pe elementul ***Use Case*** dupa cum urmeaza:

Customer la **Deposit Cash to ATM**

Customer la **Apply for Loan**

Bank Teller la **Withdraw Money**

Bank Teller la **Deposit Money**

Bank Computer la **Update Customer Database**

Technician la **Service ATMs**

Loan Officer la **Process a Loan**

8. Deselectati pictograma ***Association***.

1.3.6. Utilizarea legaturilor Extend (extindere)

O legatura ***Extend*** arata relatia dintre un ***Use Case*** si altul, specificand **felul in care comportamentul** definit **pentru cazul de utilizare extins poate fi inserat in comportamentul** definit **pentru cazul de utilizare de baza**.

1. Din sectiunea ***Basic*** a ferestrei ***Palette***, selectati pictograma ***Extend***  .

2. Faceti click o singura data pe elementul **Withdraw Cash From ATM**, si click din nou pe elmentul **Withdraw Money**.

O legatura etichetata <<extend>> este desenata cu o sageata ce arata spre elementul **Withdraw Money**.

3. Repetati pasii 1 si 2 pentru a desena legaturi ***Extend*** intre urmatoarele ***Use Cases***:

Deposit Cash at ATM > **Deposit Money**

Process a Loan > **Apply for Loan**

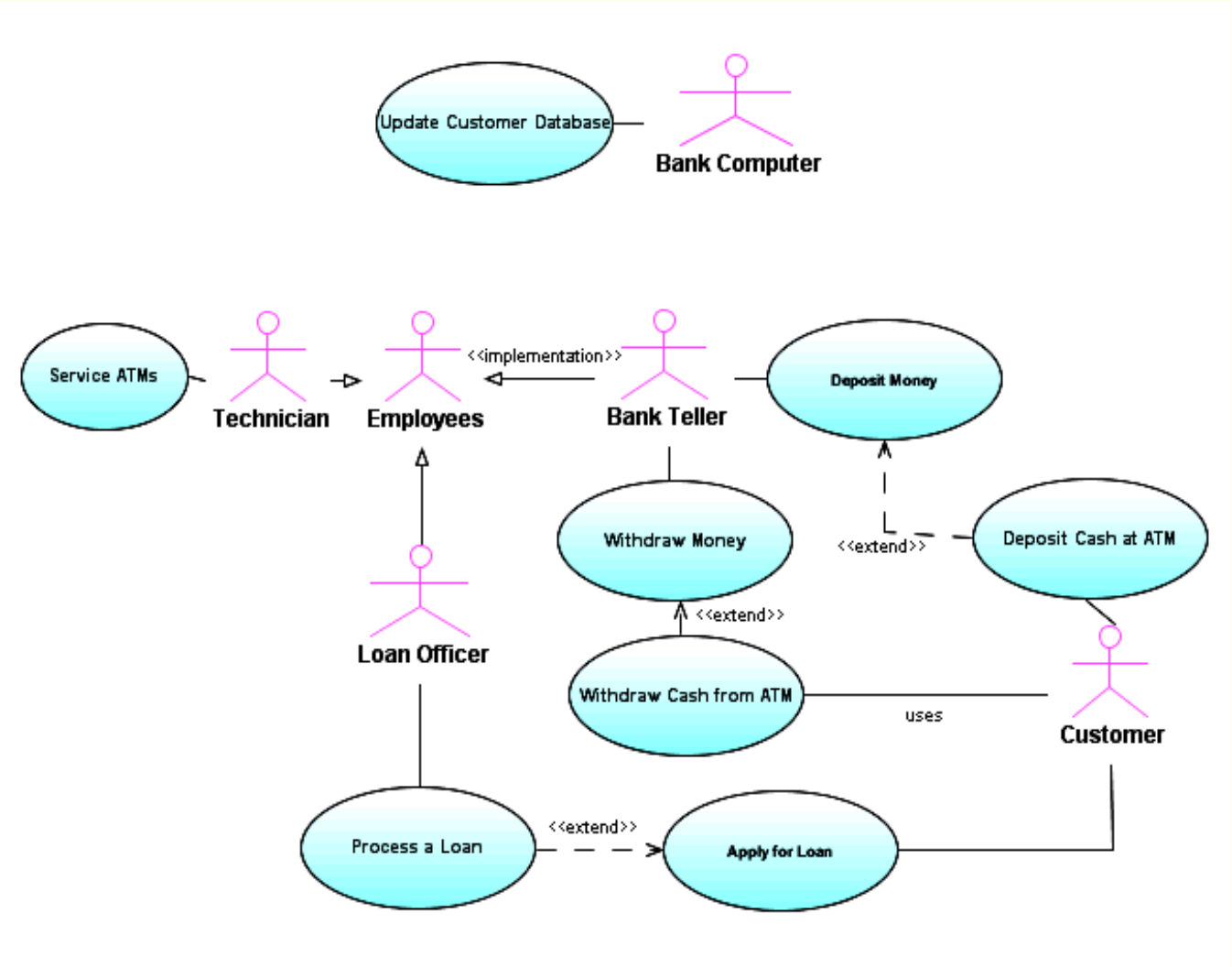
4. Click-drepta oriunde in editorul de diagrame pentru a deselecta pictograma ***Extend Link***.

5. Pentru a rearanja diagrama, faceti click pe butonul ***Orthogonal Layout***  din bara de instrumente (toolbar-ul) ***Diagram*** si click ***Yes*** in casuta de avertisment ***Layout***.

Oservatie: Ar putea fi necesar sa extindeti diagrama pentru a vedea butonul ***Orthogonal Layout*** din toolbar. Pentru a face aceasta, faceti dublu-click pe tab-ul **UseCaseDiagram**.

Puteti deasemenea sa faceti click-dreapta in editorul de diagrame si sa alegeti ***Layout*** > ***Orthogonal*** din meniul pop-up. IDE-ul rearanjeza diagrama **UseCaseDiagram** intr-un stil dreptunghiular de asezare (*layout*).

Diagrama terminata ar trebui sa se asemene cu figura urmatoare. Diagrama poate avea insa un layout usor diferit. Atata timp cat relatiile si elementele sunt corect reflectate, orice diferenta usoara de format este normala.



6. Apăsaţi **Ctrl+S** oriunde în editorul de diagrama pentru a salva modificările făcute in model.

1.3.7. Sumar

In acest tutorial ati invatat sa creati o diagrama ***Use Case*** pentru o simpla aplicatie bancara. Ati invatat cum sa efectuaţi următoarele activităţi:

Sa **creaţi un proiect UML**

Sa **creaţi o diagrama Use Case**

Să **folosiţi pictogramele UML din Palette pentru a crea cazuri de utilizare si actori**

Să **conectati cazurile de utilizare si actorii** pentru a **arata functiile aplicatiei**

1.4. Crearea diagramelor UML de clase in NetBeans 6

În continuare veti invata cum sa folositi **caracteristicile UML ale IDE-ului NetBeans** pentru a crea **diagrame UML de clase**. Acest material va arata diferite tehnici de creare a elementelor unei diagrame de clase si cum sa generati un cod de sursa Java pentru diagrama. Scopul este de a va prezinta **cateva dintre functionalitatile IDE-ului dedicate modelarii UML cu diagrame de clase**.

