



**NOKIA**  
Connecting People



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

**ITNQ**  
2011

POO – an II

TPI – an III

Laboratorul **SAIM** – sala **B123**

25/03/2012

## Inginerie Software in Comunicatii (ISC)



2011-2012

### Laborator 2

**Crearea diagramelor UML de comunicare (anterior de colaborare) si de secventa a mesajelor schimbate (MSC). Crearea diagramelor UML de activitati. Discutarea temelor de casa specificate la lucrarea 1**

#### *Descrierea laboratorului*

In aceasta lucrare de laborator vor fi acoperite urmatoarele probleme:

- [Crearea diagramelor UML de comunicare \(colaborare\)](#) din NetBeans 6.1
- [Crearea diagramelor UML de secventa \(MSC\)](#) din NetBeans 6.1
- [Crearea diagramelor UML de activitati \(organigrama\)](#) in NetBeans 6.1
- [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**.

#### **2.1. Crearea diagramelor UML de comunicare (colaborare)**

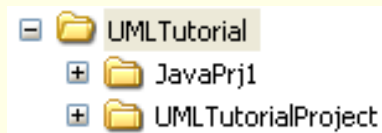
In continuare veti invata cum sa folositi functiile UML ale IDE pentru a crea diagrame UML de comunicare (foste de colaborare in UML 1).

O diagrama de comunicare (colaborare) este o **diagrama de interactiuni care subliniaza organizarea structurala a obiectelor care trimit si primesc mesaje**. Ea arata **interactiunile organizate in jurul instantelor si a legaturilor dintre ele**. Diagrama de comunicare (colaborare) poate fi folosita pentru a:

- descrie un scenariu specific prin **ilustrarea deplasarii mesajelor intre obiecte**
- arata o **organizare spatiala a obiectelor si interactiunilor**

### 2.1.1. Incarcarea proiectelor realizate in lucrarea anterioara

1. Verificati daca exista directorul numit **UMLTutorial** pe drive-ul **D:** in directorul **\isw**, in subdirectorul cu numarul grupei, (de exemplu: **D:\isw\441E\UMLTutorial**). Daca exista, stergeti-l.
2. Descarcati arhiva [UMLTutorial.zip](#), in subdirectorul cu numarul grupei, (de exemplu: **D:\isw\441E\**).
3. Dezarhivati fisierul descarcat, folosind WinZip sau WinRar, cu click dreapta pe numele fisierului **UMLTutorial.zip** selectand apoi optiunea **Extract Here**.  
Automat va fi creat subdirectorul **UMLTutorial** iar in el subdirectoarele **UMLTutorialProject** (in care se afla proiectul UML) si **JavaPrj1** (in care se afla proiectul Java).



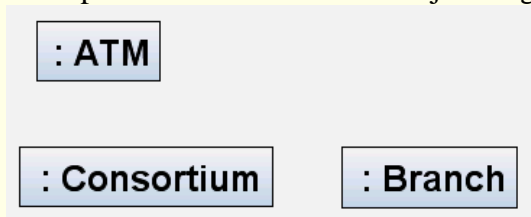
4. Din meniul principal selectați **File > Open Project**. Apare asistentul **Open Project**  
Sub **Look in**, selectati calea proiectului UML anterior dezarhivat (de forma: **D:\isw\441E\UMLTutorial\UMLTutorialProject**).  
Lasati selectat **Open as Main Project**  
Faceti **Click** pe **Open Project**.  
In **Projects** apare nodul **UMLTutorialProject**.
5. Din meniul principal selectați **File > Open Project**. Apare asistentul **Open Project**  
Sub **Look in**, selectati calea proiectului Java anterior dezarhivat (de forma: **D:\isw\441E\UMLTutorial\JavaPrj1**).  
Deselectati **Open as Main Project**  
Faceti **Click** pe **Open Project**.  
In **Projects** apare nodul **JavaPrj1**.

### 2.1.2. Generarea unei diagrame de comunicatie (colaborare)

1. In fereastra **Projects**, deschideti nodul **UMLTutorialProject > Model**
  2. Selectati urmatoarele noduri **Class**:  
**ATM      Branch      Consortium**
- Observatie:** puteti selecta mai multe clase apasand **Ctrl** si selectand fiecare nod clasa
3. Click - dreapta pe ultima clasa selectata si alegeti **Create Diagram From Selected Elements** din meniul aparut. Se deschide **New Wizard** si se afiseaza **Create New Diagram**
  4. Din lista **Diagram Type** selectati **Collaboration Diagram**
  5. In campul **Diagram Name** scrieti **CollaborationDiagram**
  6. Lasati setarea implicita in campul **Namespace** si apasati **Finish**. IDE-ul realizeaza urmatoarele:  
Creeaza nodul **CollaborationDiagram** sub nodul **Model** in fereastra **Projects**  
Afiseaza noua diagrama in editorul de diagrame (diagrama este alcatuita din trei clase descrise ca elemente vitale)  
Deschide **Modeling Palette**

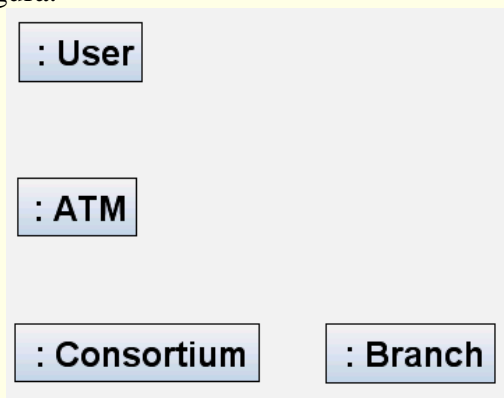
### 2.1.3. Rearanjarea unei diagrame de comunicare (colaborare)

1. Cu click si prin deplasarea elementelor rearanjati diagrama ca mai jos:



2. In fereastra *Projects* selectati nodul clasei marcat ca **User**

3. Trageti clasa selectata in editorul diagramei si plasati-o deasupra elementului **ATM**, dupa cum este prezentat in figura:



### 2.1.4. Adaugarea legaturilor conector

Fiecare element din cadrul unei diagrame de comunicare (colaborare) poate fi legat prin conexiuni la alte elemente. Aceste legaturi pot fi identificate si le pot fi adaugate fluxuri de mesaje.

1. Din sectiunea de baza a *Modeling Palette* selectati pictograma **Connector**
2. Selectati elementul *Lifeline* marcat **User** si apoi selectati **ATM**. O legatura este trasata intre cele doua elemente
3. Folositi aceeasi procedura pentru a trasa urmatoarele legaturi:

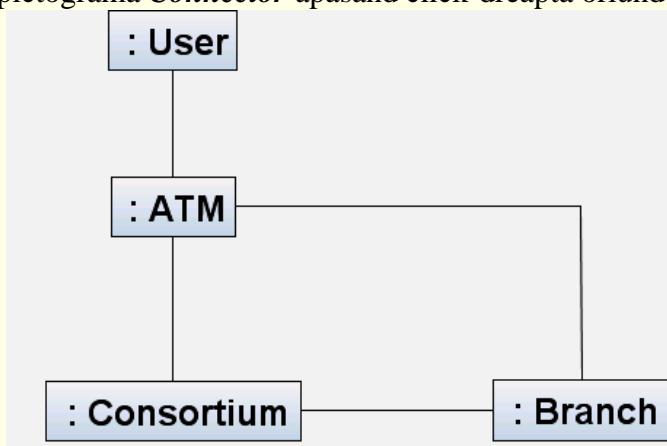
**ATM – Consortium**

**Consortium – Branch**

**ATM – Branch**

Diagrama ar trebui sa fie similara celei de mai jos:

4. Deselectati pictograma *Connector* apasand click-dreapta oriunde in editorul de diagrame.



## 2.1.5. Vizualizarea numerelor mesajelor

O diagrama de comunicare (colaborare) prezinta un scenariu specific utilizand mesaje numerotate. Setarea implicita a IDE-ului ascunde aceste numere.

Parcurgeti urmatoorii pasi pentru vizualizarea numerelor:

1. Click-dreapta in background-ul editorului diagramei **CollaborationDiagram**
2. Alegeti **Show Message Numbers** din meniul aparat.


Cand introduceti succesiunea de operatii (in urmatoarea sectiune), vor aparea numerele mesajelor.

**Observatie:** intregul reprezinta ordinea secventiala a mesajului in cadrul urmatorului nivel ierarhic al apelarii procedurale. Mesajele care difera printr-un intreg sunt secvential asociate acelu nivel. Spre exemplu, mesajul **3.1.4** urmeaza mesajului **3.1.3** in cadrul "activarii" **3.1**.

## 2.1.6. Vizualizarea fluxurilor de operatii (mesaje)

Un flux de operatii (mesaje) este reprezentat pe diagrama ca o sageata etichetata, plasata paralel cu legatura. Aceasta legatura este folosita ca sa transporte sau sa implementeze livrarea mesajului catre elementul dorit.

1. In editorul diagramei selectati legatura **Connector** dintre **User** si **ATM**
2. Click-dreapta pe portiunea legaturii cea mai apropiata de elementul **ATM**
3. Alegeti **Operations** > **public float getCashOnHand** din meniul aparat. Fluxul de operatii este plasat pe diagrama si numetotat cu intregul **1**

**Observatie:** Dati click pe butonul **Fit to Window**  pentru a vedea intreaga diagrama in editor.

4. Selectati legatura dintre **ATM** si **Consortium**, click-dreapta pe legatura apropiata elementului **Consortium**

5. Alegeti **Operations** > **public void validateAccountInfo** din meniul aparat.

IDE-ul plaseaza operatia selectata pe aceasta legatura si o numereaza cu intregul **1.1**

**Observatie:** puteti selecta si repositiona fluxul de operatii

## 2.1.7. Adaugarea unor operatii la o clasa existenta

In aceasta procedura puteti adauga o operatie noua legaturii **Connector**. Aceasta operatie este de asemenea adaugata clasei **Branch** in diagrama **ClassDiagram** si in codul sursa Java.

1. Click-dreapta pe legatura dintre **Consortium** si **Branch**, aproape de elementul **Branch**
2. Alegeti **Operations** > **Add Operation** din meniul aparat. Eticheta apare in diagrama cu termenul **Unnamed** subliniat
3. Scrieti **verifyCardWithBank**
4. Folositi sageata drapta pentru a muta cursorul pe campul operatiilor cu parametrii
5. Scrieti **int stringCardStrip** pentru parametru si apasati **Enter**.

Legatura este etichetata

```
1.1.1 : public void verifyCardWithBank(int stringCardStrip)
```

si operatia apare in clasa **Branch** in diagrama **ClassDiagram**

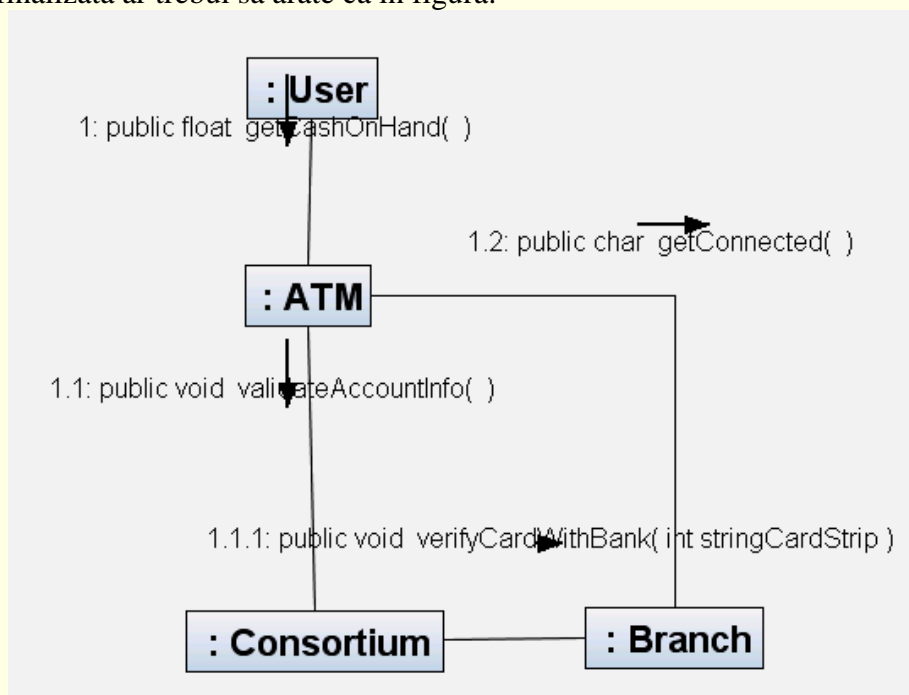
6. Pe legatura dintre **ATM** si **Branch**, click-dreapta aproape de elementul **Branch**

7. Alegeti *Operations* > **public char getConnected** din meniul aparut.

Legatura va fi etichetata

**1.2 : public char getConnected()**

Diagrama finalizata ar trebui sa arate ca in figura:



### 2.1.8. Salvarea diagramei de comunicare (colaborare)

Dupa ce ati terminat diagrama de comunicare (colaborare), o puteti salva.

1. In editorul diagramei, click-dreapta pe tab-ul **CollaborationDiagram**

2. Selectati *Save Document* din meniul aparut.

Meniul se va inchide si diagrama va fi salvata.

**Observatie:** Cand iesiti din IDE sunteti intrebati daca doriti sa salvati proiectul dvs.

### 2.1.9. Rezumat

In acest tutorial ati invatat sa creati diagrame de comunicare (colaborare) utilizand o diagrama de clasa deja existenta.

Ati invatat sa efectuati urmatoarele:

Sa **generati o diagrama de comunicare (colaborare)**

Sa **adaugati legaturi** pe diagrama

Sa **vizualizati numerele de mesaje** pe diagrama

Sa **vizualizati si etichetati fluxurile de operatii**

Sa **adaugati operatii legaturilor existente** pe diagrama

Sa **salvati diagrama**

## 2.2. Crearea diagramelor UML de secventa (MSC)

In continuare veti invata cum sa folositi functiile UML ale IDE-ului pentru a crea diagrame de secventa (MSC = *Message Sequence Chart*). O diagrama de secventa este o reprezentare vizuala a interactiunilor dintre grupuri de obiecte ce colaboreaza intr-un sistem. Diagramele de secventa sunt alcatuite din linii verticale numite *lifelines*. Fiecare element *lifeline* reprezinta viata unui obiect. *Lifeline*-urile sunt conectate prin linii orizontale care ilustreaza mesaje ce trec de la un obiect din scenariu la alt obiect in scenariu.

