Univerzitet u Kragujevcu

Fakultet inženjerskih nauka



**Softverski inženjering**

Seminarski rad:

Bankomat (ATM)

Student: Predmetni nastavnik:

Lazar Dašić 652/2017 Prof. Dr. Nenad Filipović

**Sadržaj**

**1 Postavka zadatka i detaljan opis aplikacije . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3**

**2 Opis delova programa sa samim izvornim kodom. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4**

2.1 Uvod . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4

2.2 Datoteka „Kartica.java“ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

2.3 Datoteka „Pocetno.java“ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

2.4 Datoteka „UnosKartice.java“ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

2.5 Datoteka „IzborUsluge.java“ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

2.6 Datoteka „PodizanjeNovca.java“ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

2.7 Datoteka „StampanjeIzvestaja.java“ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9

2.8 Datoteka „ATM.java“ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

2.9 Datoteka „main.java“ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13

**3 UML dijagrami . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .** **. . . . . . . . . 13**

3.1 Dijagram slučajeva korišćenja . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13

3.2 Dijagram klasa . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14

3.3 Dijagram sekvenci . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15

3.4 Dijagram aktivnosti . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18

3.5 Dijagram stanja . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20

**4 Literatura . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22**

**1 Postavka zadatka i detaljan opis aplikacije**

Napisati aplikaciju za simulaciju rada bankomata. Korisnik interaguje sa sistemom pomoću komandne linije i interaktivnog menija, koji omogućava pokretanje svih dostupnih funkcionalnosti.

**Bankomat (ATM)** ima serijski broj, naziv banke i lokaciju i funkcioniše kao automat stanja. Automat stanja je definisan kao graf prototipskih objekata stanja koji se pravi prilikom stvaranja bankomata. Bankomat ima informaciju o svom trenutnom stanju. Ne sme postojati više od jednog bankomata.

**Platna kartica** ima jedinstveni serijski broj, ime klijenta i trenutni iznos sredstava (nenegativan ceo broj). Pravi se zadavanjem imena klijenta, serijski broj joj se automatski generiše. Može da se dopuni zadatim iznosom i mogu da joj se umanje sredstva za zadati iznos. Pokušaj podizanja više novca od iznosa dostupnog na kartici je neregularna situacija i signalizira se izuzetkom *WrongAmmountException*.

**Stanje** se pravi zadavanjem naziva uvek kao prototipsko (isPrototype=true). Postoje sledeća stanja: početno stanje, unos kartice, izbor usluge, štampanje izveštaja kartice, podizanje novca. Poziv bilo koje operacije za prototipsko stanje rezultuje podizanjem izuzetka IllegalStateException.

*U početnom stanju* automat čeka na korisnika. Pri ulasku u stanje na ekranu se prikazuje poruka dobrodošlice. Stanje se izvršava tako što se čeka da korisnik inicira interakciju. Ako se unese 1, prelazi se u stanje za učitavanje kartice, a u slučaju 0 završava se program. Prilikom završetka programa, bankomat se ispisuje u fajl čije ime ime format: *atm\_<timestamp>.out* .

*U stanju „unos kartice“* pri ulasku u stanje, korisniku se izlistaju sve kartice koje su već napravljene u programu, označene rednim brojevima. Stanje se izvršava tako što se čeka na unos rednog broja kartice. Ukoliko je redni broj pogrešan, ispisuje se poruka o grešci i “prelazi“ u isto stanje. Predvideti opciju da korisnik odustane od učitavanja kartice, pri čemu automat prelazi u početno stanje. Posle učitanja kartice, automat prelazi u stanje “izbor usluge“. Ishod stanja je serijskih broj učitane kartice.

*U stanju “izbor usluge”* korisniku se prikazuju osnovne informacije o kartici (ime klijenta i broj kartice), a zatim spisak dostupnih usluga (označenih rednim brojevima). Ponuđene opcije su: podizanje novca, štampanje izveštaja o kartici, prekid rada. Stanje se izvršava čekanjem na unos rednog broja usluge i prelaskom u odgovarajuće stanje. Ishod stanja je redni broj odabrane usluge.

*U stanju “štampanje izveštaja”* o kartici prikazuje se poruka “štampanje izveštaja u toku”. Stanje se izvršava štampanjem izveštaja u fajl (*report\_<timestamp>.out*): ime vlasnika, broj kartice, datum i vreme štampanja, raspoloživa sredstva. Po završetku štampanja, prelazi se u stanje za izbor usluge. Ishod stanja je putanja do fajla u koji je sačuvan izveštaj.