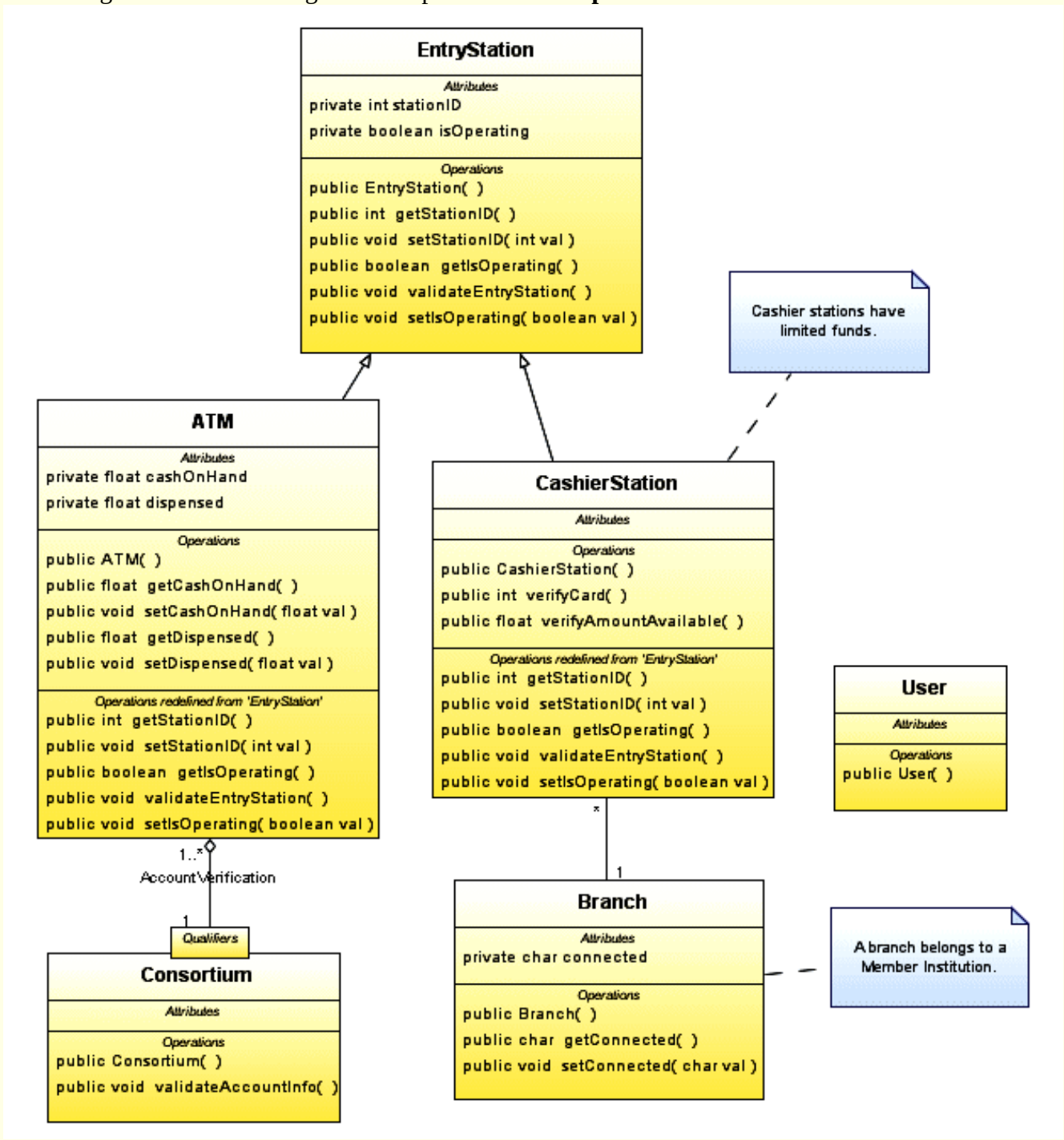
O **diagrama de clase** este o **reprezentare vizuala a unei aplicatii** care ii arata **clasele si relatiile dintre acele clase**. Cand deschideti o diagrama de clase, IDE-ul afiseaza o selectie specifica de pictograme ale elementelor UML in (*Modeling*) ***Palette***. Folosind modelul diagramei de clase, veti **descrie structura statica a elementelor din aplicatia dorita**. IDE-ul va permite sa creati grafic diagrame ce contin clase. **Clasele sunt aranjate in ierarhii** care **impart structura comuna si comportamentul** si sunt **asociate cu alte clase**.

1.4.1. Crearea diagramei de clase si adaugarea elementelor ei

Clasele definesc atributele elementelor instanta precum **si operatiile pe care fiecare element le executa sau le suporta**. Cand reprezentati o clasa intr-un model UML, puteti sa realiza urmatoarele:

Crearea elementului ce reprezinta clasa
Denumirea clasei
Definirea atributelor clasei
Definirea operatiilor clasei
Descrierea legaturilor si a asociatiilor
Adaugarea documentatiei

Sectiunile urmatoare descriu **cum se creaza o digrama de clase simpla pentru o aplicatie bancara ipotetica**. Dupa ce ati parcurs pas cu pas procedurile schitate in acest material, digrama de clase ar trebui sa arate ca in figura urmatoare. Diagrama este prezentata **doar pentru referinta**.



Folositi instructiunile incepand cu sectiunea urmatoare pentru a desena propria diagrama de clase.

1.4.1.1. Crearea diagramei de clase

În această secţiune veti crea diagrama de clase a aplicatiei.

1. Daca e necesar, porniti **NetBeans IDE** si deschideti proiectul creat anterior **UMLTutorialProject**.
2. In fereastra ***Projects***, expandati nodul **UMLTutorialProject** si apoi faceti right-click pe nodul ***Model***.
3. Selectati ***New*** > ***Diagram*** din meniul pop-up. Se va deschide ***New Wizard*** si va afisa pagina **Create New Diagram**.
4. In lista ***Diagram Type***, selectati ***Class Diagram***.
5. In campul ***Diagram Name***, scrieti **ClassDiagram**.
6. Lasati setarea implicita in campul ***Namespace*** si apasati ***Finish***. **NetBeans IDE** face urmatoarele:

   Creeaza nodul **ClassDiagram** sub nodul ***Model***

   Afiseaza noua diagrama in editorul de diagrame (diagrama este goala in acest moment)

   Deschide *Modeling* ***Palette***

1.4.1.2. Adaugarea si denumirea elementelor clasei

Acum veti adauga elementele claselor folosind ***Palette*** in IDE-ul NetBeans.

1. Din sectiunea ***Basic*** a *Modeling* ***Palette***, selectati pictograma ***Class***  si faceti click in editorul de diagrame. Aceasta actiune plaseaza un element ***Class*** pe diagrama.
2. Deselectati pictograma prin right-click oriunde in editorul de diagrame.

Observatie: De fiecare data cand selectati o pictograma, puteti plasa instante multiple ale acelui element in editorul de diagrame apasand click de mai multe ori.

3. Daca nu este inca selectat, selectati noul element ***Class*** apasand o data pe el.
4. Scrieti **EntryStation** si apasati ***Enter***. NetBeans IDE face urmatoarele:

   Atribuie elementului ***Class*** denumirea **EntryStation**

   Creeaza o metoda publica numita **EntryStation()** – constructor al clasei **EntryStation**

   Afiseaza proprietatile clasei in fereastra ***Properties***

   Adauga un element ***Class*** cu numele **EntryStation** in fereastra ***Projects*** sub nodul ***Model***

1.4.1.3. Adaugarea atributelor folosind meniul pop-up

Acum veti adauga atributele clasei **EntryStation**.

1. Daca nu este selectat, selectati elementul **EntryStation** in editorul de diagrame.
2. Faceti Right-click pe cuvantul ***Attributes*** si alegeti ***Insert Attribute*** din meniul pop-up.

O linie editor se deschide şi afişează următoarele informaţii:

```
visibility type name[ranges]=initialValue{name=value}
```

3. Scrieti **stationID** si apasati ***Enter***. Un atribut numit **stationID** de tip **int** apare in clasa **EntryStation**, si urmatoarele operaţii sunt create pe clasa:

```
public int getStationID()
public void setStationID (int val)
```

Observatie: metodele **get() si set() sunt create pentru ca ati pastrat setarile implicite**.

1.4.1.4. Adaugarea operatiilor

Acum veti adauga alte operatii clasei **EntryStation**.