### 2.2.1. Generarea unei diagrame de secventa (MSC)

1. In fereastra *Projects*, click-dreapta pe nodul **UMLTutorialProject** > *Model* > *CollaborationDiagram* si alegeti **Create Diagram From Selected Elements** din meniul aparut. Se deschide *New Wizard* si se afiseaza *Create New Diagram*.

**Observatie:** Ceea ce va intereseaza este una din cele doua tipuri de diagrame de interactiune disponibile in meniul aparut: diagrama de comunicare (colaborare) si diagrama de secventa (MSC). Aceste diagrame aceste diagrame evidentiaza interactiunile dintre obiecte.

2. In lista de *Diagram Type*, selectati **Sequence Diagram**

3. In campul *Diagram Name* scrieti **SequenceDiagram**

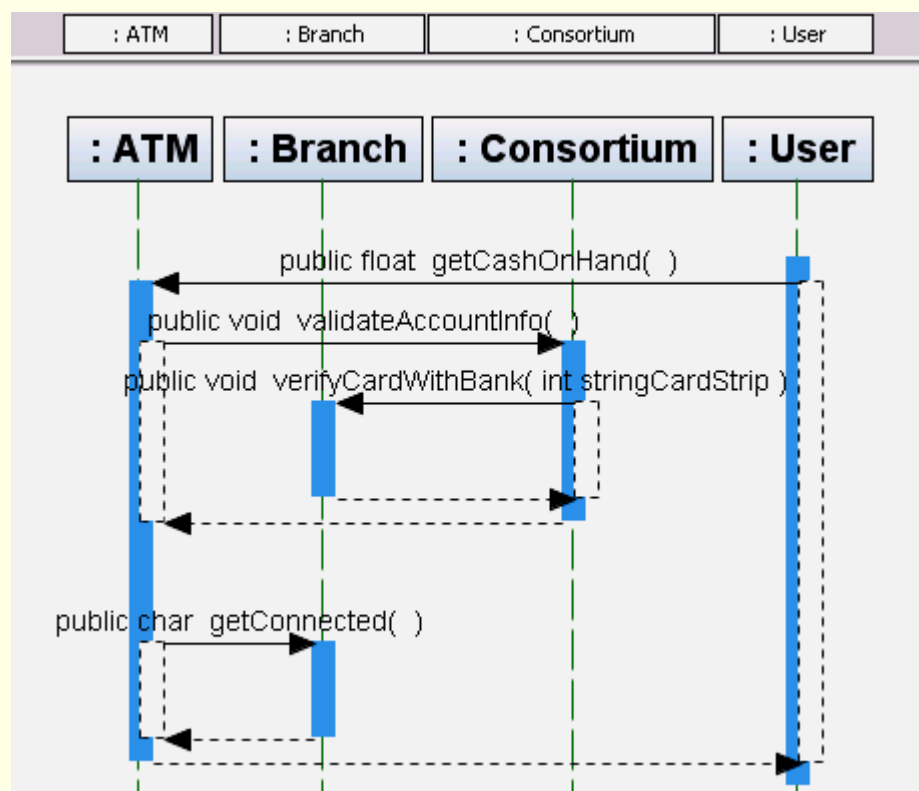
4. Lasati setarea standard in campul *Namespace* si apasati *Finish*. IDE va realiza urmatoarele:

Creaza nodul **SequenceDiagram** sub nodul **CollaborationDiagram** in fereastra *Projects*

Afiseaza noua diagrama in editorul de diagrame (diagrama este alcatuita din elemente lifeline si mesaje din diagrama de comunicare/colaborare)

Deschide *Palette* si afiseaza pictogramele utilizate in crearea diagramelor de secventa

Diagrama ar trebui sa fie asemanatoare celei de mai jos:



## 2.2.2. Reorganizarea unei diagrame de secventa (MSC)

Obiectele dintr-o tranzactie sunt simbolizate cu linii punctate verticale (*lifeline*), cu numele scrise in partea de sus. Pentru ca ordinea elementelor *lifeline* nu este neaparat predefinita la crearea diagramei de secventa, trebuie sa le rearanjati in acest moment.

1. Selectati elementul *lifeline* etichetat **User** si mutati-l in stanga diagramei
2. Rearanjati elementele ramase plasandu-le uniform pe suprafata editorului de diagrama. Aranjati elementele in urmatoarea ordine (de la stanga la dreapta) :

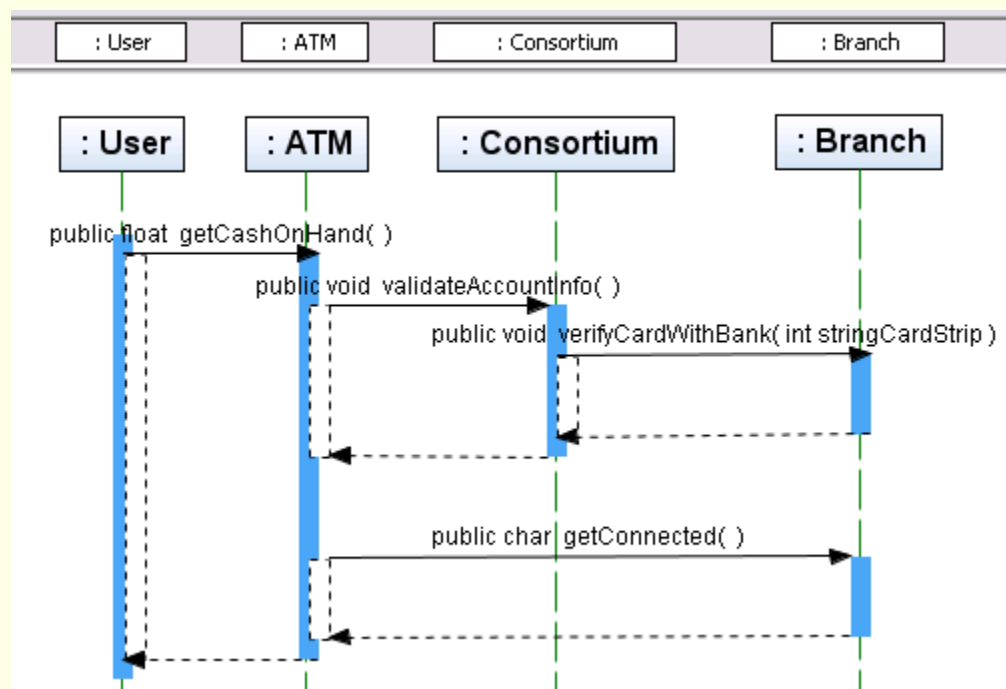
**User**

**ATM**

**Consortium**

**Branch**

Diagrama ar trebui sa fie asemanatoare figurii:



## 2.2.3. Adaugarea unui element mesaj reflexiv (catre sine)


Un element *Message to Self* reprezinta un **mesaj trimis catre obiectul care il trimite**. In continuare vor fi realizati pasii necesari pentru adaugarea unui element *Message to Self*.

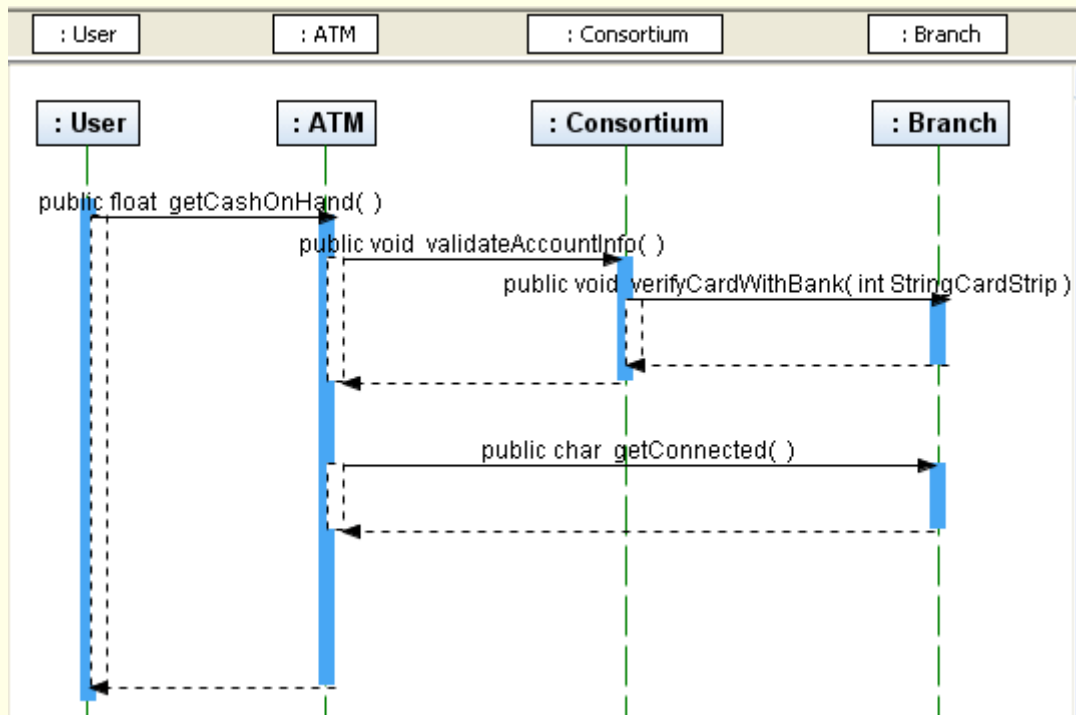
1. Ar putea fi necesara prelungirea elementelor *lifeline* pentru a usura plasarea de mesaje. Pentru prelungirea *lifeline*-urilor parcurgeti urmatoarii pasi:

Selectati elementul **User**

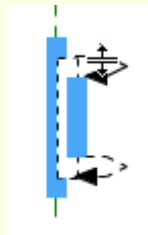
Click pe dreptunghiul albastru din centru si trageți dreptunghiul in jos pentru alungirea *lifeline*-ului

Repetati primii doi pasi pentru alungirea elementelor ramase, pana cand diagrama se aseamana celei de mai jos:

2. Din sectiunea de baza a *Modeling Palette* selectati pictograma de **Message to self** 
3. Click pe sectiunea de jos din elementul *lifeline* extins. IDE-ul plaseaza un mesaj catre sine pe *lifeline*



4. Click-dreapta oriunde in editorul de diagrama pentru deselectarea pictogramei *Message to self*
5. Pe mesajul catre sine click pe sageata de jos a mesajului. Proprietatile mesajului apar in fereastra *Properties*
6. In fereastra *Properties*, in campul *Name*, scrieti **validateCashOnHandOperation** si apasati *Enter*
7. Click-dreapta pe sageata de sus a mesajului catre sine si selectati *Operations* din meniul aparut. Urmatoarea figura indica unde sa pozitionati cursorul. Atentie la linia dubla aflata deasupra sagetii de sus



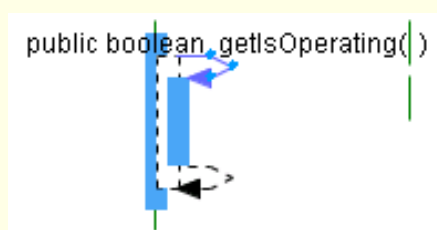
8. Alegeti *Operations* > *Add Operation* din meniul aparut. Un editor de o singura linie se deschide si afiseaza urmatoarea informatie:

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

definiti operatia in felul urmatoar:

```
public boolean getIsOperating()
```

Partea de sus a mesajului este etichetata ca in figura:

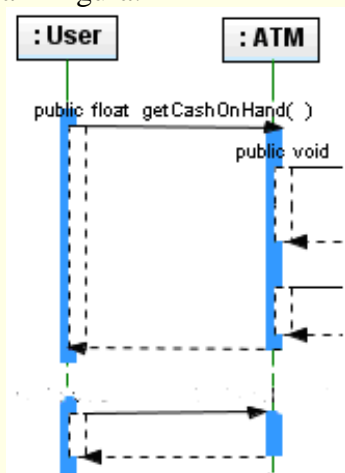




## 2.2.5. Adaugarea unui element mesaj apel de operatie

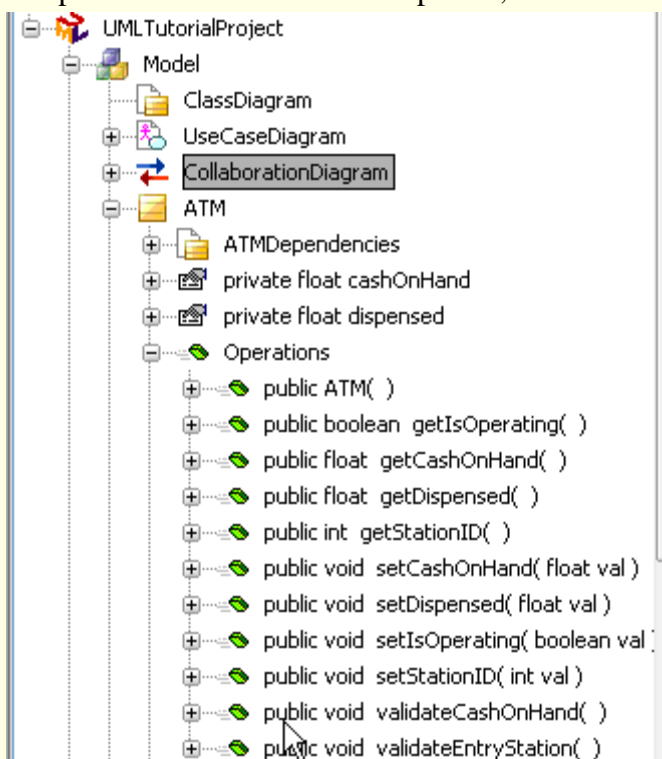
Deoarece operatia **validateCashOnHand** face parte din clasa **ATM**, este necesara plasarea fluxului de mesaje pe *lifeline* pentru apelarea acelei operatii.