Predvideti neke izuzetke (neispravne unose itd.). Obezbediti ispravno brisanje čvora. Preporučuje se razvojno okruženje programskog jezika Visual C++ 6.0 ili Visual Studio (verzije sve do 2015).

1. **Opis delova programa sa samim izvorim kodom**

**2.1 Uvod**

Za izradu ovog projektnog zadatka korišćeno je razvojno okruženje(IDE) IntelliJ IDEA, a za implementaciju je korišćen programski jezik Java.

U postavci zadatka je rečeno da Bankomat radi kao automat stanja. To znači da su sve funkcionalnosti koje korisnik može iskoristite zavisne od stanja u kome se Bankomat u tom trenutnku nalazi. Automat stanja će promeniti trenutno stanje u zavisnosti od ulazne vrednosti zadate od strane korisnika. Promena stanja se naziva tranzicija. Dakle automat stanja je definisan svojom listom stanja, svojim početnim stanjem i ulaznim vrednostima koje dovode do tranzicije.

Zbog osobina automata stanja, kao veoma korisna stvar za implementaciju automata se pokazao dizajn šablon „State Design Pattern“. Međutim, kako su dizajn šabloni tema predmeta Softverski inženjering 2, odlučeno je da se iskoristi suštinska ideja ovog šabloni i na adekvatan način „emulira“ rad automata stanja urađen korićenjem tog šablona.

U izradi projekta korišćeno je par biblioteka iz različitih paketa koje su bile neophodne za rad:

1. **FileWriter**, iz paketa java.io, neophodna radi pravljenja datoteka i upisa u njih(*report\_<timestamp>.out* i *atm\_<timestamp>.out*).
2. **IOException**, iz paketa java.io, neophodna za rad sa mogućim greškama pri radu sa datotekama
3. **SimpleDateFormat**, iz paketa java.text, radi formatiranja timestamp-a tako da on može biti u nazivu datoteke
4. **Date**, iz paketa java.util, neophodna radi pravljenja timestamp-a
5. **Scanner**, iz paketa java.util, neophodna za učitavanje korisničkog izbora
6. **ArrayList**, iz paketa java.util, neophodna za čuvanje svih kartica sa kojima Bankomat može da radi, kao i za čuvanje istorije stanja kroz koje Bankomat prolazi prilikom rada sa jednim korisnikom
7. **Random**, iz paketa java.util, neophodna za generisanje serijskih brojeva platnih kartica
   1. **Datoteka „Kartica.java“**

U ovoj datoteci smeštene se dve klase: Kartica i WrongAmmountException. Klasa Kartica predstavlja entitet **Platna Kartica** iz postavke zadatka koja zadovoljava sve što je od nje traženo u postavci. Klasa WrongAmmountException nasleđuje klasu Exception i predstavlja izuzetak koji će biti signaliziran kada je pokušano podizanje veće sume novca od one koja je dostupna na kartici.

**import** java.util.ArrayList;  
**import** java.util.Random;  
  
**public class** Kartica {  
  
 **private** String **ime**;  
 **private** String **prezime**;  
 **private** String **serBroj**;  
 **private int sredstva**;  
  
 **public** Kartica(String ime, String prezime, **int** sredstva,ArrayList<Kartica> sveKartice) {  
  
 **this**.**ime** = ime;  
 **this**.**prezime** = prezime;  
 **this**.**sredstva** = sredstva;  
 *//sveKartice predstavljaju sve kartice sa kojim Bankomat vec moze da radi(odnosno koje se nalaze u klasi ATM)* Random rng=**new** Random();  
 *//kartica je jedinstveni broj od 16 cifara* **int** duzina=16;  
 **boolean** nijeJedinstven;  
 *//broj koji generisemo* String genSerBroj;  
  
 **do**{  
 genSerBroj = **""**;  
 **for** (**int** c = 0; c < duzina; c++) {  
 *//generisanje int od 0 do 9, stavljanje u wrapper class Integer i pozivanje metode toString* genSerBroj += ((Integer)rng.nextInt(10)).toString();  
 }  
  
 *// provera da li je novo generisani broj jedinstven* nijeJedinstven = **false**;  
 **for** (Kartica k : sveKartice) {  
 **if** (genSerBroj.compareTo(k.getSer()) == 0) {  
 nijeJedinstven = **true**;  
 **break**;  
 }  
 }  
  