1. In editorul de diagrame selectati elementul clasa numit **EntryStation**.

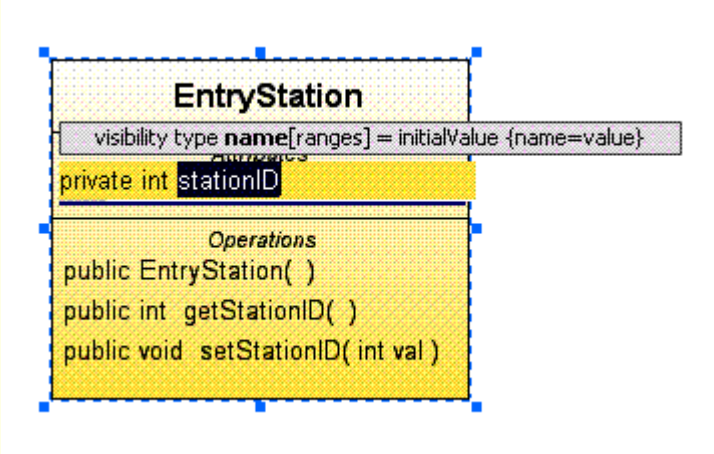
2. Right-click pe cuvantul ***Operations*** si alegeti ***Insert Operation*** din meniul pop-up. O linie editor se deschide şi afişează următoarele informaţii:

```
visibility returnType name(parameter) {properties...}
```

3. Scrieti **validateEntryStation** si apasati ***Enter***. NetBeans IDE creeaza o noua metoda.

1.4.1.5. Editarea atributelor sau a operatiilor

Cand faceti double-click pe un atribut sau o operatie a unei clase, o fereastra editor ***combo*** se deschide ca in figura urmatoare.



Pe masura ce faceti click pe fiecare parte a atributului sau operatiei, eticheta acelei parti apare evidentiata cu ***bold*** in fereastra ***combo***. Daca e posibil, partea selectata a operatiei sau atributului are o lista drop-down de valori.

De exemplu:

1. Faceti double-click pe atributul **stationID** in clasa **EntryStation**.

2. Faceti click pe cuvantul **private**. Observati faptul ca **visibility** apare evidentiat cu ***bold***.

3. Apasati ***Ctrl + down arrow***. O lista drop-down se deschide si sunt afisate valorile pe care le puteti selecta pentru atributul **visibility**.

4. Alegeti o noua valoare din lista drop-down si apasati ***Enter***. Atributul este actualizat cu noua valoare.

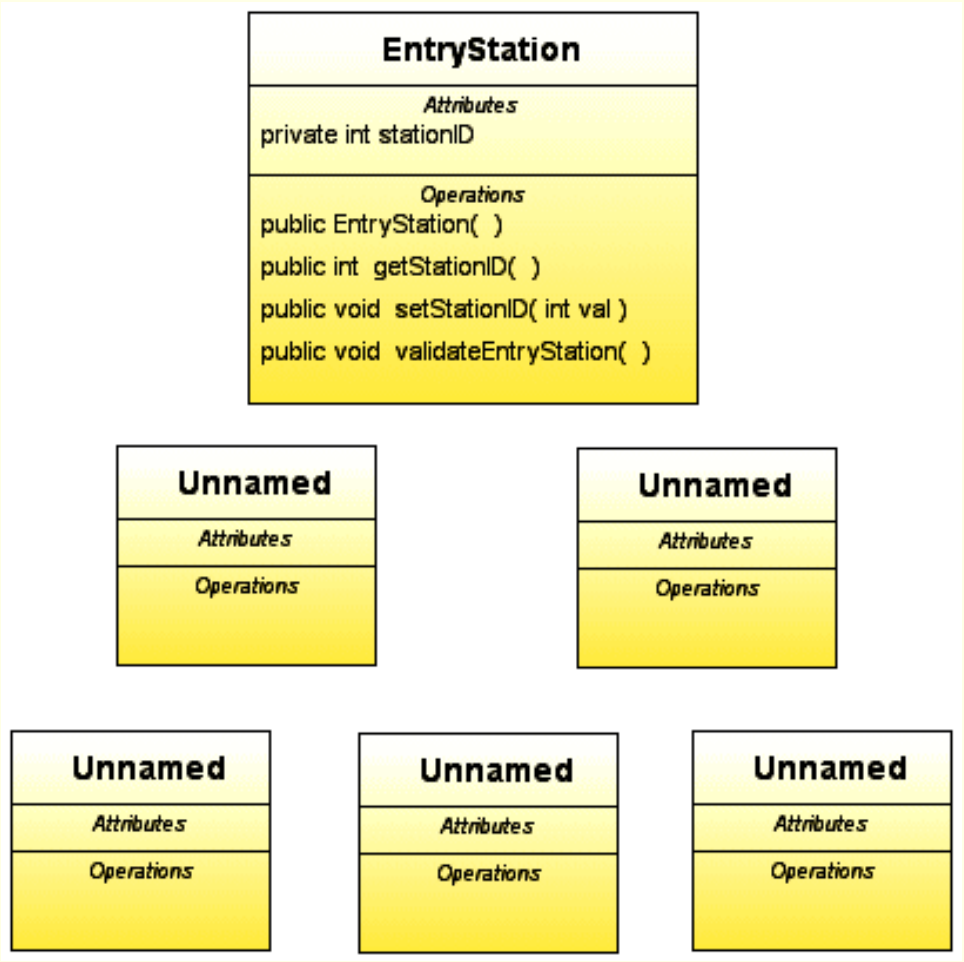
Observatie: Puteti scrie direct noua valoare in editor. Daca nu vedeti ceea ce aveti nevoie in lista drop-down, folositi tastele right si left arrow pentru a pozitiona cursorul si a scrie valoarea corespunzatoare.

5. Pentru acest material folositi valoarea **private** si apasati ***Enter*** pentru a inchide editorul.

1.4.1.6. Adaugarea operatiilor

Acum trebuie sa adaugati mai multe clase pentru a completa diagrama de clase pentru aplicatia din domeniul bancar. Dupa ce adaugati clasele, le denumiti si adaugati atribute si metode dupa cum apar mai jos folosind tehnicile pe care le-ati invatat pana acum in acest tutorial.

1. Din sectiunea ***Basic*** a *Modeling* ***Palette***, selectati pictograma ***Class*** si apasati de 5 ori in editorul de diagrame pentru a plasa elemente aditionale de tip clasa dupa cum se vede in figura de mai jos.



2. Deselectati pictograma ***Class*** printr-un right-click oriunde in editorul de diagrame.

Observatie: Puteti selecta si trage noile elemente de tip clasa pentru a le aranja asa cum apar in figura precedenta, pentru a putea fi distinse una de cealalta in mod clar.

3. Selectati primul element fara nume de tip clasa de sub elementul **EntryStation** si denumiti-l **ATM**.

4. Avand elementul clasa **ATM** inca selectat, adaugati un atribut dupa cum urmeaza:

```
private float cashOnHand
```

5. Adaugati un al doilea atribut clasei ATM si definiti-l dupa cum urmeaza:

```
private float dispensed
```

Atributele apar in diagrama de clase.

Observatie: Pe masura ce adaugati atribute si metode claselor, marimea elementelor clasa se mareste. Pentru a imbunatati aspectul diagramei mutati elementele de tip clasa astfel incat fiecare sa se vada in mod clar. Cand realizati acest lucru aveti grija sa selectati elementul ***Class***, si nu un atribut sau o metoda a acesteia.

6. Selectati primul element de tip clasa situat sub clasa **ATM** si denumiti-l **Consortium**.

7. Adaugati o metoda in clasa **Consortium**. Adaugarea metodelor este similara cu adaugarea atributelor. Right-click pe cuvantul ***Operations*** si selectati ***Insert Operations***.

8. Scrieti **validateAccountInfo** si apasati ***Enter***. NetBeans IDE creeaza o noua metoda dupa cum urmeaza:

```
public void validateAccountInfo()
```

9. Selectati elementul ***Class*** din dreapta clasei **ATM** si denumiti-l **CashierStation**.

10. Adaugati 2 metode pentru aceasta clasa dupa cum urmeaza:

```
public int verifyCard()
public float verifyAmountAvailable()
```

11. Denumiti cele 2 clase ramase **Branch** si **User**. Pentru clasa **User** nu exista atribute sau metode.

12 Pentru clasa **Branch**, adaugati un atribut dupa cum urmeaza:

```
private char connected
```

1.4.2. Generarea si editarea codului sursa Java

Aceasta sectiune prezinta **modul in care puteti genera codul sursa Java pentru diagrama de clase pe care ati creat-o in sectiunea precedenta**. Dupa ce ati creat modelul UML, puteti genera codul corespunzator Java pentru acesta. **Daca modificati modelul, puteti genera din nou codul Java folosind facilitatea UML** numita ***Generate Code***.