1. Din sectiunea de baza a *Modeling Palette* selectati pictograma **Synchronous Message**
2. Click pe elementul **User**, exact sub mesajul **getCashOnHand()**
3. Click din nou pe *lifeline*-ul **ATM** direct in dreapta primului click. Un mesaj si un mesaj *return* apar pe diagrama, ca in figura:



**Observatie:** Daca nu doriti afisarea mesajului *return* pe diagrama, click-dreapta in spatiul alb al editorului de diagrama. **Deselectati Show All Return Messages** din meniul aparut

4. Click-dreapta oriunde in editorul diagramei pentru deselectarea pictogramei **Synchronous Message**
5. Click-dreapta pe mesajul pe care tocmai l-ati creat si selectati **Operations > Add Operation** din meniul aparut
6. Scrieti **validateCashOnHand** si apasati **Enter**. IDE-ul eticheteaza mesajul pe diagrama si adauga mesajul ca o operatie in cadrul clasei **ATM**, in diagrama de clasa. In fereastra **Projects** extindeti nodul clasei **ATM** pentru verificarea listei de operatii, similara celei de mai jos



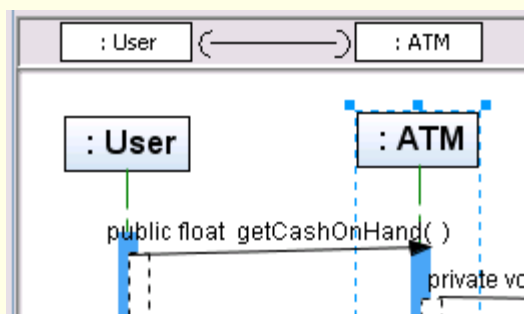
## 2.2.6. Utilizarea optiunii auto-exopandarii mesajelor

Pentru usurarea procesului de design, IDE-ul are o optiune de auto-expansiune a mesajelor. Cu auto-expansiunea selectata, cand un mesaj este plasat pe diagrama, spatiul este extins la latimea mesajului.

1. In *header*-ul gri de deasupra elementelor lifeline, click-dreapta in zona dintre numele elementelor User si ATM pentru setarea optiunii de auto-expansiune. Un meniu se deschide.

2. Selectati *Set Width to Message Width*

Un indicator apare in in spatiul dintre cele doua elemente, dupa cum este ilustrat in figura:



3. Repetati pasii pentru a seta optiunea pentru latimea mesajului intre elementele **ATM** si **Consortium** si intre **Consortium** si **Branch**

## 2.2.7. Utilizarea mesajelor care creeaza obiecte noi

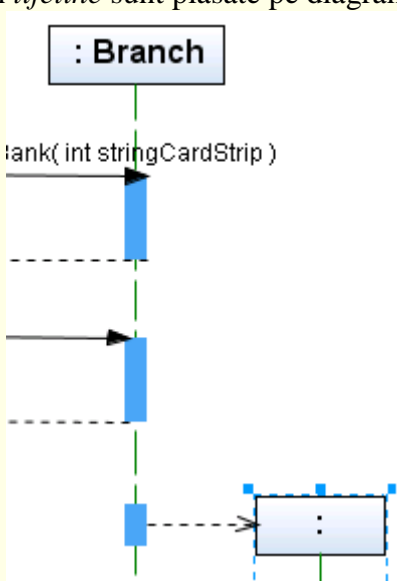
Notatia de creare mesaj permite crearea unui element lifeline sau a unei instante.

1. Din sectiunea de baza a *Modeling Palette*, selectati pictograma de *Create Message*

2. Click pe portiunea de jos a *lifeline*-ului **Branch**

3. Trasati legatura spre dreapta elementului **Branch** si click din nou

Un mesaj si un *lifeline* sunt plasate pe diagrama dupa cum este ilustrat in figura de mai jos :



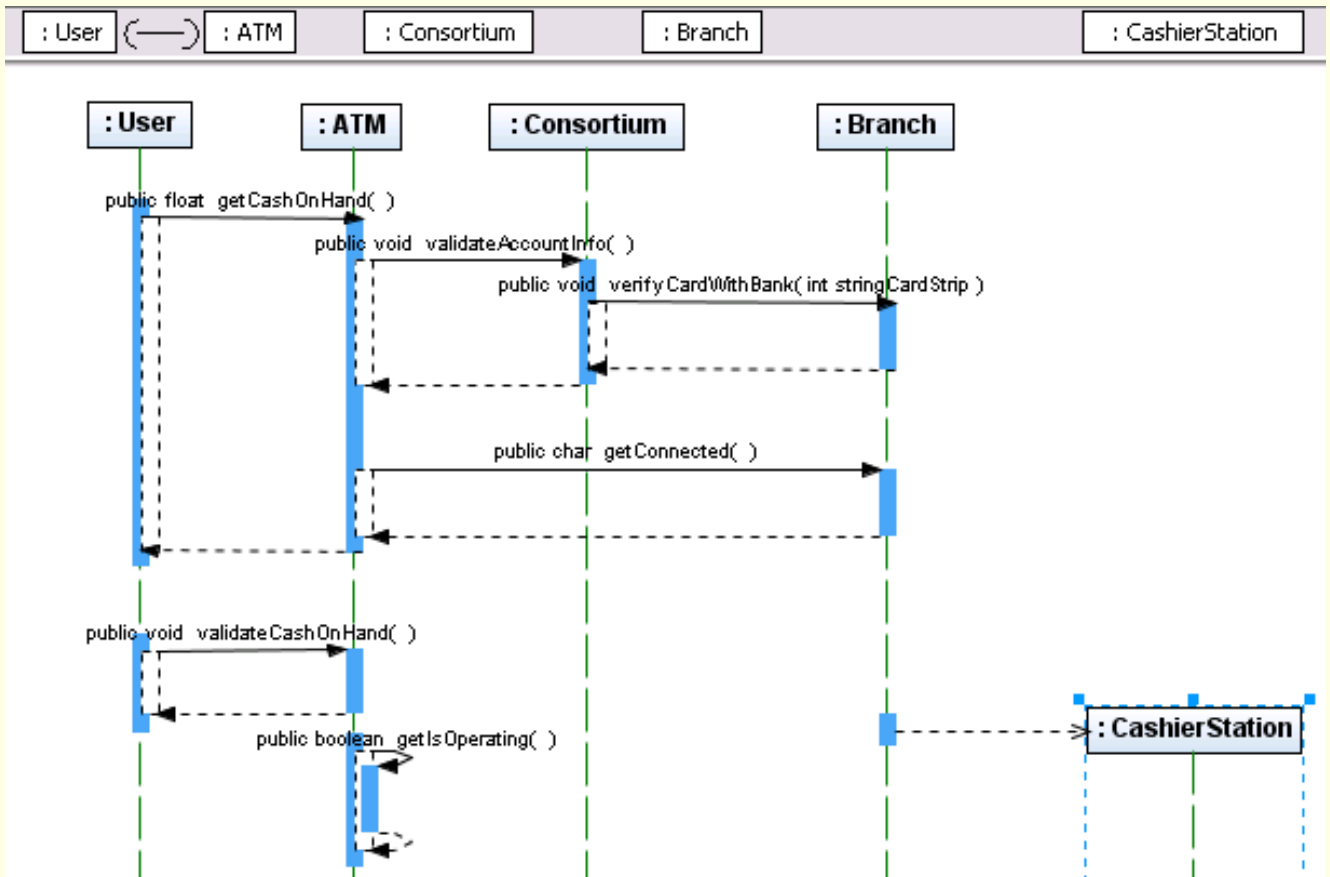
4. Apasati tasta *ESC* pentru a deselecta pictograma`

5. Selectati *lifeline*-ul nou.

Fereastra *Properties* afiseaza proprietatile pentru acest element *lifeline*

6. In fereastra **Properties**, pe randul etichetat **Representing Classifier**, click pe sageata care indica in jos. Apare o lista *drop-down*
7. Selectati **CashierStation** din lista *drop-down* si apasati **Enter**
8. Click in editorul diagramei pentru vizualizarea elementului *lifeline* etichetat

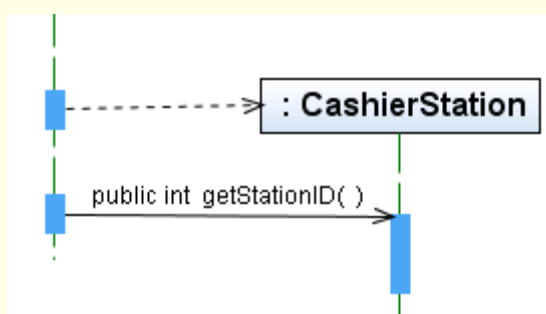
Diagrama ar trebui sa fie similara celei de mai jos



## 2.2.8. Adaugarea mesajelor asincrone


Un **mesaj asincron** reprezinta un mesaj care **nu blocheaza obiectul apelat**. Acest tip de mesaj este capabil sa creeze un nou obiect, sa creeze un nou fir de executie sau sa comunice cu un fir de executie deja existent.

1. Din sectiunea de baza a **Modeling Palette**, selectati pictograma **Asynchronous Message**
2. Desenati o legatura intre lifeline-urile **Branch** si **CashierStation** selectand elementul **Branch**, iar apoi elementul **CashierStation**
3. Apasati **ESC** pentru a deselecta pictograma
4. Click-dreapta pe noua legatura si alegeti **Operation** > **public int getStationID** din meniul aparut. Aceasta actiune adauga operatia mesajului, dupa cum este ilustrat in figura de mai jos

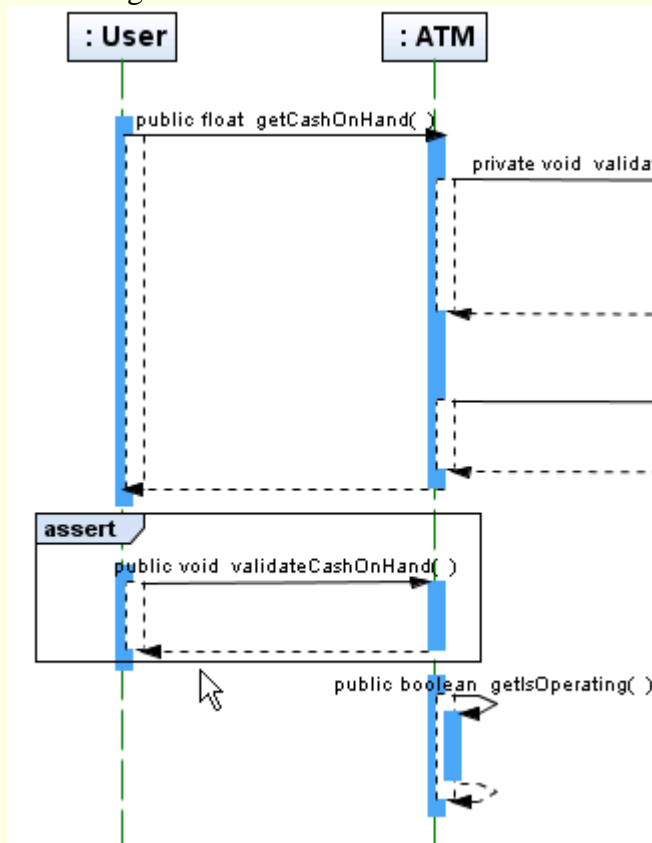


## 2.2.9. Utilizarea elementelor de control cu Combined Fragment

Fragmentul combinat permite exprimarea logica a alternativelor, optiunilor, exceptiilor, executiilor paralele, buclelor, negatiilor, regiunilor critice si a asertiilor, direct pe diagrama de secventa. Fragmentele combinate asigura o metoda de definire a **conditiilor speciale si subproceselor** pentru orice sectiuni sau *lifeline*-uri, specificand o zona unde conditiile sau subprocesele se aplica.

1. Din sectiunea **Control** a *Modeling Palette* selectati pictograma **Combined Fragment** 
2. Desenati elementul fragment combinat astfel incat sa incapsuleze mesajul **public void validateCashOnHand**, click in afara dreptunghiului punctat albastru, reprezentand mesajul, si click si trageți dreptunghiul pentru a inconjura mesajul.

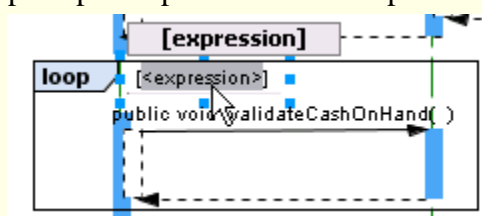
Atentie sa nu selectati nici unul din elementele *lifeline* cand dati click. Fragmentul combinat ar trebui sa fie similar celui din figura:



3. Apasati **ESC** pentru a deselecta pictograma
4. Click-dreapta pe eticheta **assert** declarata in fragmentul combinat si alegeti **Interaction Operator** > **loop**

**Observatie:** o bucla **Interaction Operator** arata ca fragmentul combinat reprezinta o bucla. De fiecare data cand o bucla se repeta, conditia este evaluata. Conditia poate include un numar specificat de repetari ale buclei.

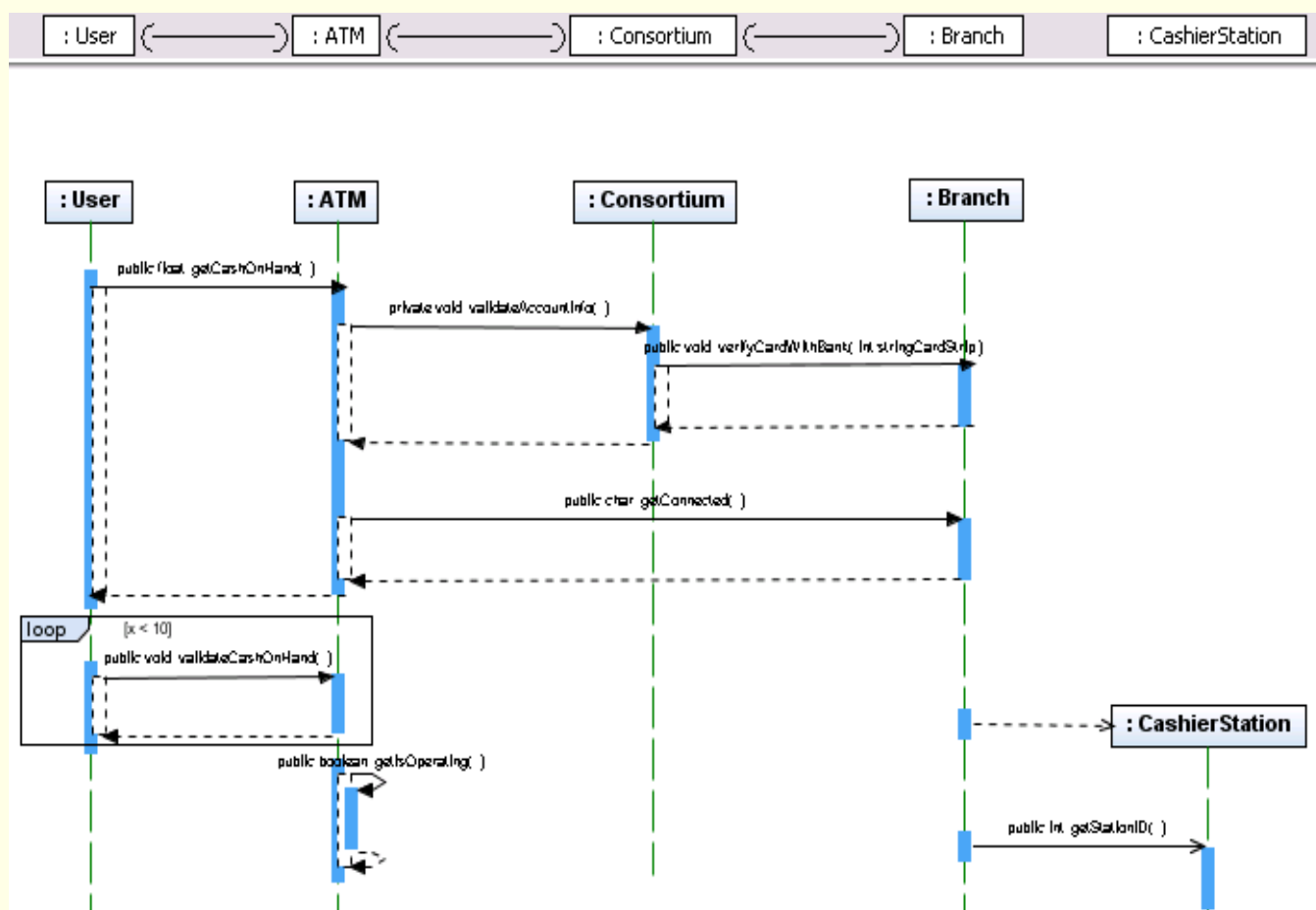
5. Click-dreapta pe eticheta buclei fragmentului combinat, selectati **Interaction Operand** > **Edit Interaction Constraint** din meniul aparut. Apare un camp expresie in fragmentul combinat
6. Dublu-click pe expresie pentru a face campul editabil, dupa cum este ilustrat in figura :



7. Scrieti **x < 10** si apasati **Enter**

8. Click in editorul diagramei. Expresia este acceptata si afisata in campul destinat.

Diagrama trebuie sa semene cu cea din figura:



## 2.2.10. Salvarea diagramei de secventa (MSC)

Dupa finalizarea diagramei de secventa, trebuie sa o salvati.