  
 }**while**(nijeJedinstven);  
  
 **this**.**serBroj**=genSerBroj;  
  
 }  
  
 **public** String getIme() {  
 **return ime**;  
 }  
  
 **public** String getPrezime() {  
 **return prezime**;  
 }  
  
 **public** String getSer(){  
 **return this**.**serBroj**;  
 }

**public void** dopuniSredstva(**int** novac){

**this**.**sredstva**+=novac;  
 System.***out***.println(**"Uplaćen novac na karticu, trenutno stanje: "**+**this**.**sredstva**);  
 }  
  
 **public boolean** podizanjeSredstva(**int** novac){  
 **try**{  
 **if**(**this**.**sredstva**-novac<0)  
 **throw new** WrongAmmountException(**this**.**sredstva**);  
 **else**{  
 **this**.**sredstva**-=novac;  
 System.***out***.println(**"Trenutno stanje na kartici: "**+**this**.getSredstva());  
 **return true**;}  
 }**catch**(Exception e){  
 System.***out***.println(e);  
 **return false**;  
 }  
 }  
  
 **public double** getSredstva() {  
 **return sredstva**;  
 }  
}  
  
**class** WrongAmmountException **extends** Exception{  
 **private double stanje**;  
 WrongAmmountException(**double** sredstva){  
 **this**.**stanje**=sredstva;  
 }  
 **public** String toString(){  
 **return**(**"Nedovoljno sredstava na računu, trenutno stanje: "**+**this**.**stanje**);  
 }  
}

* 1. **Datoteka „Pocetno.java“**

U ovoj datoteci nalazi se klasa Pocetno koja predstavlja „početno stanje“ u kome se Bankomat nalazi na početku rada. Sadrži konstruktor kome se prosleđuju informacije o Bankomatu(naziv banke, lokacija i serijski broj). Sadrži funkciju printWelcome() koja štampa poruku dobrodošlice i pita korisnika da se odluči za dalje akcije(0- završi program i 1-prelazak u sledeće stanje „unos kartice“). Ukoliko je klijent odabrao opciju 0 poziva se funkcija zavrsiProgram() koja uzima istoriju stanja kroz koja je Bankomat prošao u opsluživanju datog klijenta i, zajedno sa informacijama o bankomatu, štampa u fajl atm\_<timestamp>.out. Za timestamp se koristi klasa Date koja je formatirana u odgovarajuću formu korišćenjem funkcija klase SimpleDateFormat.

**import** java.io.FileWriter;  
**import** java.io.IOException;  
**import** java.text.SimpleDateFormat;  
**import** java.util.ArrayList;  
**import** java.util.Date;  
**import** java.util.Scanner;  
  
**public class** Pocetno {  
  
  
 **private** String **nazivBanke**;  
 **private** String **lokacija**;  
 **private** String **serijskiBrojATM**;  
  
 **public** Pocetno(String nazivBanke,String lokacija,String serijskiBrojATM) {  
 **this**.**nazivBanke**=nazivBanke;  
 **this**.**lokacija**=lokacija;  
 **this**.**serijskiBrojATM**=serijskiBrojATM;  
 }  
  
  
 **public int** printWelcome(){  
 System.***out***.println(**"\nTrenutno ste u stanju: Početno"**);  
 System.***out***.println(**"\t\tDobrodošli"**);  
 System.***out***.println(**"\tBanka: "**+**nazivBanke**);  
 System.***out***.println(**"\tLokacija: "**+**lokacija**);  
 System.***out***.println(**"\tBankomat: "**+**serijskiBrojATM**);  
 System.***out***.println(**"\n\nIzaberite šta želite uraditi:\n0 - Završi program\n1 - Učitaj kartice"**);  
 Scanner sc = **new** Scanner(System.***in***);  
 **return** sc.nextInt();  
 }  
  
 **public void** zavrsiProgram(ArrayList<String> istorija){  
 String timeStamp = **new** SimpleDateFormat(**"yyyy.MM.dd.HH.mm.ss"**).format(**new** Date());  
  