1.4.2.1. Generarea codului sursa Java

Puteti sa modelati aplicatia Java in UML si apoi sa generati codul Java corespunzator. Pentru a genera codul sursa trebuie sa creati un proiect Java pentru a stoca codul generat din proiectul UML.

1. Din meniul principal, alegeti ***File*** > ***New Project*** si apoi realizati urmatoarele actiuni:

   La ***Categories*** selectati ***Java***.

   La ***Projects*** selectati ***Java Application***.

   Apasati ***Next***.

2. In campul ***Project Name***, alegeti **JavaPrj1**.

3. Pentru ***Project Location*** apasati ***Browse*** si selectati directorul **UMLTutorial**.

4. **Deselectati** casutele ***Set as Main Project*** si ***Create Main Class***.

5. Apasati **Finish**. O fereastra de dialog care arata progresul va aparea. Cand proiectul **JavaPrj1** este creat, el va aparea in fereastra ***Projects***.

6. In fereastra ***Projects***, dati right-click pe nodul **UMLTutorialProject** si alegeti ***Generate Code*** din meniul pop-up.

7. In fereastra de dialog ***Generate Code***, apasati ***Browse***.

8. In fereastra de dialog ***Choose the Target Source Folder***, localizati directorul sursa pentru proiectul **JavaPrj1** pe care tocmai l-ati creat.

De exemplu: **D:\isw\441E\UMLTutorial\ JavaPrj1\src**

9. Apasati ***Open***.

10. Inapoi in fereastra de dialog ***Generate Code***, **deselectati** casuta ***Backup Existing Source Files*** si apasati ***OK***.

11. NetBeans IDE genereaza codul sursa Java iar fereastra ***Output*** afiseaza progresul operatiei de generare a codului.

12. In fereastra ***Projects*** expandati nodul **JavaPrj1** > ***Source Package*** si faceti double-click pe nodul ***<default package>*** **.**

Observati ca folderul contine fisierele sursa Java care sunt denumite in mod similar cu elementele clasa pe care le-ati creat in modelul **ClassDiagram**.

1.4.2.2. Adaugarea unor atribute folosind editorul de surse Java

Acum, adaugati un alt atribut clasei **EntryStation** folosind ***Source Editor***. Dupa ce modificati fisierul sursa Java, veti folosi facilitatea ***Reverse Engineer*** pentru a reflecta aceasta schimbare in elementul modelului UML corespunzator.

1. In fereastra ***Projects***, expandati nodul **UMLTutorialProject** si nodul ***Model*** daca e necesar.

2. Right-click pe nodul **EntryStation** si alegeti ***Navigate To Source*** din meniul pop-up. Un nou tab al ***Source Editor*** numit **EntryStation.java** apare si afiseaza codul sursa al clasei.

3. In ***Source Editor***, scrieti urmatorul cod sub primul atribut:

```
    private boolean isOperating;
```

4. Apasati ***Ctrl-S*** pentru a salva modificarea.

5. Right-click in ***Source Editor*** si alegeti ***Reverse Engineer*** din meniul pop-up. Fereastra de dialog ***Reverse Engineer*** apare.

6. Selectati ***Use Existing UML Project*** in fereastra de dialog ***Reverse Engineer*** si alegeti **UMLTutorialProject** ca proiect tinta.

7. Apasati ***OK*** pentru a initia procesul de inginerie inversa.

8. In fereastra de dialog ***Model Element Overwrite Authorization***, apasati ***Yes*** pentru a suprascrie modelul existent pentru clasa **EntryStation**.

9. Apasati tab-ul ***ClassDiagram*** pentru a va intoarce la editorul de diagrame. Atributul **isOperating** apare in elementul clasa numit **EntryStation**.

Observatie: Apasati butonul ***Fit to Window***  de pe bara ***Diagram*** pentru a centra diagrama in fereastra. Acest buton va permite sa ajustati nivelul de ***Zoom*** daca e necesar pentru a citi etichetele elementelor sau pentru a plasa alte elemente pe diagrama.

1.4.3. Gasirea elementelor diagramei in diferitele ferestre ale IDE-ului NetBeans

Puteti utiliza diferite metode pentru a localiza rapid obiecte intr-o diagrama sau in fereastra ***Projects***. Toate atributele si operatiile pe care le adaugati elementelor ***Class*** in editorul de diagrame apar in fereastra ***Projects***. Atributele sunt reprezentate de pictograma atribut  . Operatiile sunt reprezentate de pictograma operatie  .

1.4.3.1. Localizarea obiectelor din fereastra Projects in editorul de diagrame

In fereastra ***Projects***, dati double-click pe nodul **EntryStation**. In editorul de diagrame elementul de tip clasa numit **EntryStation** este selectat si centrat.

1.4.3.2. Localizarea obiectelor din editorul de diagrame in fereastra Projects

In editorul de diagrame selectati clasa **Consortium** si dati right-click pe ea. Alegeti ***Select in Model*** din meniul pop-up. In fereastra ***Projects***, numele obiectului tinta este subliniat.

1.4.4. Documentarea claselor si a diagramelor

Exista **3 metode diferite dintre care puteti alege atunci cand doriti sa introduceti o documentatie** descriptiva **pentru clase, atribute, metode si diagrame**. Procedurile nu fac obiectul acestui laborator.

1.4.5. Adaugarea asocierilor si a generalizarilor

O asociere descrie **un grup de legaturi care impart o structura si o semantica** (un inteles) comune. **Multiplicitatea** specifica **numarul de instante ale unei clase care se pot corela cu o instanta a clasei asociate**. Multiplicitatea limiteaza numarul total de componente corelate.

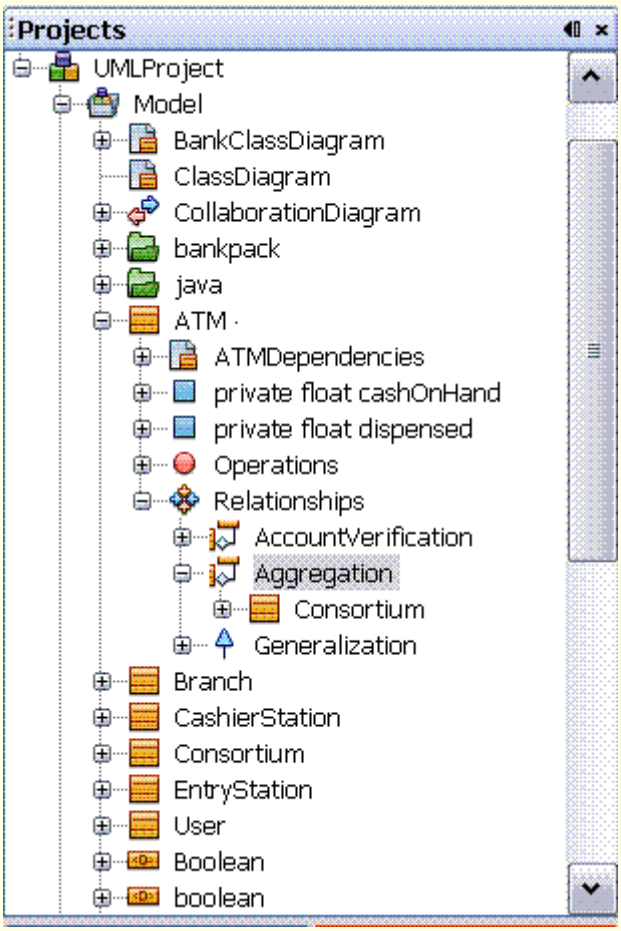
Aceasta sectiune contine urmatoarele proceduri:

1.4.5.1. Adaugarea unei asocieri intre clase

1. Din sectiunea ***Association*** a *Modeling* ***Palette***, selectati pictograma ***Aggregation***  .

2. Faceti click in interiorul elementului **ATM**, si apoi dati click pe elementul **Consortium**. O legatura apare intre cele doua clase.