1. In editorul diagramei, click-dreapta pe tab-ul **SequenceDiagram**
2. Selectati **Save Document** din meniul aparut. Meniul se inchide si diagrama este salvata.

**Observatie:** sunteti intrebat daca doriti sa salvati sau sa renuntati la diagrama, cand iesiti din IDE.

## 2.2.11. Rezumat

In acest tutorial ati invatat sa creati o diagrama de secventa utilizand o diagrama de comunicatie (colaborare) deja existenta. Ati invatat sa efectuati urmatoarele:

- Sa generati o diagrama de secventa dintr-o diagrama de comunicatie (colaborare) existenta
- Sa adaugati mesaje diagramei utilizand pictograme din *Modeling Palette*
- Sa adaugati o legatura diagramei
- Sa adaugati elemente fragment combinat
- Sa salvati diagrama

## 2.3. Crearea diagramelor UML de activitati (organigrame)

In continuare veti invata cum sa folositi functiile UML ale IDE pentru a crea diagrame UML de activitati.

**Diagrama UML de activitati** este o forma de organigrama care permite reprezentarea activitatilor oricarui sistem si a fluxului de date si/sau de decizii intre activitati.

**Diagrama UML de activitati** este o forma de organigrama care poate fi utilizata pentru:

Descrierea activitatilor oricarui sistem si a fluxurilor de date si/sau decizii intre activitati

O vedere de ansamblu a proceselor aplicatiei (*business process*)

Descrierea activitatilor ce apar in interiorul unui caz de utilizare

Ilustrarea diferitelor tipuri de activitati utilizand diferite simboluri

Ilustrarea firelor paralele de executie

### 2.3.1. Crearea unui proiect UML independent de platforma

1. Pentru a crea un proiect UML, selectati **File > New Project** si apoi faceti urmatoarele:

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

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

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

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

3. Pentru campul **Project Location**, click **Browse**, si navigati 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**).

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

5. Faceti **Click** pe **Cancel**. IDE-ul va face urmatoarele

Creeaza un proiect de modelare **Platform-Independent** gol.

Afiseaza pictograma proiectului in fereastra **Project**.

### 2.3.2. Crearea unui pachet pentru diagrama de activitati

1. Pentru a crea un pachet pentru diagrama de activitati, right-click pe nodul **Model**, si selectati **New > Package** din meniul pop-up

2. In campul **Name** completati numele pachetului: **ActDiagPkg**. Acceptati valoarea implicita in campul **Namespace**

3. Bifati caseta **Create Scoped Diagram** In campul **Diagram Name** completati numele diagramei: **actDiagram**.

4. Din lista **Diagram Type** selectati **Activity Diagram** si apasati **Finish**. IDE-ul va face urmatoarele

Creeaza un nod pachet sub nodul **Model** avand numele ales anterior.

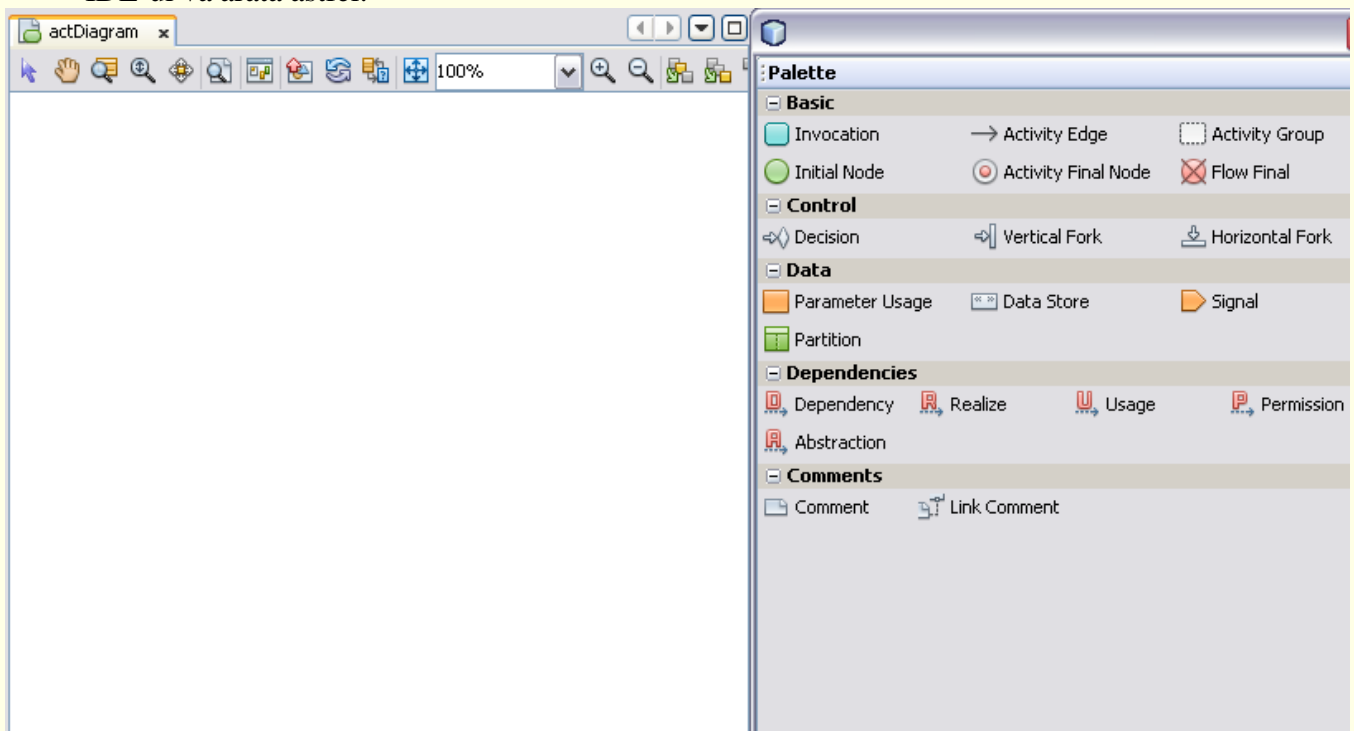
Creeaza un nod diagrama sub nodul packet

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

---

Deschide (*Modeling*) *Palette* si afiseaza pictogramele folosite pentru a construi diagrame de activitati.

IDE-ul va arata astfel:



### 2.3.3. Plasarea partiilor

IDE-ul permite **adaugarea unor partitii** (coridoare, *swimlanes*) **in diagrama de activitati**, acestea **divizand pe verticala** diagrama pentru a putea **reprezenta activitatile efectuate in paralel**.


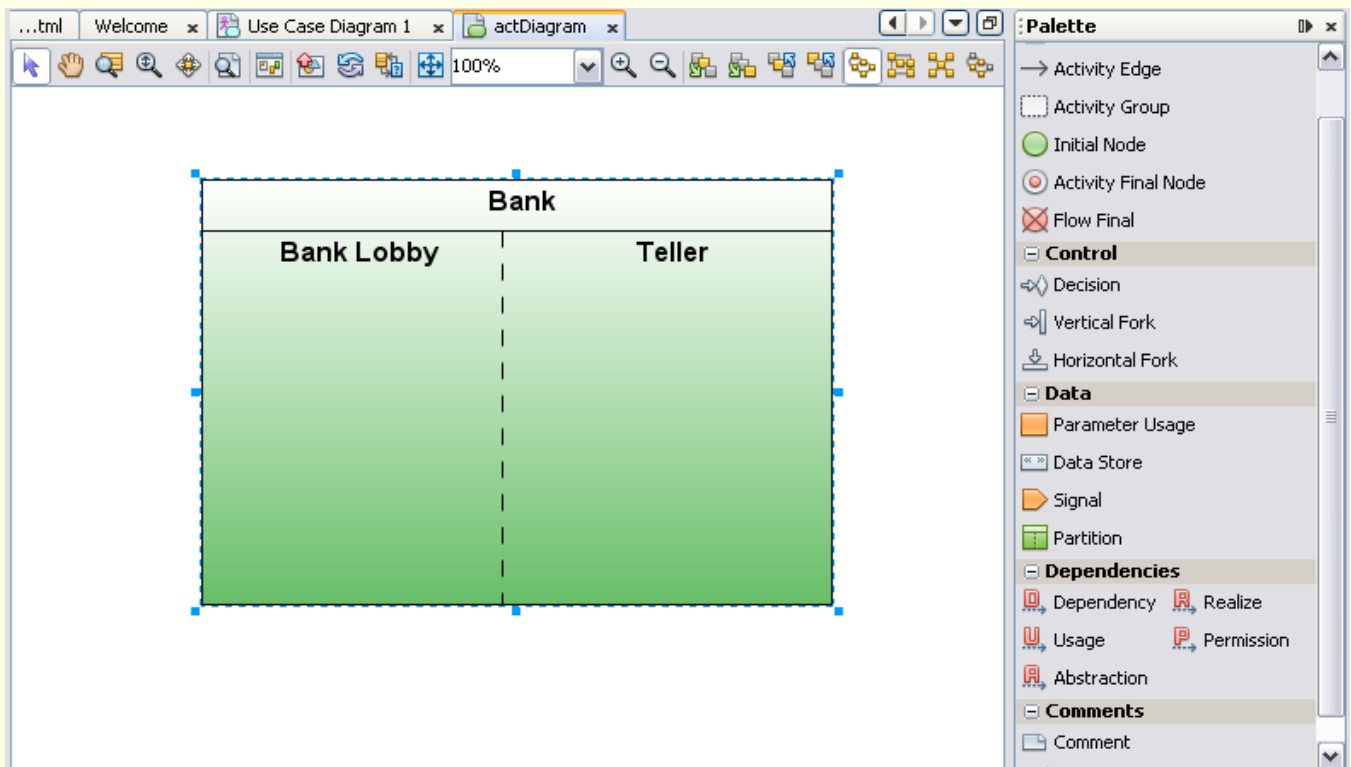
1. Din sectiunea **Data** a (*Modeling*) *Palette* selectati pictograma **Partition** .
2. Click in editorul diagramelor pentru a plasa elementul **Partition** in diagrama. Un element **Partition** fara nume este plasat in editorul diagramelor
3. Deselectati pictograma **Partition** prin right-click in interiorul editorului diagramelor
4. Largiti dreptunghiul elementului catre stanga. Cu elementul **Partition** selectat, right-click si selectati **Partitions > Add Partition Column to the Right** din meniul pop-up
5. Largiti dreptunghiul elementului catre dreapta si in jos.
6. Denumiti partitia prin dublu click pe cuvantul **Unnamed** aflat central deasupra si inlocuirea lui cu **Bank**. Apasati **Enter**.
7. Denumiti coloana din stanga a partitiei prin dublu click pe cuvantul **Unnamed** aflat in stanga si inlocuirea lui cu **Bank Lobby**.
8. Denumiti coloana din dreapta **Teller**.