 String stanja=**"Banka: "**+**nazivBanke**+**"\nLokacija: "**+**lokacija**+**"\nBankomat: "**+**serijskiBrojATM**+**"\n\n"**;  
 **for**(String s: istorija){  
 stanja+=s+**" -> "**;  
 }  
 stanja+=**"kraj"**;  
 **try**{  
 FileWriter myWriter = **new** FileWriter(**"atm\_"**+timeStamp+**".out"**);  
  
 myWriter.write(stanja);  
 myWriter.close();  
 }**catch** (IOException e) {  
 System.***out***.println(**"An error occurred."**);  
 e.printStackTrace();  
 }  
 }  
  
  
}

* 1. **Datoteka „UnosKartice.java“**

Ova datoteka predstavlja stanje „unos kartice“. Ima svoj konstruktor kome se prosleđuje lista svih kartica sa kojima Bankomat može raditi. Po ulazu u stanje „unos kartice“ poziva se funkcija izlistajKartice() koja štampa sve kartice i traži od korisnika da odabere karticu nad kojom želi da izvršave dalje ponuđene usluge.

**import** java.util.ArrayList;  
**import** java.util.Scanner;  
  
**public class** UnosKartice {  
  
 **private** ArrayList<Kartica> **sveKartice**;  
  
 **public** UnosKartice(ArrayList<Kartica> sveKartice) {  
 **this**.**sveKartice** = sveKartice;  
 }  
  
 **public int** izlistajKartice(){  
 System.***out***.println(**"Trenutno ste u stanju: Unos Kartice"**);  
 System.***out***.println(**"Izaberite redni broj kartice:\n"**);  
 System.***out***.println(**"0 - Odustani"**);  
 **for**(**int** i=1;i<=**sveKartice**.size();i++){  
  
 System.***out***.println(i+**" - "**+**sveKartice**.get(i-1).getSer());  
 }  
 Scanner sc = **new** Scanner(System.***in***);  
 **return** sc.nextInt();  
 }  
}

* 1. **Datoteka „IzborUsluge.java“**

Ova datoteka predstavlja stanje “izbor usluge”. Ima svoj konstruktor kome se prosleđuje trenutna kartica koja je “unutar” Bankomat, odnosno kartica koju je korisnik u prethodnom stanju odabrao. Međutim rečeno je da se stanja prave prilikom stvaranja bankomat, te će se na početku vrednost trenutne kartice biti null(korisnik ne može da izabere karticu pre nego što je Bankomat kreiran). Zbog toga imamo funkciju setTrenutnaKartica() koja postavlja vrednost trenutne kartice u ovoj klasi na odgovarajuću vrednost koju je korisnik odabrao. Nad tom karticom se mogu izvršiti usluge zadate u postavci zadatka, a ponuda i izbor usluge od strane korisnika se vrši korišćenjem funkcije izlistajUsluge().

**import** java.util.Scanner;  
  
**public class** IzborUsluge {  
 **private** Kartica **trenutnaKartica**;  
 Scanner **sc** = **new** Scanner(System.***in***);  
  
 **public** IzborUsluge(Kartica trenutnaKartica) {  
  
 **this**.**trenutnaKartica** = trenutnaKartica;  
  
 }  
  
 **public int** izlistajUsluge(){  
 System.***out***.println(**"\nTrenutno ste u stanju: Izbor usluge"**);  
  
 System.***out***.println(**"\nSerijski broj kartice: "**+**trenutnaKartica**.getSer());  
 System.***out***.println(**"Prezime korisnika kartice: "**+**trenutnaKartica**.getPrezime());  
 System.***out***.println(**"Ime korisnika kartice: "**+**trenutnaKartica**.getIme());  
  
  
 System.***out***.println(**"\nUsluge koje pružamo: "**);  
 System.***out***.println(**"1 - Podizanje novca"**);  
 System.***out***.println(**"2 - Štampanje izveštaja o kartici"**);  
 System.***out***.println(**"3 - Prekid rada"**);  
 System.***out***.println(**"\nUnesite redni broj usluge:"**);  
  
 **return sc**.nextInt();  
 }  
  
 **public void** setTrenutnaKartica(Kartica trenutnaKartica) {  
 **this**.**trenutnaKartica** = trenutnaKartica;  
 }  
}