3. Efectuati right-click oriunde in editorul de diagrame pentru a deselecta pictograma. Relatia este reprezentata in fereastra ***Projects*** asa cum apare mai jos:

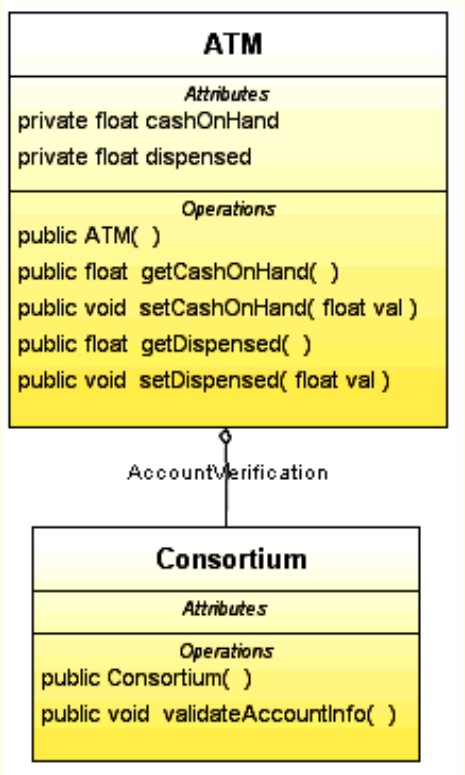


4. Selectati legatura ***Aggregation*** dintre **ATM** and **Consortium**. Cand legatura este selectata, culoarea ei va deveni albastra.

5. Pozitionati cursorul in apropierea mijlocului liniei selectate si faceti right-click.

6. Alegeti ***Labels*** > ***Link Name*** din meniul pop-up.

7. Scrieti **AccountVerification** in campul ***Name*** si apasati ***Enter***. Legatura primeste o eticheta dupa cum apare in figura urmatoare.



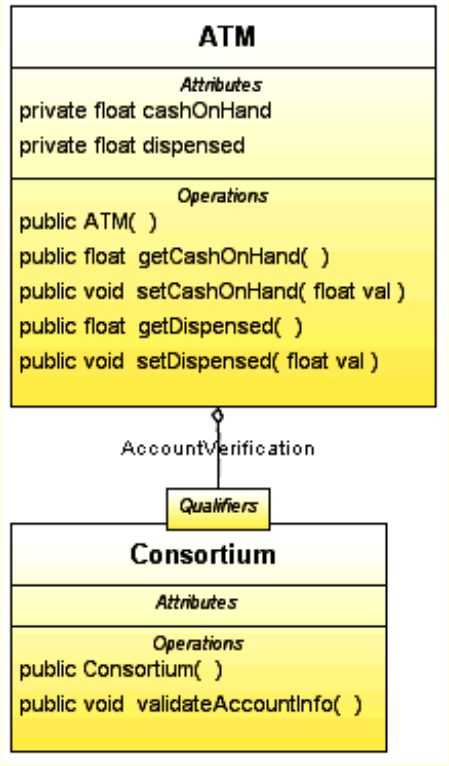
8. Din sectiunea ***Association*** a *Modeling* ***Palette***, selectati pictograma ***Association*** si desenati o legatura intre **CashierStation** si **Branch**.

9. Efectuati right-click oriunde in editorul de diagrame pentru a deselecta pictograma.

1.4.5.2. Adaugarea unei asocieri calificate intre clase

O asociere calificata coreleaza doua clase si un calificator. Calificatorul este un atribut special care reduce multiplicitatea efectiva a asocierii. Puteti descrie o asociere calificata ca fiind o casuta la capatul liniei de asociere in apropierea clasei pe care o califica. Urmatorul pas va arata **cum puteti crea o asociere calificata intre** clasa **ATM** si clasa **Consortium**.

Efectuati right-click pe legatura ***Aggregation*** unde uneste clasa **Consortium** si alegeti ***Show Qualifier***. Un calificator se va atasa de elementul clasa **Consortium** ca in figura.



1.4.5.3. Adaugarea multiplicitatii unei asocieri

In mod implicit etichetele de multiplicitate sunt ascunse.

Pentru a stabiliti multiplicitatea unei asocieri trebuie sa urmati pasii de mai jos pentru a afisa etichetele pe legatura de asociere corespunzatoare.

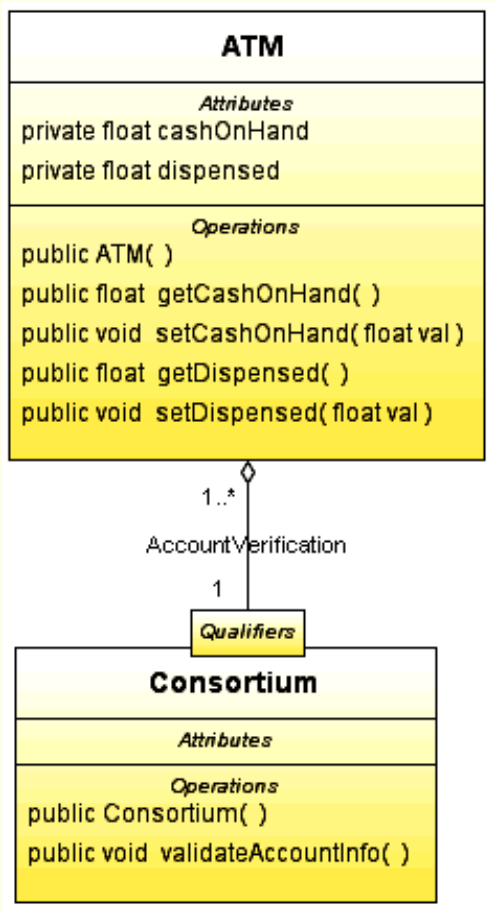
1. Faceti right-click pe legatura dintre **Consortium** si **ATM** si alegeti ***Labels*** > ***Both End Multiplicities***. Meniul pop-up se inchide si apar etichetele pentru legatura.

2. Faceti right-click pe micul romb din partea superioara a legaturii de agregare (in apropierea elementului **ATM**) si alegeti ***Set Multiplicity***.

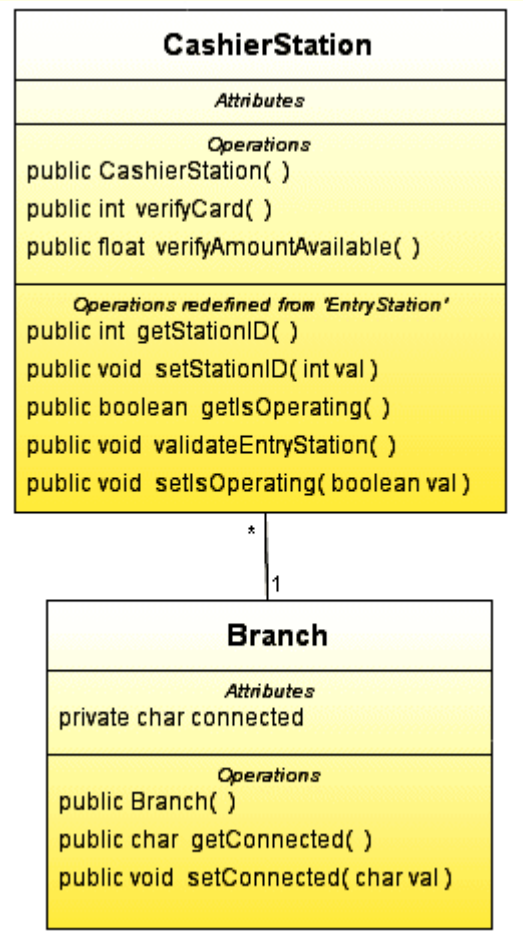
Observatie: Daca aveti dificultati in a face sa apara meniul pop-up corect, lungiti sageata de agregare mutand elementul Consortium mai departe de elementul ATM.

3. Selectati **`1..*`**

Observati ca portiunea inferioara a legaturii este etichetata **1** dupa cum apare in figura



4. Executati aceeasi procedura si setati multiplicitatea pentru legatura de asociere intre **CashierStation** si **Branch** dupa cum apare in figura urmatoare.



1.4.5.4. Adaugarea generalizarii si a mostenirii

Generalizarea este **relatia dintre o clasa si una sau mai multe versiuni extinse ale acelei clase**. **Clasa care va fi extinsa** se numeste **superclasa** iar fiecare **versiune extinsa** va fi numita **subclasa**.