Diagrama va arata astfel:




### 2.3.4. Adaugarea elementelor de tip nod

Aceasta sectiune contine urmatoarele proceduri:

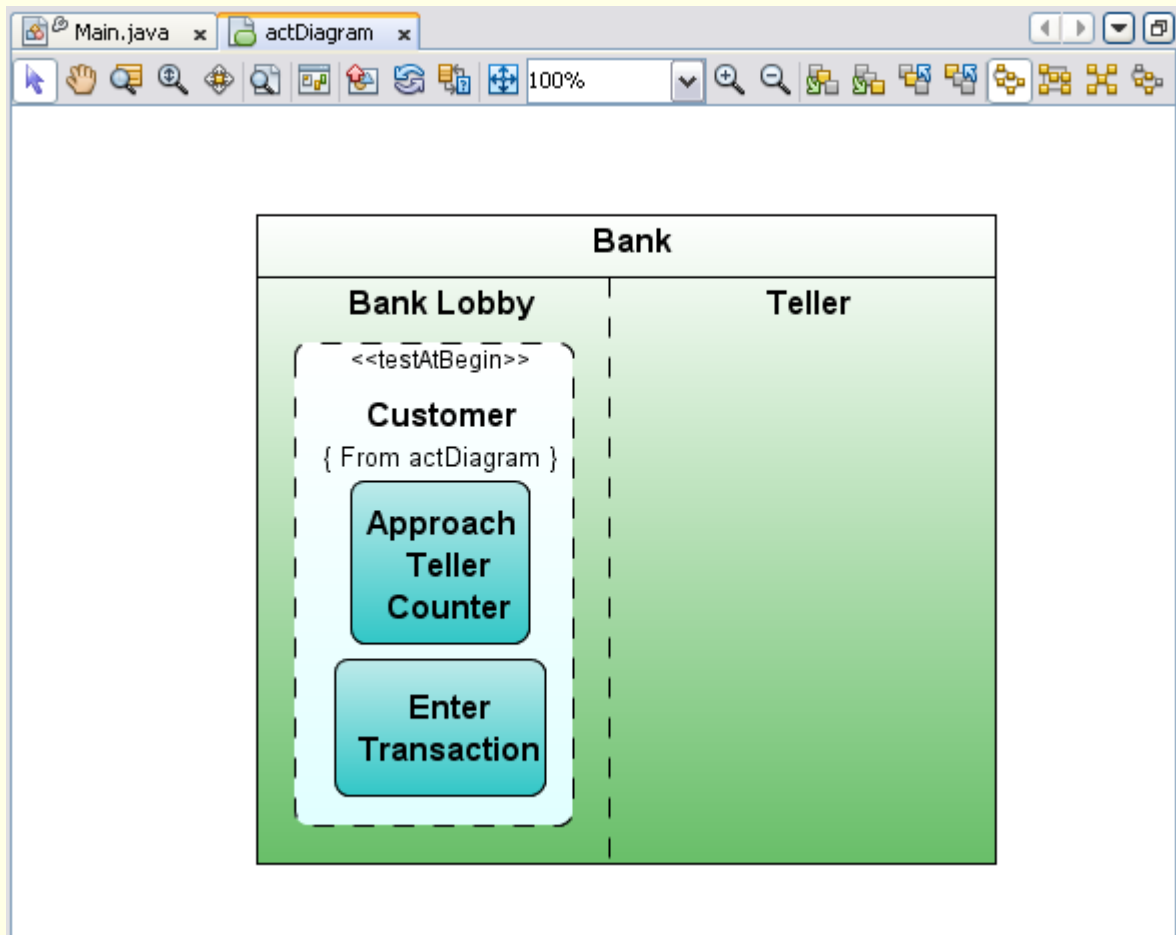
#### 2.3.4.1. Adaugarea unui grup de activitati

1. Din sectiunea **Basic** a **Pallete** selectati pictograma **Activity Group** .
2. Faceti click in subpartitia **Bank Lobby** pentru a plasa elementul **Activity Group** in subpartitia din stanga.
3. Deselectati pictograma **Activity Group**
4. Redenumiti elementul **Activity Group** cu dublu-click pe cuvantul **Unnamed** si inlocuirea lui cu **Customer**. Apasati **Enter**.
5. Selectati elementul **Activity Group** adaugat, apoi redimensionati-l pentru a ocupa aproape toata subpartitia din stanga

#### 2.3.4.2. Adaugarea unei invocari

1. Din sectiunea **Basic** a **Pallete** selectati pictograma **Invocation** .
2. Faceti click in elementul **Activity Group Customer** din subpartitia **Bank Lobby** pentru a plasa 2 elemente **Invocation** unul sub altul.
3. Deselectati pictograma **Invocation**
4. Selectati elementele invocare, apoi redimensionati-le pentru a se potrivi cat mai bine in elementul **Activity Group Customer** ca mai jos





5. Denumiti nodul invocare de deasupra prin dublu-click pe el si editarea textului **Approach Teller Counter**. Apasati *Enter*.

6. Denumiti nodul invocare de dedesupt **Enter Transaction**

7. Plasati inca 6 elemente invocare in interiorul subpartitiei **Teller** si denumiti-le astfel

**Receive Transaction Request**

**Search Customer Info**

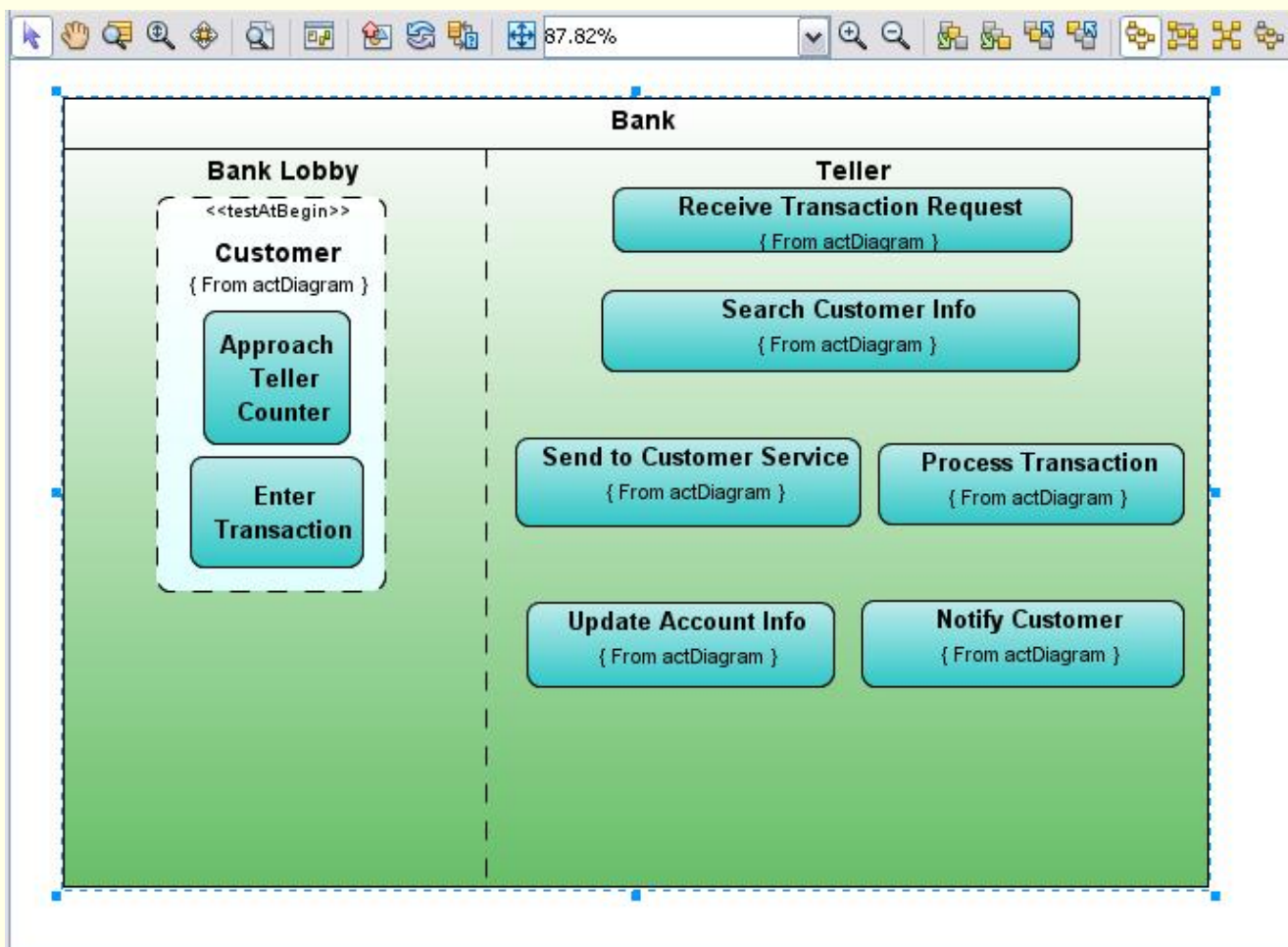
**Send to Customer Service**

**Process Transaction**


**Update Account Info**

**Notify Customer**


Diagrama ar trebui sa arate astfel:




### 2.3.4.3. Adaugarea unui element nod initial


1. Din sectiunea **Basic** a **Pallete** selectati pictograma **Initial Node** .
2. Faceti click in subpartitia **Bank Lobby** in stanga elementului **Approach Teller Counter**.
3. Deselectati pictograma

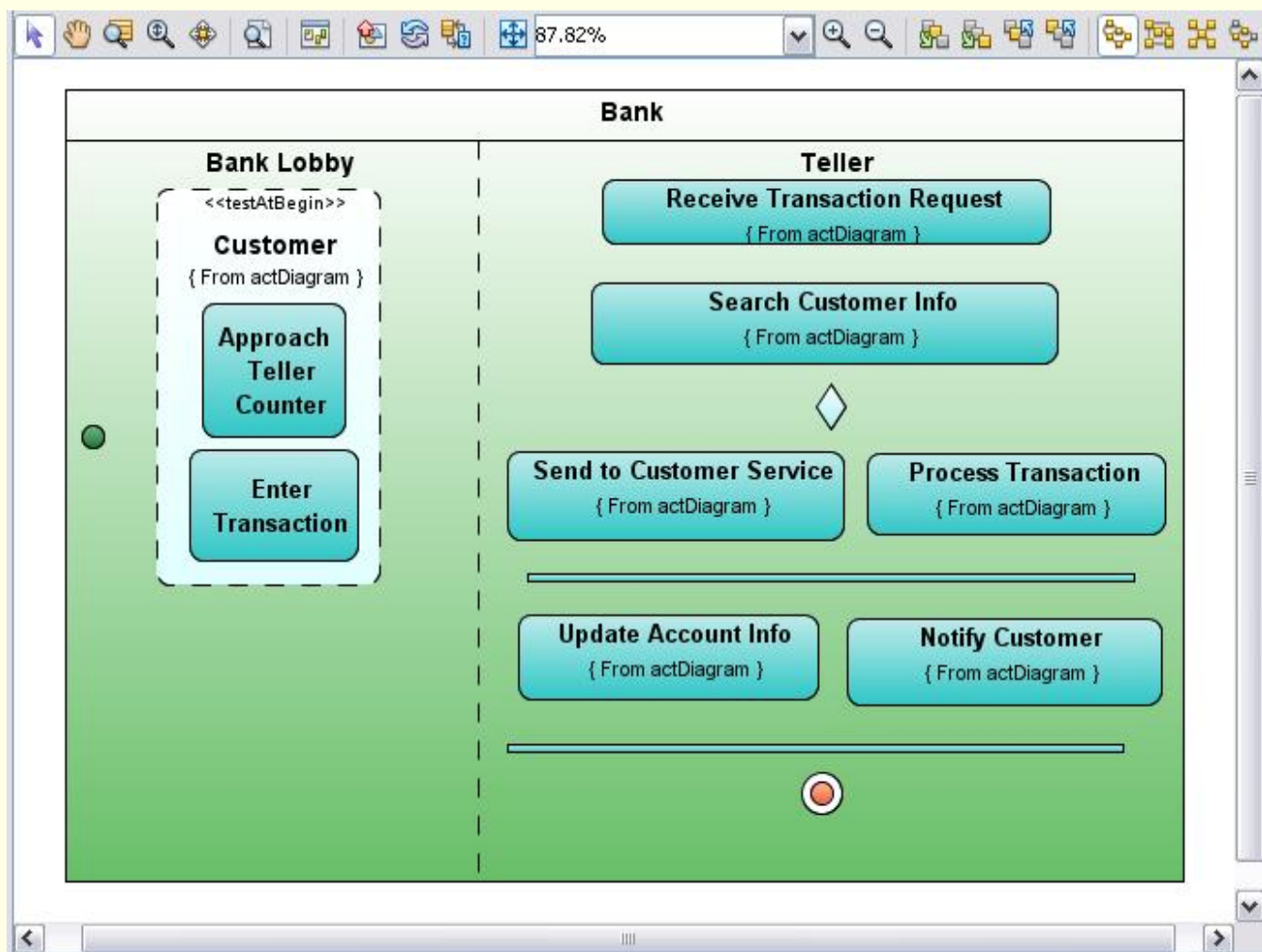
### 2.3.4.4. Adaugarea unui element bifurcatie orizontala

1. Din sectiunea **Control** a **Pallete** selectati pictograma **Horizontal Fork** .
2. Plasati bara reprezentand elementul **Horizontal Fork** deasupra elementelor invocare **Update Account Info** si **Notify Customer**
3. Deselectati pictograma
4. Lungiti bara pentru a acoperi latimea ambelor elemente invocare.
5. Plasati un alt element **Horizontal Fork** sub elementele invocare **Update Account Info** si **Notify Customer**
6. Lungiti bara pentru a acoperi latimea ambelor elemente invocare.

### 2.3.4.5. Adaugarea unui element nod activitate finala si a unui nod decizie

1. Din sectiunea **Basic** a **Pallete** selectati pictograma **Activity Final Node** .
2. Plasati elementul **Final Node** sub elementul **Horizontal Fork** aflat mai jos


3. Deselectati pictograma *Activity Final Node*
  4. Din sectiunea *Control* a *Pallete* selectati pictograma **Decision** .
  5. Plasati elementul *Decision* intre elementele **Send to Customer Service**, **Process Transaction** si **Search Customer Info**
  6. Deselectati pictograma
- Diagrama ar trebui sa arate astfel:



### 2.3.5. Adaugarea elementelor de tip legatura (tranzitie si dependenta)

Aceasta sectiune contine urmatoarele proceduri:

#### 2.3.5.1. Adaugarea unui element tranzitie intre activitati

1. Din sectiunea *Basic* a *Pallete* selectati pictograma **Activity Edge** . Veti folosi elementul *Activity Edge* pentru a conecta elementul *Initial Node* cu un element *Invocation*
2. Faceti click in elementul *Initial Node* si din nou click pe elementul *Invocation* cu numele **Approach Teller Counter**. O legatura *Activity Edge* uneste acum cele doua elemente. Etichetele pentru legaturile *Activity Edge* sunt ascunse dar trebuie sa fie afisate.
3. Deselectati pictograma *Activity Edge*
4. Selectati elementul *Activity Edge* si right-click pe el
5. Selectati din meniul pop-up *Labels* > *Show Name*. Legatura este acum etichetata cu **Unnamed**
6. Pentru a denumi legatura scrieti **Initiate Cash Withdrawal** si apasati *Enter*.

### 2.3.5.2. Adaugarea mai multor elemente tranzitie intre activitati

1. Din sectiunea *Basic* a *Pallete* selectati pictograma **Activity Edge**

2. Adaugati urmatoarele legaturi

De la **Approach Teller Counter** la **Enter Transaction**

De la **Enter Transaction** la **Receive Transaction Request**

De la **Receive Transaction Request** la **Search Customer Info**

De la **Search Customer Info** la nodul *Decision*

De la nodul *Decision* la **Send to Customer Service**

De la nodul *Decision* la **Process Transaction**

De la **Process Transaction** la bara *Horizontal Fork* de deasupra

De la bara *Horizontal Fork* de deasupra la **Update Account Info**

De la bara *Horizontal Fork* de deasupra la **Notify Customer**

De la **Notify Customer** la bara *Horizontal Fork* de dedesupt

De la **Update Account Info** la bara *Horizontal Fork* de dedesupt

De la **Lower Horizontal Fork** la elementul *Final State*

3. Deselectati pictograma *Activity Edge*

### 2.3.6. Lucrul cu conditii si grupuri

Aceasta sectiune contine urmatoarele proceduri:

#### 2.3.6.1. Adaugarea conditiilor logice la tranzitiile intre activitati

1. In editorul diagramelor right-click pe elementul *Activity Edge* aflat intre nodul *Decision* si elementul **Send to Customer Service**.