* 1. **Datoteka „PodizanjeNovca.java“**

Ova datoteka predstavlja stanje “podizanje novca”. Ima svoj konstruktor kome se prosleđuje trenutna kartica koja je “unutar” Bankomat, odnosno kartica koju je korisnik odabrao u stanju “unos kartice”. Međutim rečeno je da se stanja prave prilikom stvaranja bankomat, te će se na početku vrednost trenutne kartice biti null(korisnik ne može da izabere karticu pre nego što je Bankomat kreiran). Zbog toga imamo funkciju setTrenutnaKartica() koja postavlja vrednost trenutne kartice u ovoj klasi na odgovarajuću vrednost koju je korisnik odabrao. Ima funkciju zeljeniIznos() koja prosleđuje bankomatu sumu novca koju korisnik želi da povuče sa kartice. Takođe ima i funkciju podigniNovac() koja prethodno unetu željenu sumu pokušava da povuče sa kartice.

**import** java.util.Scanner;  
  
**public class** PodizanjeNovca {  
 **private** Kartica **trenutnaKartica**;  
 Scanner **sc** = **new** Scanner(System.***in***);  
  
 **public** PodizanjeNovca(Kartica trenuntaKartica) {  
 **this**.**trenutnaKartica** = trenuntaKartica;  
 }  
  
 **public int** zeljeniIznos(){  
 System.***out***.println(**"Trenutno ste u stanju: podizanje novca"**);  
  
 System.***out***.println(**"\nTrenutno stanje na kartici: "**+**trenutnaKartica**.getSredstva());  
 System.***out***.println(**"Koliko novca želite podići: "**);  
 **return sc**.nextInt();  
 }  
  
 **public boolean** podigniNovac(**int** novac){  
 **return trenutnaKartica**.podizanjeSredstva(novac);  
 }  
  
 **public void** setTrenutnaKartica(Kartica trenutnaKartica) {  
 **this**.**trenutnaKartica** = trenutnaKartica;  
 }  
}

* 1. **Datoteka „StampanjeIzvestaja.java“**

Ova datoteka predstavlja stanje “stampanje izvestaja”. Ima svoj konstruktor kome se prosleđuje trenutna kartica koja je “unutar” Bankomat, odnosno kartica koju je korisnik odabrao u stanju “unos kartice”. Međutim rečeno je da se stanja prave prilikom stvaranja bankomat, te će se na početku vrednost trenutne kartice biti null(korisnik ne može da izabere karticu pre nego što je Bankomat kreiran). Zbog toga imamo funkciju setTrenutnaKartica() koja postavlja vrednost trenutne kartice u ovoj klasi na odgovarajuću vrednost koju je korisnik odabrao. Ima funkciju stampajIzvestaj() koja štampa odgovarajuće informacije tražene u postavci zadatka u fajl report\_<timestamp>.out.

**import** java.io.FileWriter;  
**import** java.io.IOException;  
**import** java.text.SimpleDateFormat;  
**import** java.util.Date;  
**import** java.util.Scanner;  
  
**public class** StampanjeIzvestaja {  
  
 **private** Kartica **trenutnaKartica**;  
 Scanner **sc** = **new** Scanner(System.***in***);  
  
 **public** StampanjeIzvestaja(Kartica trenutnaKartica) {  
 **this**.**trenutnaKartica** = trenutnaKartica;  
 }  
  
 **public boolean** stampajIzvestaj() {  
 System.***out***.println(**"Trenutno ste u stanju: Štampanje Izveštaja"**);  
  
 System.***out***.println(**"Štampanje u toku..."**);  
  
 String timeStamp = **new** SimpleDateFormat(**"yyyy.MM.dd.HH.mm.ss"**).format(**new** Date());  
  
 String stanja = **"Ime: "** + **trenutnaKartica**.getIme() + **"\nPrezime: "** + **trenutnaKartica**.getPrezime() + **"\nSerijski broj kartice: "** + **trenutnaKartica**.getSer() + **"\nDatum i vreme stampanja: "** + timeStamp + **"\nRaspoloživa sredstva: "** + **trenutnaKartica**.getSredstva() + **"\n\n"**;  
 **try** {  
 FileWriter myWriter = **new** FileWriter(**"report"** + timeStamp + **".out"**);  
  
 myWriter.write(stanja);  
 myWriter.close();  
 } **catch** (IOException e) {  
 System.***out***.println(**"An error occurred."**);  
 e.printStackTrace();  
 }  
  
 **return true**;  
 }  
  
 **public void** setTrenutnaKartica(Kartica trenutnaKartica) {  
 **this**.**trenutnaKartica** = trenutnaKartica;  
 }  
}