Atributele si metodele din superclasa pot fi utilizate in subclase, si astfel, fiecare **subclasa** se spune ca **mosteneste toate facilitatitle superclasei**. Puteti organiza clasele folosind mostenirea astfel incat acestea sa aiba o structura comuna. O legatura de generalizare semnifica faptul ca o clasa poate mosteni o multime de atribute si metode de la clasa parinte.

1. Din sectiunea ***Basic*** a *Modeling* ***Palette***, selectati pictograma ***Generalization*** .

2. Faceti click in interiorul elementului clasa **ATM** , si apoi click pe elementul clasa **EntryStation**.

Va aparea fereastra de dialog ***Select Methods to Redefine***.

3. Bifati casuta de langa nodul **ATM** pentru a selecta toate metodele si apasati ***OK***.

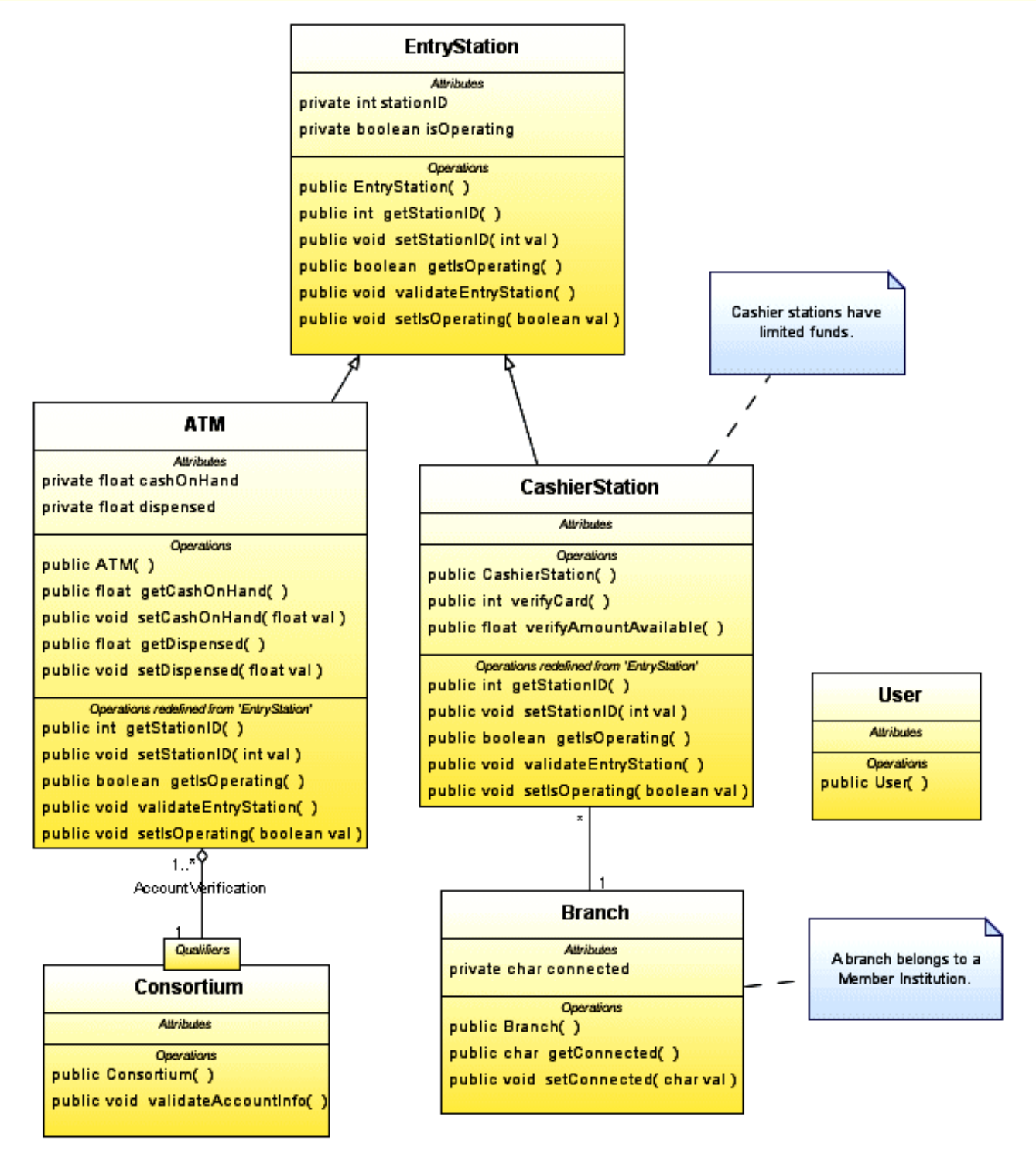
Metodele selectate sunt adaugate elementului clasa **ATM** si apare o legatura de generalizare intre cele 2 clase.

4. Desenati inca o legatura de generalizare apasand intai pe **CashierStation**, si apoi pe **EntryStation**.

5. In fereastra de dialog ***Select Methods to Redefine***, bifati casuta de langa nodul **CashierStation** pentru a selecta toate metodele si apoi apasati ***OK***.

6. Faceti right-click oriunde in editorul de diagrame pentru a deselecta pictograma ***Generalization***.

7. Diagrama finala ar trebui sa arate ca in figura urmatoare



1.4.6. Salvarea diagramelor si actualizarea fisierelor sursa Java

Executati urmatorii pasi pentru a salva diagramele si a actualiza fisierele sursa Java pentru aceste diagrame.

1. Dupa ce terminati diagrama dati right-click pe tab-ul **ClassDiagram** si alegeti ***Save Document*** din meniul pop-up. Meniul se inchide si diagrama de clase este salvata.

2. In fereastra ***Projects***, faceti right-click pe nodul **UMLTutorialProject** si alegeti ***Generate Code*** din meniul pop-up.

3. In fereastra de dialog ***Generate Code*** acceptati directorul sursa implicit **JavaPrj1** si executati backup la fisierele sursa existente. Optional, puteti apasa ***Browse*** pentru a localiza si folosi alt director sursa.

4. Apasati ***OK***.

NetBeans IDE genereaza codul si fereastra ***Output*** afiseaza progrsul operatiei de generare a codului. Directorul sursa Java pentru proiectul **JavaPrj1** ar trebui sa fie actualizat cu schimbarile pe care le-ati facut la diagrame.

Observatie: Directorul sursa pentru proiectul **JavaPrj1** ar putea include si fisierele de backup care au fost create **daca ati bifat** casuta ***Backup Existing Source Files*** in fereastra de dialog ***Generate Code*** si ati ales sa folositi acelasi director sursa. Fisierele de backup sunt denumite in acelasi mod cu fisierele originale, dar extensiile fisierelor contin numere. De exemplu, **ATM.java** are un fisier de backup numit **ATM.java1**

1.4.7. Sumar

In acest material ati invatat **sa creati o diagrama de clase pentru o aplicatie bancara simpla**. Ati invatat cum sa realizati urmatoarele lucruri:

Crearea unei diagrame de clase

Adaugarea si definirea atributelor si metodelor pentru elementele de tip clasa

Descrierea legaturilor si asocierilor intre elementele clasa

Generarea codului sursa Java pentru diagrame

Atentie: **La finalul laboratoarelor stergeti proiectele create** (click dreapta pe nodul proiectului in fereastra ***Projects***, selectati ***Delete*** si **confirmati ca doriti sa fie sterse sursele – in cazul Java**)

Kituri si documentatie:

netbeans-6.1-windows.exe,
jdk-6u5-windows-i586-p.exe,
Javadoc.zip,
jdk-6u5-windows-**amd64**.exe,
jdk-6u5-windows-**x64**.exe

1.5. Teme pentru acasa

1.5.1. Enunturi

Tema de casa are trei parti:

(I) constructia unei diagrame UML de clase pornind de la clase si relatii intre ele date (in cadrul unui nou proiect UML) – diagrama de clase a unui domeniu/sistem

(II) identificarea unor cazuri de utilizare posibile pentru domeniul/sistemul al carei structuri este reprezentat de diagrama UML de clase creata si constructia unei diagrame UML de cazuri de utilizare (in cadrul aceluiasi proiect UML),

(III) scrierea naratiunii (textului) asociate unuia dintre cazuri de utilizare anterior create

Urmatoarele variante de teme de casa vor fi realizate de grupuri de 2-3 studenti.