2. Selectati din meniul pop-up *Labels* > *Show Guard Condition*

3. In interiorul parantezelor drepte ale conditiei logice scrieti **No Customer Info** si apasati *Enter*.

4. Repetati pasii 1 si 2 pentru elementul *Activity Edge* aflat intre nodul *Decision* si elementul **Process Transaction**

5. Scrieti [ **Customer Info** ] pe post de conditia logica

#### 2.3.6.2. Selectarea si modificarea proprietatii GroupKind

IDE-ul reprezinta bucele din fluxurile de activitati ca elemente *Activity Group*. Exista 3 tipuri de elemente *Activity Group* diferite:

- *Iteration*,

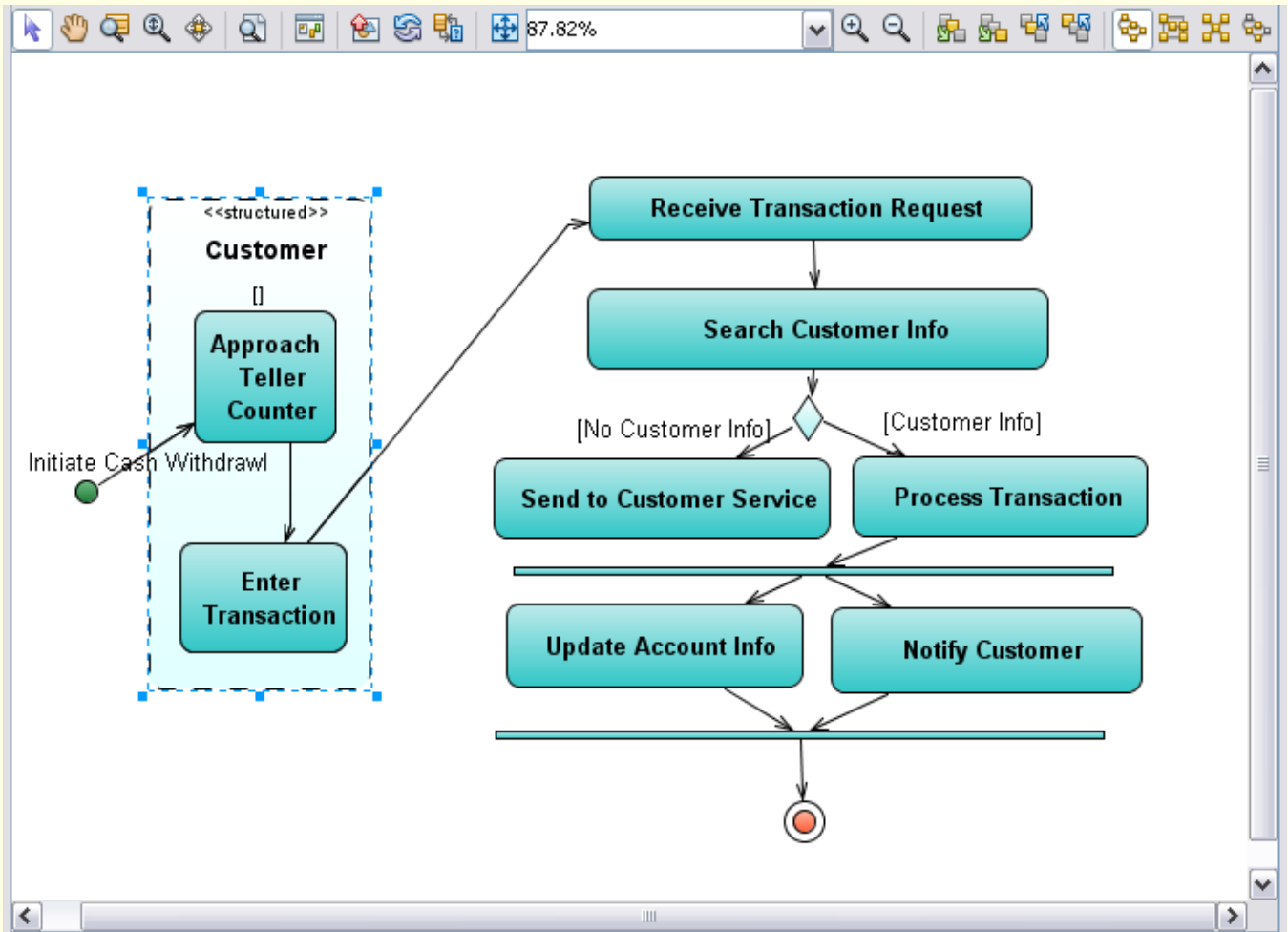
- *Structured* si

- *Interruptible*.

---

1. In editorul diagramelor right-click pe elementul *Activity Group* denumit **Customer**.
2. In fereastra *Properties* pe linia proprietatii *GroupKind* faceti click pe sageata in jos.
3. Selectati **Structured** din lista. Elementul *Activity Group* denumit **Customer** este reetichetat pe diagrama ca grup structurat.

Diagrama finala ar trebui sa arate astfel:



## 2.4. Teme pentru acasa

### 2.4.1. Enunt

Tema de casa este o continuare a temei de la laboratorul trecut, si are SASE parti:

(I) **constructia unei diagrame UML de comunicatie** (in NetBeans numita diagrama de colaborare) pornind de la

- **clasele din diagrama de clase** propusa in tema de la lucrarea anterioara
- **un caz de utilizare si textul naratiunii sale** (poate fi naratiunea realizata ca tema, sau naratiunea altui caz de utilizare)

**detaaliind la nivel de comportament intern scenariul din naratiune**

- **prin adaugarea de mesaje schimbate,**
- **si eventual adaugand noi clase**

(II) **constructia unei diagrame UML de secventa (MSC)** echivalenta cu diagrama de comunicatie creata in partea I

(III) **crearea unei extensii a cazului de utilizare utilizat in partea I si pe baza ei**

- **actualizarea diagramei de secventa (MSC) prin adaugarea de clase (minimum 1)**
  - **ceea ce are ca efect implicit, observabil, actualizarea diagramei de comunicatie**
- **crearea unei diagrame de clase noi care sa contina toate clasele si metodele**

(IV) **constructia unei diagrame UML de activitati**

- **pentru a descrie interactiunea actor-sistem**
  - **pornind de la textul naratiunii unui caz de utilizare** (poate fi naratiunea realizata ca tema, sau naratiunea altui caz de utilizare)

(V) **constructia unei diagrame UML de activitati**

- **pentru a descrie interactiunea interna sistemului**
  - **pornind de la scenariul descris prin diagrame de comunicatie si secventa (MSC) in lucrarea anterioara**

(VI) **generarea codurilor Java din diagrama de clase creata in lucrarea anterioara**

### 2.4.2. Exemplu de rezolvare

**Partea I-a** **Constructia unei diagrame UML de comunicatie** (in NetBeans numita diagrama de colaborare) pornind de la

- **clasele din diagrama de clase** propusa in tema de la lucrarea anterioara
- **un caz de utilizare din tema de la lucrarea anterioara si textul naratiunii sale**

**detaaliind la nivel de comportament intern scenariul din naratiune**

- **prin adaugarea de mesaje schimbate,**
  - **si eventual adaugand noi clase**
-

Posibila rezolvare:

Cazul de utilizare ales pentru a fi detaliat este **AccesServiciuBusiness**

Naratiunea cazului de utilizare AccesServiciuBusiness:

1. Numele cazului de utilizare  
**Acces Serviciu Business**
2. Scurta descriere a cazului de utilizare  
**Clientul lanseaza serviciul business iar sistemului ii pune la dispozitie serviciile componente**
3. Actori  
**Clientul**
4. Preconditii  
**Sistemul este aflat in executie**
5. Evenimentul care declanseaza cazul de utilizare  
**Clientul cere lansarea serviciului**
6. Descriere a interactiunii dintre actori si fiecare caz de utilizare
  1. **Clientul lanseaza serviciul prin intermediul unei Interfete oferite de sistem**
  2. **Interfata notifica sistemul (Cazul de utilizare este extins de cazul de utilizare Autorizare)**
  3. **Sistemul pune la dispozitia Clientului serviciile componente cerute**
7. Alternative la cazul de utilizare principal
  - E1. **Daca Autorizarea esueaza, serviciile nu sunt oferite iar Clientul este notificat asupra motivului**
8. Evenimentul care produce oprirea cazului de utilizare  
**Toate componentele au fost lansate**
9. Postconditii  
**Clientul capata poate utiliza serviciile componente**

Din text reiese necesitatea introducerii unei clase, pe care o putem numi deocamdata generic **InterfataClient**.

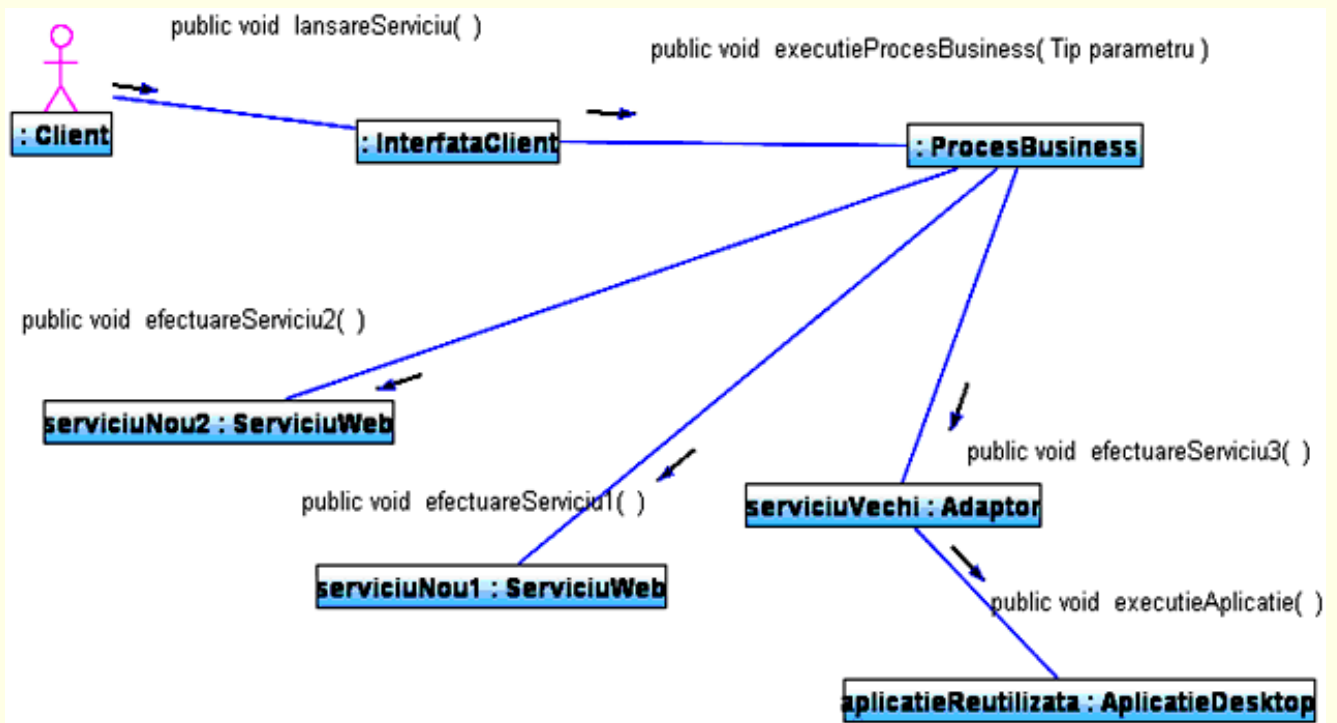
Detalierea scenariului folosind clasele sistemului:

1. **Clientul lanseaza serviciul prin intermediul entitatii InterfataClient**
2. **InterfataClient notifica/lanseaza ProcesBusiness (care ofera serviciul business agregat)**
3. **ProcesBusiness activeaza entitatile ServiciuWeb si Adaptor care ofera serviciile componente**
4. **Entitatea Adaptor lanseaza in executie AplicatieDesktop**

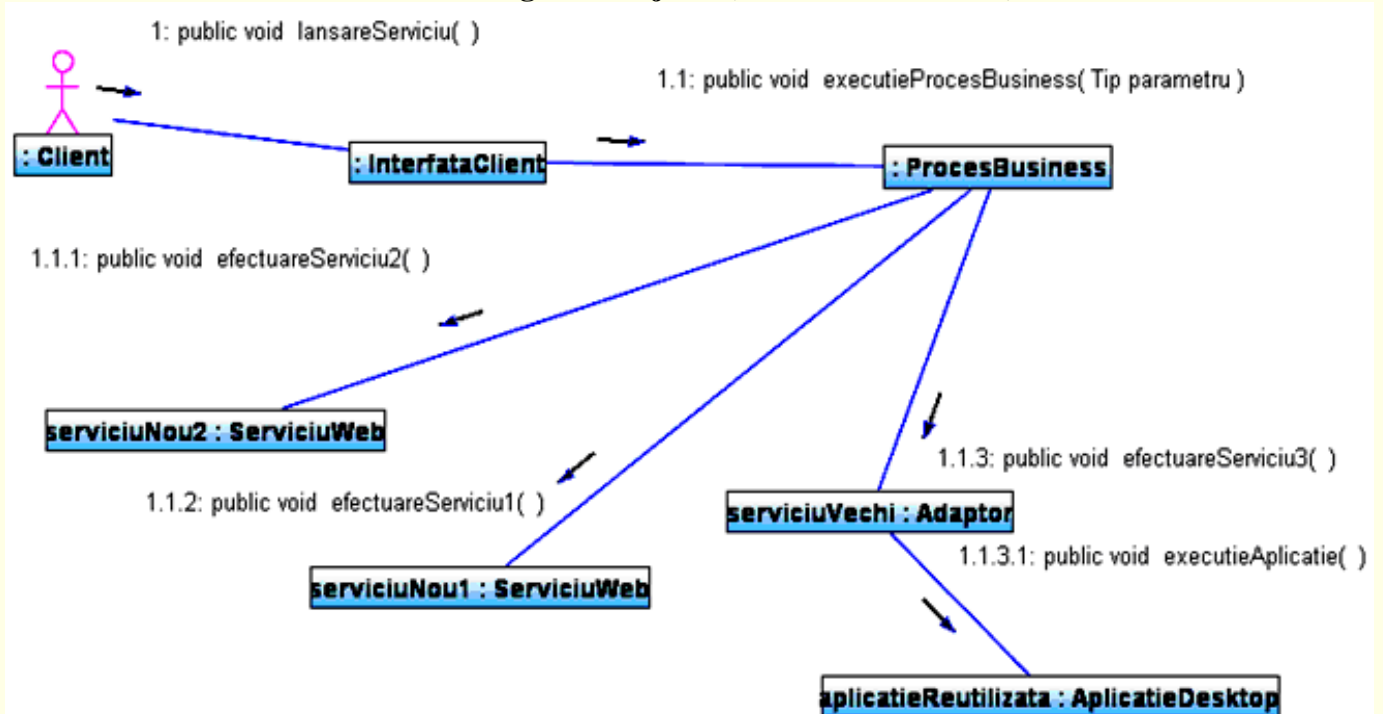
Detalierea scenariului adaugand mesaje si construind diagrama de comunicatie:

O posibila diagrama de comunicatie este urmatoarea:

---



Se observa existenta numelor mesajelor, inasa lipsa numerelor de ordine ale mesajelor. Diagrama de comunicare cu numere de ordine adaugate mesajelor (ca in sectiunea 3.1.5) este urmatoarea:

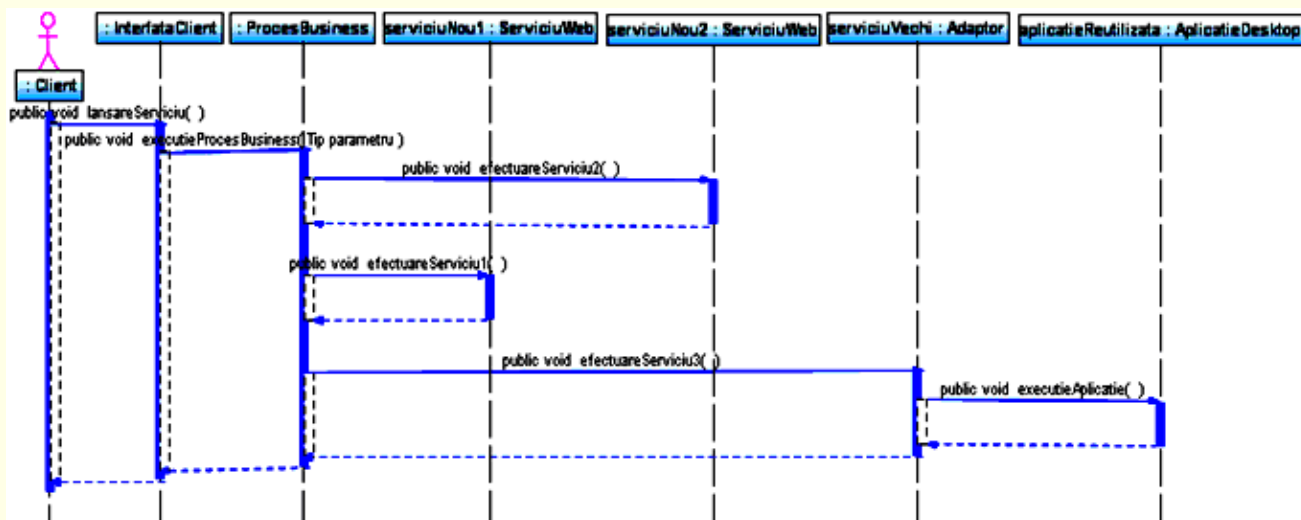


**Partea a II-a** Constructia unei diagrame UML de secventa (MSC) echivalenta cu diagrama de comunicare creata in partea I

Posibila rezolvare:

Generarea diagramei MSC utilizand pasii din sectiunea 3.2.1





### Partea III-a Crearea unei extensii a cazului de utilizare utilizat in partea I si pe baza ei

- actualizarea diagramei de secventa (MSC) prin adaugarea de clase
- si implicit actualizarea diagramei de comunicatie
- crearea unei diagrame de clase noi care sa contina toate clasele

Posibila rezolvare:

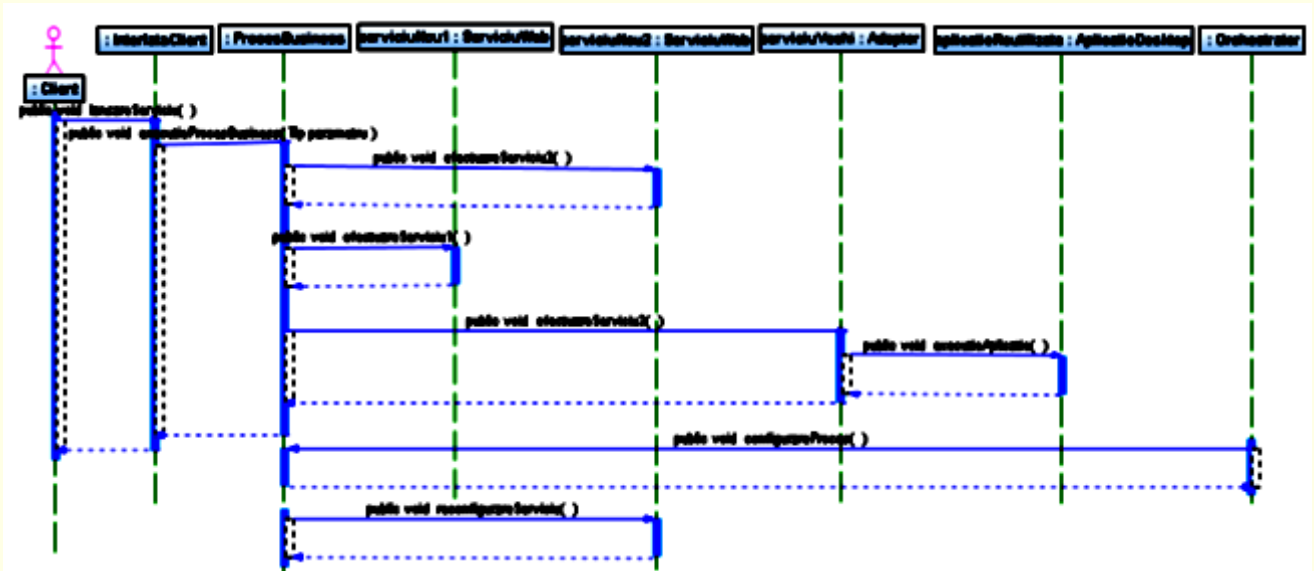
Cazul de utilizare **AccesServiciuBusiness** poate fi extins printr-o functionalitate de tipul **reconfigurare proces**, parte a cazului de utilizare **ExecutieProces**, ceea ce ar implica si clasa **Orchestrator**.

#### Detalierea noului scenariu:

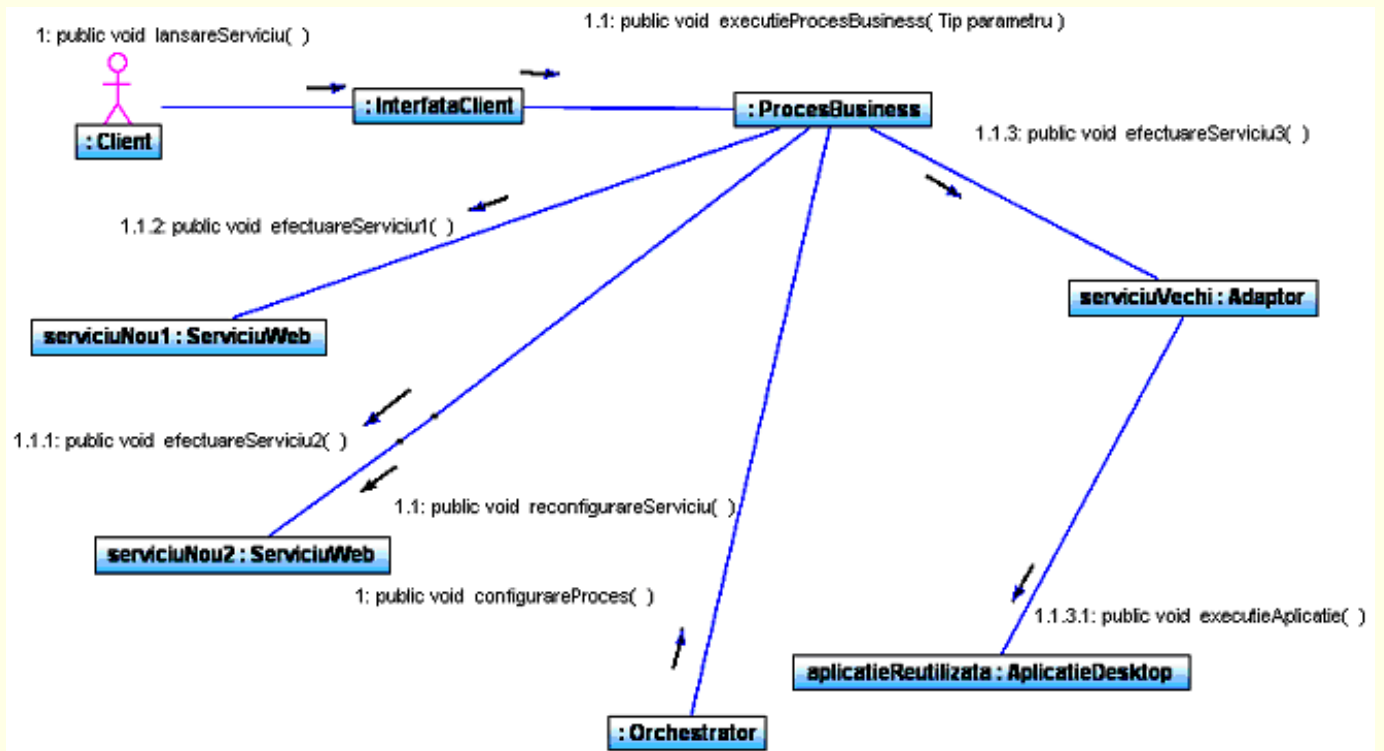
1. Clientul lanseaza serviciul prin intermediul entitatii InterfataClient
2. InterfataClient notifica/lanseaza ProcesBusiness (care ofera serviciul business agregat)
3. ProcesBusiness activeaza entitatile ServiciuWeb si Adaptor care ofera serviciile componentele
4. Entitatea Adaptor lanseaza in executie AplicatieDesktop
5. Entitatea Orchestrator poate interveni pentru reconfigurarea entitatii ProcesBusiness
6. Entitatea ProcesBusiness intervine atunci reconfigurand entitati ServiciuWeb si / sau Adaptor

#### Actualizarea diagramei MSC utilizand pasii din sectiunea 3.2.1

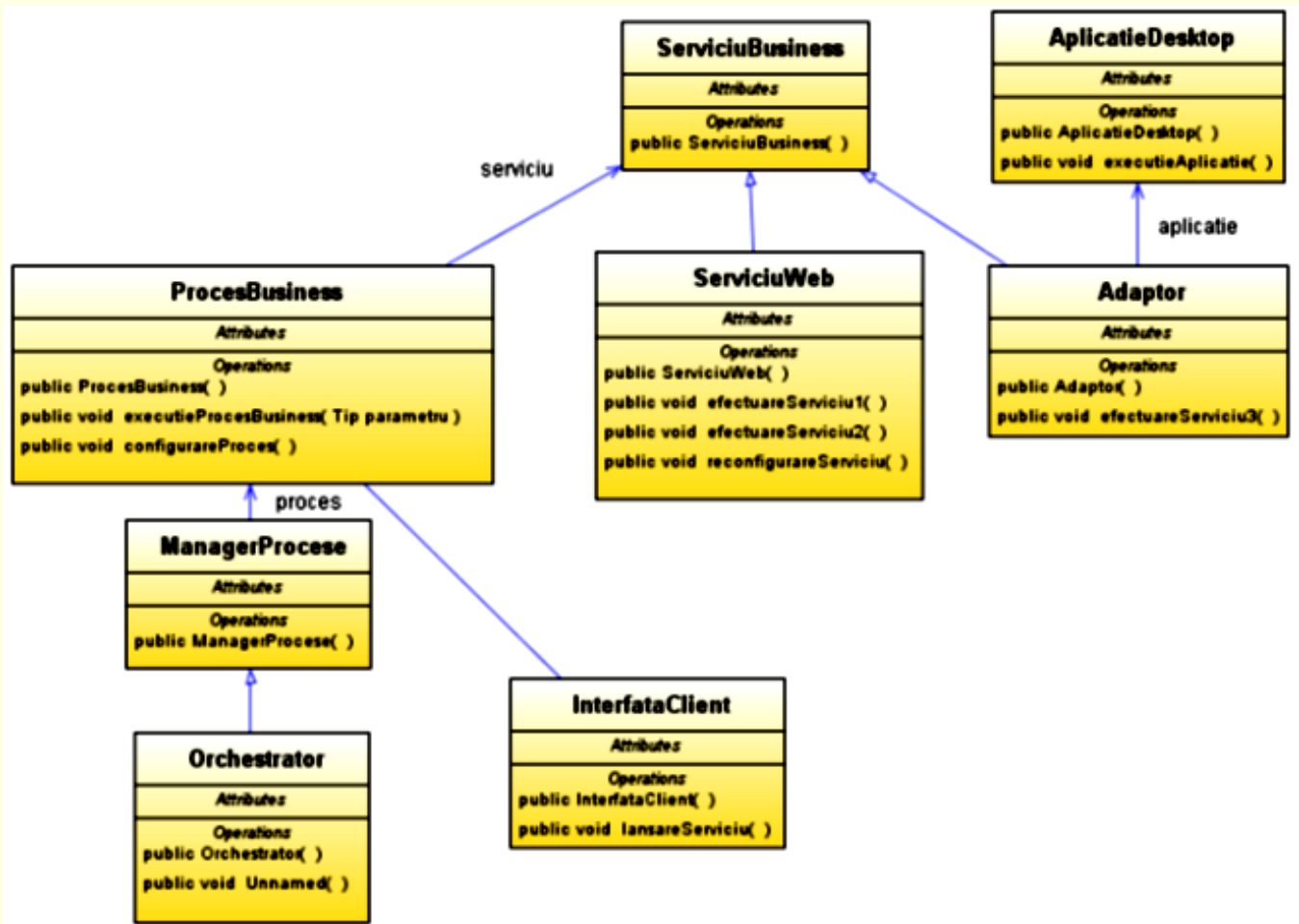
O posibila diagrama MSC actualizata este urmatoarea:



**Diagrama de comunicatie actualizata (implicit de catre NetBeans)**



**Diagrama de clase actualizata creata pornind de la toate clasele existente in proiect este urmatoarea:**



Se observa aparitia metodelor create mai intai pe prima versiune a diagramei de comunicatie si apoi pe a doua versiune a diagramei de secventa (MSC).