**2.8 Datoteka „ATM.java“**

Ova datoteka predstavlja srž ovog programa. U ovoj datoteci se nalazi klasa ATM koja predstavlja Bankomat i u kojoj se menjaju odgovarajuća stanja. Ima svoj konstruktor kojim se postavlja serijski broj bankomat, njegova lokacija i naziv banke kojoj bankomat pripada. Takođe incijalizujemo i liste svih kartica sa kojima Bankomat može raditi i lista istorije stanja kroz koje prolazi Bankomat prilikom jednog opsluživanja korisnika. U Konstrukturu stvaramo i, gorepomenuta, stanja u kojima se može Bankomat naći.

Imamo funkciju dodajKarticu() koja dodaje novokreiranu platnu karticu na listu kartica unutar Bankomata.

Funkcija pocetno() postavlja stanje Bankomata u početno i vrši štampu poruke dobrodošlice i na osnovu korisničkog izbora može prekinuti rad programa ili preći u stanje „unos kartice“. Ako je izabrana nevalidna opcija korisnik će opet „preći“ u isto stanje u kome se nalazi, dakle početno.

Takođe novopostavljeno stanje se dodaje u listu istorije stanja.

Funkcija izlistajKartice() postavlja stanje Bankomata u stanje „unos kartice“ i vrši štampu svih kartica sa kojima Bankomat može raditi i na osnovu korisničkog izbora može se vratiti u početno stanje ili postaviti korisnički izabranu karticu kao karticu koja se nalazi „unutar“ Bankomata i preći u stanje „izbor usluge“. Ako je izabrana nevalidna opcija korisnik će opet „preći“ u isto stanje u kome se nalazi, dakle „unos kartice“. Takođe novopostavljeno stanje se dodaje u listu istorije stanja.

Funkcija izlistajUsluge() postavlja stanje Bankomata u stanje „izbor usluge“ i vrši štampu svih usluga koje Bankomat pruža i na osnovu korisničkog izbora može preći u stanje „štampanje izveštaja“ ili u stanje „podizanje novca“ ili izvršiti prekid rada Bankomata. Ako je izabrana nevalidna opcija korisnik će opet „preći“ u isto stanje u kome se nalazi, dakle „izbor usluge“. Takođe novopostavljeno stanje se dodaje u listu istorije stanja.

Funkcija stampajIzvestaj() postavlja stanje Bankomata u stanje „štampanje izveštaja“ i vrši stvaranje fajla report\_<timestamp>.out i upis odgovarajućih informacija u njega. Za vreme kreiranja fajla korisniku se pokazuje poruka „Štampanje u toku...“. Po završetku pravljenja fajla Bankomat se postavlja u stanje „izbor usluge“. Dodat je i lag od 1 sekunde kako bi korisnik mogao da pročita poruku „Štampanje u toku...“ pre nego što bude vraćen u stanje „izbor usluge“. Takođe novopostavljeno stanje se dodaje u listu istorije stanja.

Funkcija podigniNovac() postavlja stanje Bankomata u stanje „podizanje novca“ i traži željenu sumu od korisnika. Pošto korisnik unese željenu sumu vrši se podizanje novca sa prethodno izabrane kartice. Ukoliko je željena suma veća od dostupne svote novca na kartici, korisnik se obaveštava o grešci i traži mu se da unese novu željenu sumu. Po uspešnom podizanju novca Bankomat prekida rad i vrši štampanje atm\_<timestamp>.out fajla.Takođe novopostavljeno stanje se dodaje u listu istorije stanja.

Funkcija getSveKartice() vraća sve kartice sa kojim Bankomat može da operiše. Koristi se prilikom kreiranja nove Platne kartice kako se ne bi desilo da se serijski broj nove kartice poklopi sa karticom koja već postoji.