Partea I-a Construiti (mai intai pe hartie si apoi in cadrul unui nou proiect UML) o diagrama UML de clase care cuprinde urmatoarele clase:

Varianta A (Sistem de gestiune a produselor dintr-un magazin virtual)

StocProduse, Produs, CosCumparaturi, ProdusPromotional, ProdusObisnuit, ProdusCuPretRedus, ProdusCuAltProdusCadou,

intre care exista urmatoarele relatii de asociere intre clasele:

(1) **StocProduse** si **Produs**,
(2) **Produs** si **CosCumparaturi**,

si urmatoarele relatii de generalizare/specializare intre clasele:
(1) **Produs** si respectiv **ProdusPromotional, ProdusObisnuit**,
(2) **ProdusPromotional** si respectiv **ProdusCuPretRedus, ProdusCuAltProdusCadou**.

Varianta B (Sistem de gestiune a elementelor unei retele):

Retea, ElementRetea, SoftwareDeConfigurare, RouterWiFi, **RouterEthernet, AplicatieWeb**, **AplicatieDesktop**

intre care exista urmatoarele relatii de asociere intre clasele:
(1) **Retea** si **ElementRetea**,
(2) **ElementRetea** si **SoftwareDeConfigurare**,

si urmatoarele relatii de generalizare/specializare intre clasele:
(1) **ElementRetea** si respectiv **RouterWiFi**, **RouterEthernet**,
(2) **SoftwareDeConfigurare** si respectiv **AplicatieWeb**, **AplicatieDesktop**.

Varianta C (Sistem de gestiune a fluxurilor media):

MediaController, StreamingNode,* Transmitator**, **Receptor, Retransmitator, *MediaPlayer, MediaMixer

intre care exista urmatoarele relatii de asociere intre clasele:
(1) **Receptor** si ***MediaPlayer***,
(2) ***MediaController*** si ***StreamingNode***,

si urmatoarele relatii de generalizare/specializare intre clasele:
(1) ***StreamingNode*** si respectiv **Transmitator**, **Receptor** si **Retransmitator**,
(2) **Receptor** si respectiv ***MediaMixer***.

Varianta D (Sistem de gestiune a dobanzilor bancare):

Depozit, Dobanda, DobandaLunara, DobandaAnuala, ConturiClient, ContCurent, ContCurentPremium,

intre care exista urmatoarele relatii de asociere intre clasele:
(1) **ConturiClient** si **Depozit**,
(2) **ConturiClient** si **ContCurent**,
(3) **Depozit** si **Dobanda**

si urmatoarele relatii de generalizare/specializare intre clasele:
(1) **Dobanda** si respectiv **DobandaLunara** si **DobandaAnuala**,
(2) **ContCurent** si respectiv **ContCurentPremium**.

Varianta E (Sistem de gestiune a autovehiculelor):

Autovehicul, Detinator, PersoanaParticulara, AngajatFirma, Asigurare, AsigurareObligatorie, AsigurareFacultativa,

intre care exista urmatoarele relatii de asociere intre clasele:
(1) **Autovehicul** si **Detinator**,
(2) **Autovehicul** si **Asigurare**,

si urmatoarele relatii de generalizare/specializare intre clasele:
(1) **Asigurare** si respectiv **AsigurareObligatorie** si **AsigurareFacultativa**,
(2) **Detinator** si respectiv **PersoanaParticulara** si **AngajatFirma**.

Varianta F (Sistem de gestiune a biletelor de avion):

BiletAvion, Loc, Cursa, CursaLowCost, CursaDeLinie, LocClasaEconomic, LocClasaBusiness,

intre care exista urmatoarele relatii de asociere intre clasele:
(1) **BiletAvion** si **Loc**,
(2) **BiletAvion** si **Cursa**,

si urmatoarele relatii de generalizare/specializare intre clasele:
(1) **Cursa** si respectiv **CursaLowCost** si **CursaDeLinie**,
(2) **Loc** si respectiv **LocClasaEconomic** si **LocClasaBusiness**.

Partea a II-a Incercati sa identificati cat mai multe cazuri de utilizare posibile pentru domeniul/sistemul al carei structuri este reprezentat de diagrama UML de clase creata si construiti cu ele o diagrama UML de cazuri de utilizare (in cadrul aceluiasi proiect UML),

Partea a III-a Pentru unul dintre cazuri de utilizare anterior create scrieti naratiunea (textul) asociat.

Template-ul care trebuie folosit pentru scrierea naratiunii unui caz de utilizare:

1. **Numele cazului de utilizare** - cat mai sugestiv, pentru a sintetiza cazul de utilizare
2. **Scurta descriere a cazului de utilizare** - rezumatul cazului de utilizare (in 1-3 linii de text)
3. **Actori** - entitati exterioare sistemului modelat, implicate in cazul de utilizare
4. **Preconditii** - conditiile necesare pentru declansarea cazului de utilizare
5. **Evenimentul care declanseaza cazul de utilizare** - modul in care incepe cazul de utilizare
6. **Descriere a interactiunii dintre actori si fiecare caz de utilizare** - "scenariul" principal
7. **Alternative la cazul de utilizare principal** - scenariile alternative si situatiile exceptionale
8. **Evenimentul care produce oprirea cazului de utilizare** - modul in care se incheie cazul
9. **Postconditii** - efectele incheierii cazului de utilizare

1.5.2. Modul de realizare si prezentare a temei

Tema trebuie realizata pana la lucrarea 3, sub **forma** unui **proiect NetBeans + fisier DOC** (**NU unul docx!!!**).

In **laboratorul urmator (lucrarea 3) proiectul NetBeans** creat acasa trebuie sa fie adus (pe USB, de pe mail, etc.) si **incarcat intr-un calculator din laborator** (**sau** adus pe un **laptop personal**), si **prezentat**.

Separat se cere si un **listing al diagramelor + textului cazului de utilizare** (cu numele echipei care le-a realizat pe prima pagina).

Temele de casa **la urmatoarele 2 laboratoare** vor fi **continuari** ale acesteia, asa incat este **importanta pastrarea proiectului creat** pentru a fi ulterior actualizat, **si a listingurilor** (sub forma unui portofoliu) **pentru a fi predate la ultimul laborator**.

1.5.3. Exemplu de rezolvare

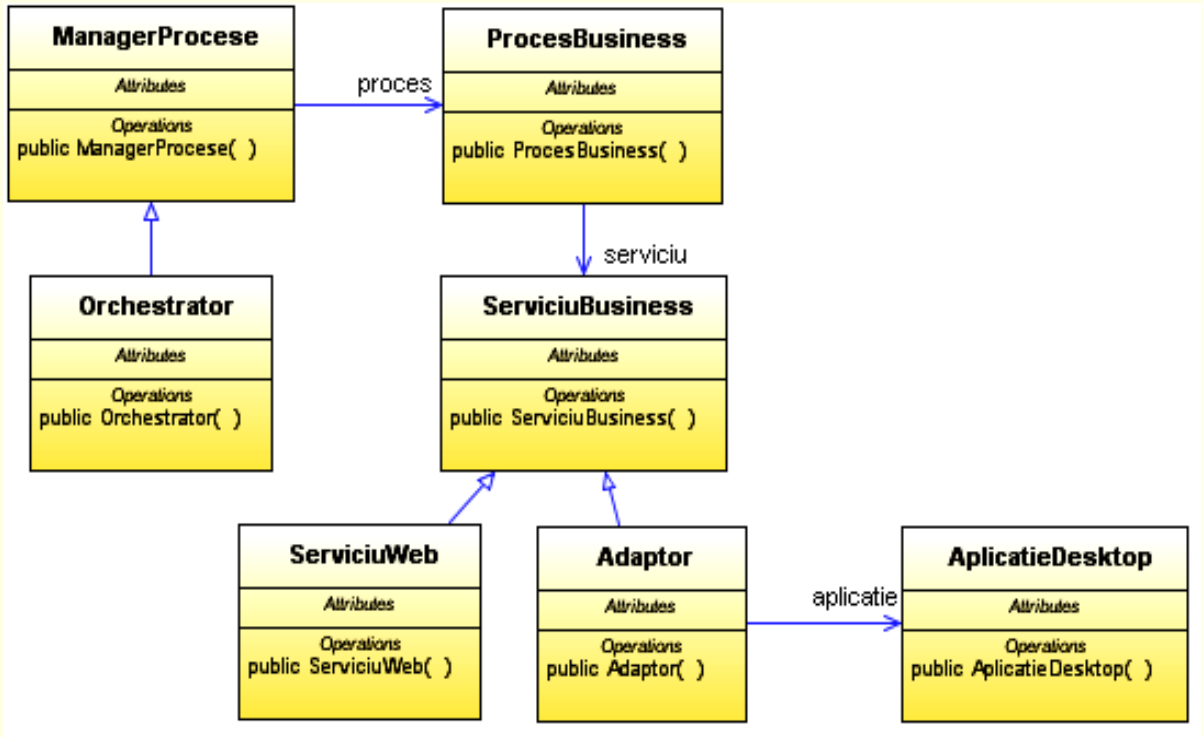
Partea I-a Constructia unei diagrame UML de clase care cuprinde urmatoarele clase (Sistem de management al proceselor *business*):