Noua clasa a impus crearea unei noi asocieri, cu clasa ProcessBusiness, pentru a corespunde diagramei de comunicatie.

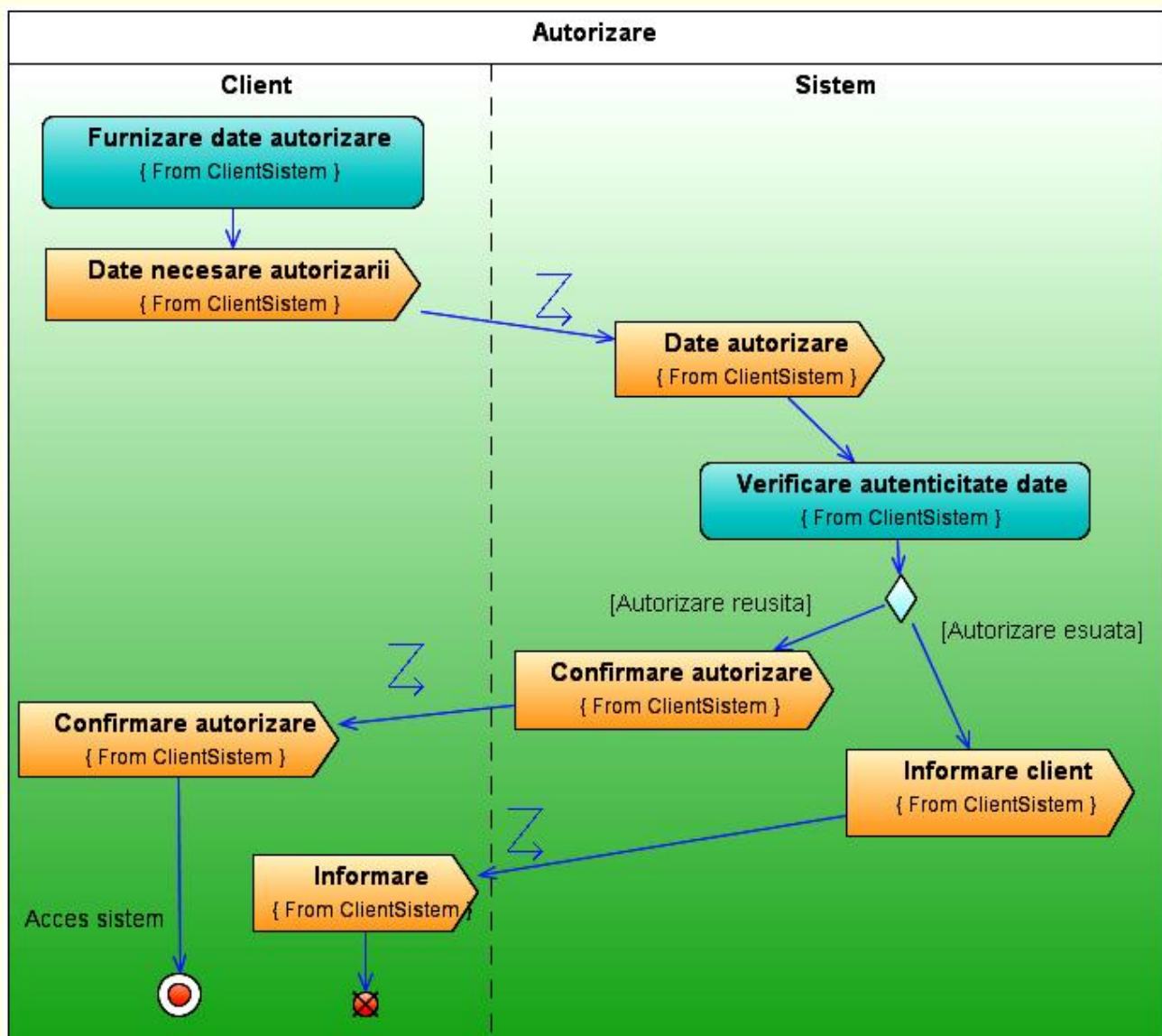
#### Partea a IV-a Constructia unei diagrame UML de activitati

- pentru a descrie interactiunea actor-sistem pornind de la textul naratiunii cazului de utilizare Autorizare (din Lucrarea a 2-a)

Pornim de la naratiunea cazului de utilizare Autorizare, care include

- Descrierea 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
  3. Sistemul prezinta un mesaj de confirmare catre Client.
  4. Clientul capata acces la sistem.
- Alternative la cazul de utilizare principal
  - E2. Daca sistemul nu cunoaste datele de autorizare, autorizarea esueaza.
  - E3. Daca Clientul transmite eronat datele de autorizare, autorizarea esueaza.

Posibila rezolvare:



Se observa **utilizarea simbolurilor semnal (Signal)** utilizate pentru a specifica detaliat schimbul de mesaje între entitățile descrise în cele 2 partiții.

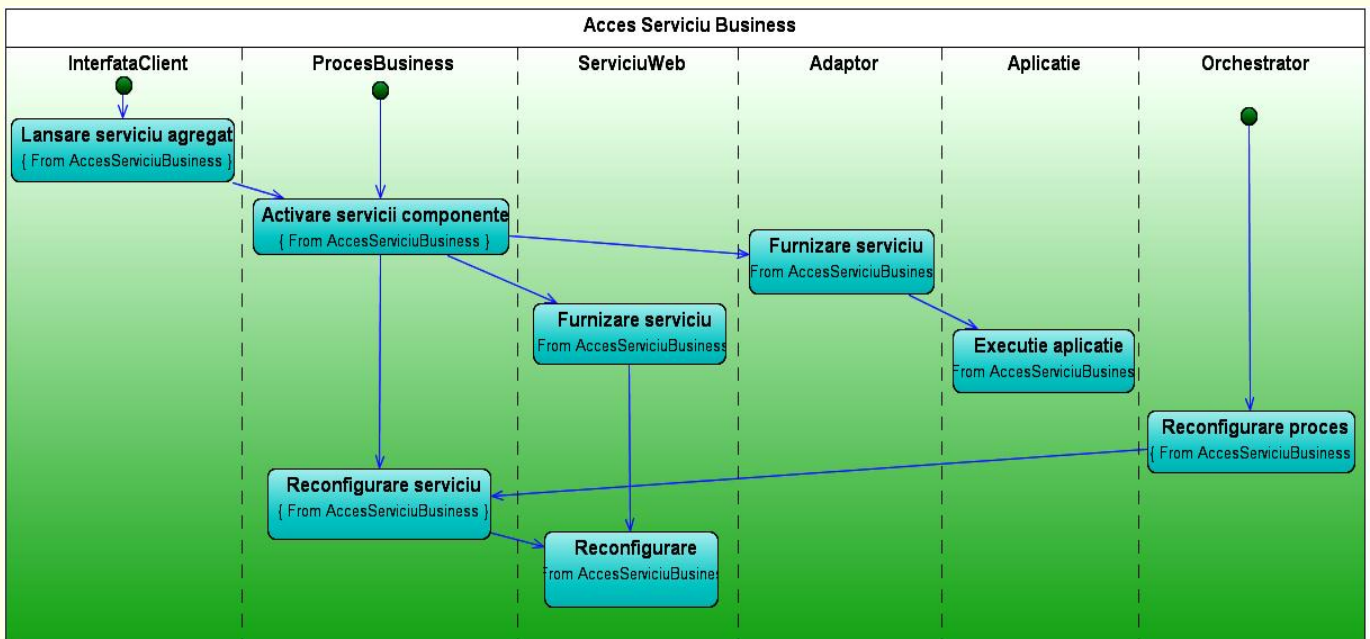
### Partea a V-a Constructia unei diagrame UML de activitati

- pentru a descrie interactiunea internă sistemului pornind de la scenariul descris prin diagrame de comunicare și secvență (MSC) în lucrarea anterioară

Pornim de la detalierea scenariului cazului de utilizare **AccesServiciuBusiness** extins prin funcționalitatea **reconfigurare proces** (parte a cazului de utilizare **ExecutieProces**):

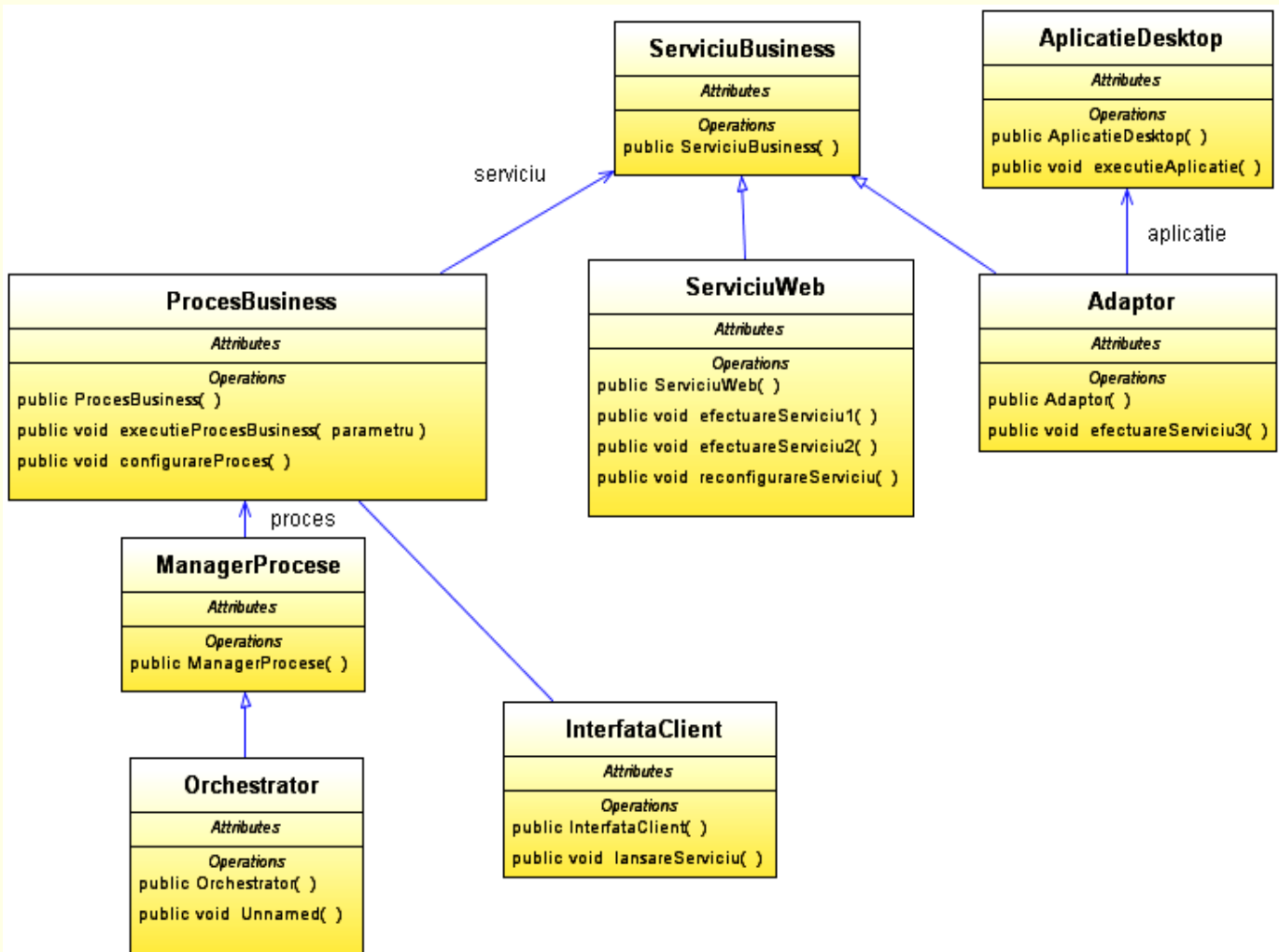
1. Clientul lansează serviciul prin intermediul entității InterfataClient
2. InterfataClient notifică/lansează ProcesBusiness (care oferă serviciul business agregat)
3. ProcesBusiness activează entitățile ServiciuWeb și Adaptor care oferă serviciile componente
4. Entitatea Adaptor lansează în execuție AplicatieDesktop
5. Entitatea Orchestrator poate interveni pentru reconfigurarea entității ProcesBusiness
6. Entitatea ProcesBusiness intervine atunci reconfigurând entități ServiciuWeb/Adaptor

Posibila rezolvare:



Se observa de data aceasta **utilizarea exclusiv a simbolurilor activitate (Invocation)** pentru a simplifica diagrama.

**Partea a VI-a Generarea codurilor Java din diagrama de clase creata in lucrarea anterioara**



**Rezolvare:**

```
1 public class InterfataClient {
2
3     public InterfataClient () {
4     }
5     public void lansareServiciu () {
6     }
7 }
```

```
1 public class ProcesBusiness {
2
3     private ServiciuBusiness serviciu;
4
5     public ProcesBusiness () {
6     }
7
8     public void executieProcesBusiness (Tip parametru) {
9     }
10
11     public void configurareProces () {
12     }
13 }
```

```
1 public class ServiciuBusiness {
2
3     public ServiciuBusiness () {
4     }
5 }
```

```
1 public class ServiciuWeb extends ServiciuBusiness {
2
3     public ServiciuWeb () {
4     }
5
6     public void efectuareServiciu1 () {
7     }
8     public void efectuareServiciu2 () {
9     }
10    public void reconfigurareServiciu () {
11    }
12 }
```

```
1 public class Adaptor extends ServiciuBusiness {
2
3     private AplicatieDesktop aplicatie;
4
5     public Adaptor () {
6     }
7
8     public void efectuareServiciu3 () {
9     }
10 }
```

```
1 public class AplicatieDesktop {
2
3     public AplicatieDesktop () {
4     }
5
6     public void executieAplicatie () {
7     }
8 }
```

```
1 public class ManagerProcese {
2
3     private ProcesBusiness proces;
4
5     public ManagerProcese () {
6     }
7 }
```

```
1 public class Orchestrator extends ManagerProcese {
2
3     public Orchestrator () {
4     }
5 }
```