**import** java.util.ArrayList;  
  
**public class** ATM{  
  
 **private** String **nazivBanke**;  
 **private** String **lokacija**;  
 **private** String **serijskiBrojAtm**;  
 *//lista svih kartica sa kojima bankomat moze raditi* **private** ArrayList<Kartica> **sveKartice**;  
 *//kartica koja je "u bankomatu" zato sto Bankomat moze raditi samo sa jednom karticom u jednom trenutku* **private** Kartica **trenutnaKartica**;  
  
 *//zadatkom navedena stanja* **private** Pocetno **pocetno**;  
 **private** UnosKartice **unosKartice**;  
 **private** IzborUsluge **izborUsluge**;  
 **private** StampanjeIzvestaja **stampanjeIzvestaja**;  
 **private** PodizanjeNovca **podizanjeNovca**;  
  
 *//cuva kroz koja stanja je prosao bankomat tokom jednog koriscenja od strane nekog korisnika* ArrayList<String> **istorijaStanja**;  
  
 **public** ATM(String nazivBanke, String lokacija,String serijskiBrojAtm) {  
 **this**.**nazivBanke** = nazivBanke;  
 **this**.**lokacija** = lokacija;  
 **this**.**serijskiBrojAtm** = serijskiBrojAtm;  
  
 **sveKartice** = **new** ArrayList<Kartica>();  
 **istorijaStanja**= **new** ArrayList<String>();  
  
 *//inicijalizacija svih stanja u kojima se moze naci bankomat, a data su postavkom zadatka* **pocetno**=**new** Pocetno(nazivBanke,lokacija,serijskiBrojAtm);  
 **unosKartice** = **new** UnosKartice(**sveKartice**);  
 **this**.**izborUsluge** = **new** IzborUsluge(**trenutnaKartica**);  
 **this**.**stampanjeIzvestaja** = **new** StampanjeIzvestaja(**trenutnaKartica**);  
 **this**.**podizanjeNovca** = **new** PodizanjeNovca(**trenutnaKartica**);  
  
  
 }  
  
 **public void** dodajKarticu(Kartica k){  
 **sveKartice**.add(k);  
 }  
  
 **public void** pocetno(){  
 **istorijaStanja**.add(**"početno"**);  
 **int** i=**pocetno**.printWelcome();  
  
 **if**(i==0){  
 **pocetno**.zavrsiProgram(**istorijaStanja**);  
 }**else if**(i==1){  
  
 **this**.izlistajKartice();  
  
 }**else**{  
 System.***out***.println(**"\nNevalidna opcija, molimo unesite validnu opciju(0 ili 1): "**);  
 **this**.pocetno();  
 }  
  
  
 }  
  
 **private void** izlistajKartice(){  
 **istorijaStanja**.add(**"unos kartice"**);  
 **int** i=**unosKartice**.izlistajKartice();  
 **if**(i==0){  
 **this**.pocetno();  
 }**else if**(i>=0&&i<=**sveKartice**.size()){  
 **trenutnaKartica**=**sveKartice**.get(i-1);  
  
 **this**.izlistajUsluge();  
  
  
 }**else**{  
 System.***out***.println(**"Pogrešan redni broj kartice! Pokušajte ponovo. "**);  
 **this**.izlistajKartice();  
 }  
 }  
  
 **private void** izlistajUsluge(){  
 **izborUsluge**.setTrenutnaKartica(**trenutnaKartica**);  
 **istorijaStanja**.add(**"izbor usluge"**);  
 **int** i = **izborUsluge**.izlistajUsluge();  
 **if**(i==1){  
 **this**.podigniNovac();  
  
 }**else if**(i==2){  
  
 **this**.stampajIzvestaj();  
 }**else if**(i==3){  
 **pocetno**.zavrsiProgram(**istorijaStanja**);  
  
 }**else**{  
 System.***out***.println(**"Pogrešan redni broj usluge! Pokušajte ponovo. "**);  
 **this**.izlistajUsluge();  
 }  
  
 }  
  
 **private void** stampajIzvestaj(){  
 **stampanjeIzvestaja**.setTrenutnaKartica(**trenutnaKartica**);  
  
 **istorijaStanja**.add(**"štampanje izveštaja"**);  
 **boolean** isPrinted=**stampanjeIzvestaja**.stampajIzvestaj();  
 **if**(isPrinted){  
 **try** {  
 Thread.*sleep*(1000);  
 }  
 **catch**(InterruptedException ex)  
 {  
 Thread.*currentThread*().interrupt();  
 }  
  
 **this**.izlistajUsluge();}  
 }  
  
 **private void** podigniNovac(){  
 **podizanjeNovca**.setTrenutnaKartica(**trenutnaKartica**);  
  
 **istorijaStanja**.add(**"podizanje novca"**);  
 **int** novac=**podizanjeNovca**.zeljeniIznos();  
 **boolean** isValid=**podizanjeNovca**.podigniNovac(novac);  
 **while**(!isValid){  
 novac=**podizanjeNovca**.zeljeniIznos();  
 isValid=**podizanjeNovca**.podigniNovac(novac);  
 }  
  