AplicatieDesktop, Adaptor, ServiciuWeb, ProcesBusiness, ServiciuBusiness, ManagerProcese, Orchestrator,

intre care exista urmatoarele relatii de asociere intre clasele:
(1) **Adaptor** si **AplicatieDesktop**,
(2) **ProcesBusiness** si **ServiciuBusiness**,
(3) **ManagerProcese** si **ProcesBusiness**

si urmatoarele relatii de generalizare/specializare intre clasele:
(1) **ServiciuBusiness** si respectiv **ServiciuWeb** si **Adaptor**,
(2) **ManagerProcese** si respectiv **Orchestrator**.

Posibila rezolvare:



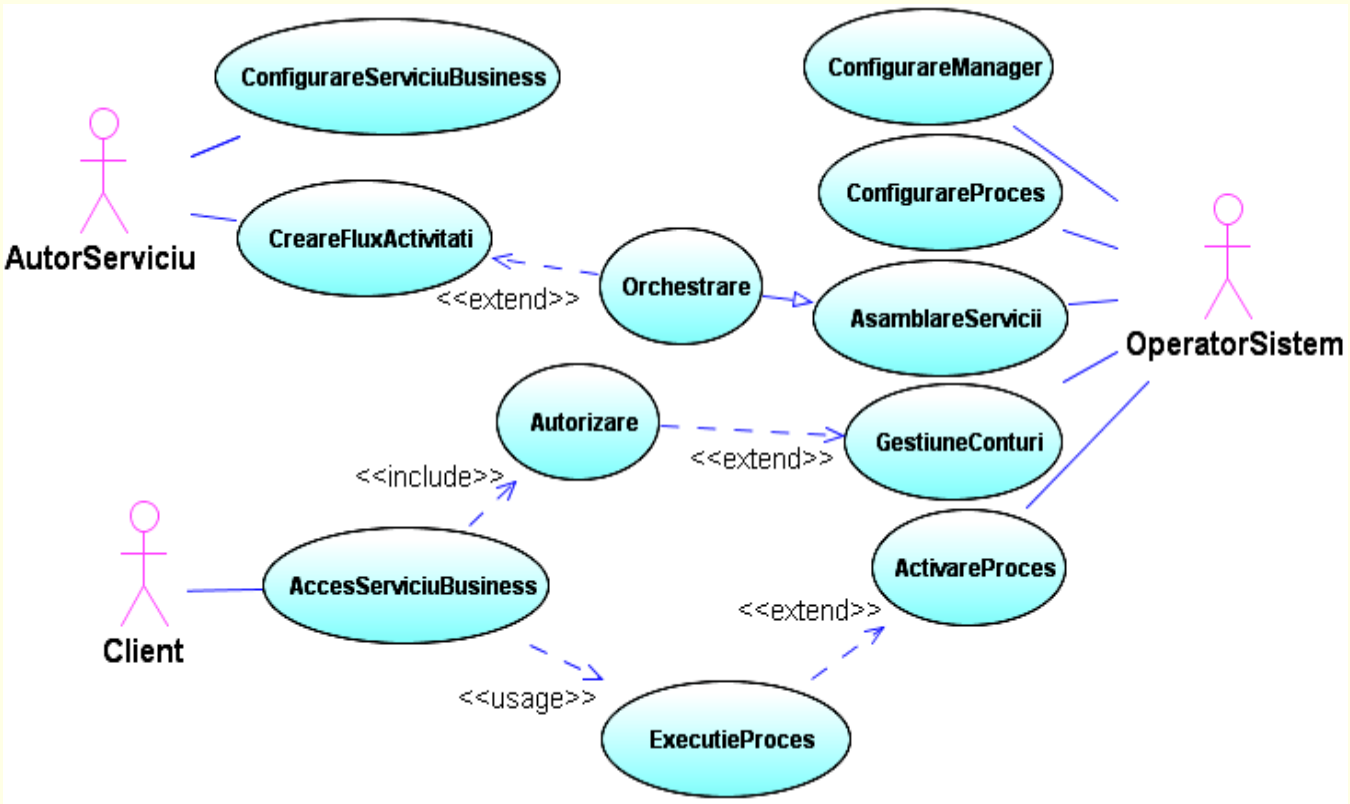
Se pot observa in plus fata de specificatie detalii precum

- **navigabilitatea asocierilor (de ex. dinspre Adaptor catre AplicatieDesktop)** sau
- **numele unor roluri (proces, aplicatie** si **serviciu).**

Astfel de detalii pot fi adaugate si diagramelor date ca tema.

Partea a II-a Identificarea a cat mai multe cazuri de utilizare posibile pentru domeniul/sistemul al carei structuri este reprezentat de diagrama UML de clase creata, si constructia cu ele a unei diagrame UML de cazuri de utilizare

Posibila rezolvare:



Se pot observa

- actori: Client, AutorServiciu, OperatorSistem

- cazuri de utilizare de baza: ConfigurareServiciuBusiness, ConfigurareManager, ConfigurareProces, CreareFluxActivitati, AccesServiciuBusiness, GestiuneConturi, ActivareProces

- cazuri de utilizare extensie: Orchestrare, Autorizare, ExecutieProces

- cazuri de utilizare incluse: Autorizare

- cazuri de utilizare utilizate: ExecutieProces

Partea a III-a Naratiunea asociata unuia dintre cazuri de utilizare anterior create.

Posibila rezolvare:

Naratiunea cazului de utilizare Autorizare:

1. Numele cazului de utilizare
Autorizare

2. Scurta descriere a cazului de utilizare
Clientul ii transmite sistemului datele necesare autorizarii, si in caz de succes capata accesul.

3. Actori
Clientul

4. Preconditii
Sistemul e aflat in executie, cunoaste datele necesare autorizarii Clientului si ii ofera Clientului o modalitate de a-i transmite aceste date.

5. Evenimentul care declanseaza cazul de utilizare
Clientul cere autorizarea de catre sistem.

6. Descriere a interactiunii dintre actori si fiecare caz de utilizare
1. Clientul transmite datele necesare autorizarii de catre sistem.
2. Sistemul verifica autenticitatea datelor primite de la Client (extinzand cazul de utilizare GestiuneConturi).
3. Sistemul prezinta un mesaj de confirmare catre Client.
4. Clientul capata acces la sistem.

7. Alternative la cazul de utilizare principal
E1. Daca sistemul nu este lansat, autorizarea nu poate avea loc.
E2. Daca sistemul nu cunoaste datele de autorizare, autorizarea esueaza.
E3. Daca Clientul transmite in mod eronat datele de autorizare, autorizarea esueaza.

8. Evenimentul care produce oprirea cazului de utilizare
Clientului i se prezinta un mesaj de confirmare.

9. Postconditii
Clientul capata acces la sistem.

Cum ar putea arata naratiunea cazului de utilizare AccesServiciuBusiness?

Dar naratiunea cazului de utilizare ExecutieProces?