 *//cisto stampanje stanja kroz koja smo prosli, nije navedeno u zadatku* **pocetno**.zavrsiProgram(**istorijaStanja**);  
  
 }  
  
 **public** ArrayList<Kartica> getSveKartice() {  
 **return sveKartice**;  
 }  
}

* 1. **Datoteka „main.java“**

Ovo je datoteka u kojoj se dodaju nove platne kartice i u kojoj se kreira Bankomat. Po stvaranju Bankomata i dodavanju par kartica kao primer možemo pokrenuti bankomat funkcijom pocetno().

**public class** main {  
 **public static void** main(String[] args) {  
 ATM bankomat= **new** ATM(**"Komercijalna"**, **"Kralja Aleksandra 1"**,**"20394"**);  
 Kartica korisnik1 = **new** Kartica(**"Petar"**, **"Petrović"**,17000,bankomat.getSveKartice());  
 Kartica korisnik2 = **new** Kartica(**"Marko"**, **"Marković"**,50000,bankomat.getSveKartice());  
 bankomat.dodajKarticu(korisnik1);  
 bankomat.dodajKarticu(korisnik2);  
  
 bankomat.pocetno();  
  
  
  
 }  
  
}

1. **UML dijagrami**

The Unified Modeling Language ili skraćeno UML je standardni grafički jezik za

modelovanje objektno-orjentisanog softvera.

Zbog toga što je UML bogat jezik koji obuhvata velik broj dijagrama biće prikazani samo

dijagrami koji se najčešće koriste u praksi, i koji su obrađeni na vežbama, a to su:

- dijagram slučajeva korišćenja;

- dijagram klasa;

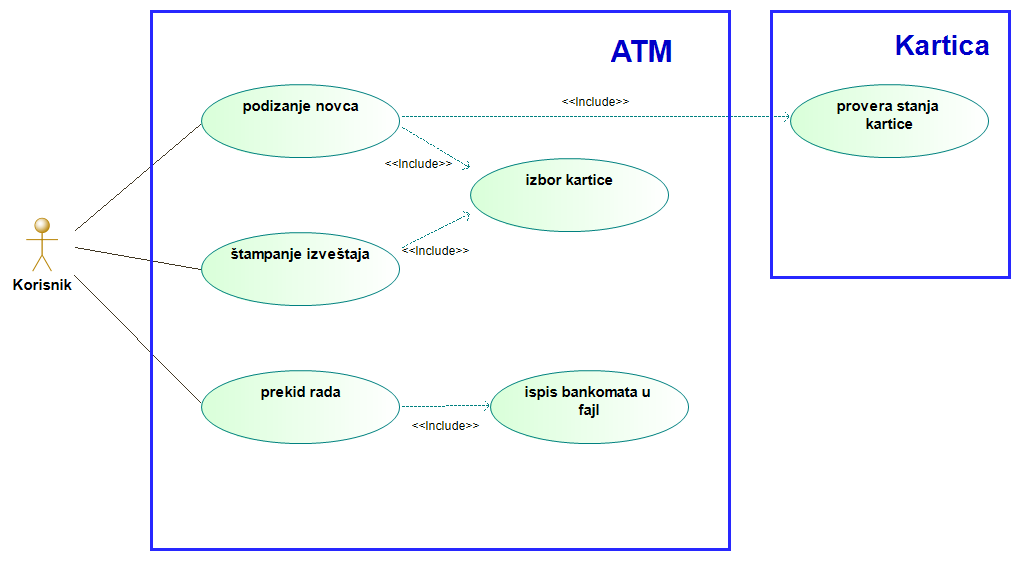
- dijagram sekvenci;

- dijagram aktivnosti;

- dijagram stanja.

* 1. **Dijagram slučajeva korišćenja**

Dijagram slučajeva korišćenja (use-case diagram) prikazuje skup slučajeva korišćenja i aktera. Tipično se koristi da specificira neku funkcionalnost i ponašanje nekog sistema. Slučaj korišćenja specificira šta subjekt radi, a ne kako radi.



Slika 1 Dijagram slučajeva korišćenja

* 1. **Dijagram klasa**

Dijagram klasa (class diagram) je vrsta strukturnog dijagrama u softverskom

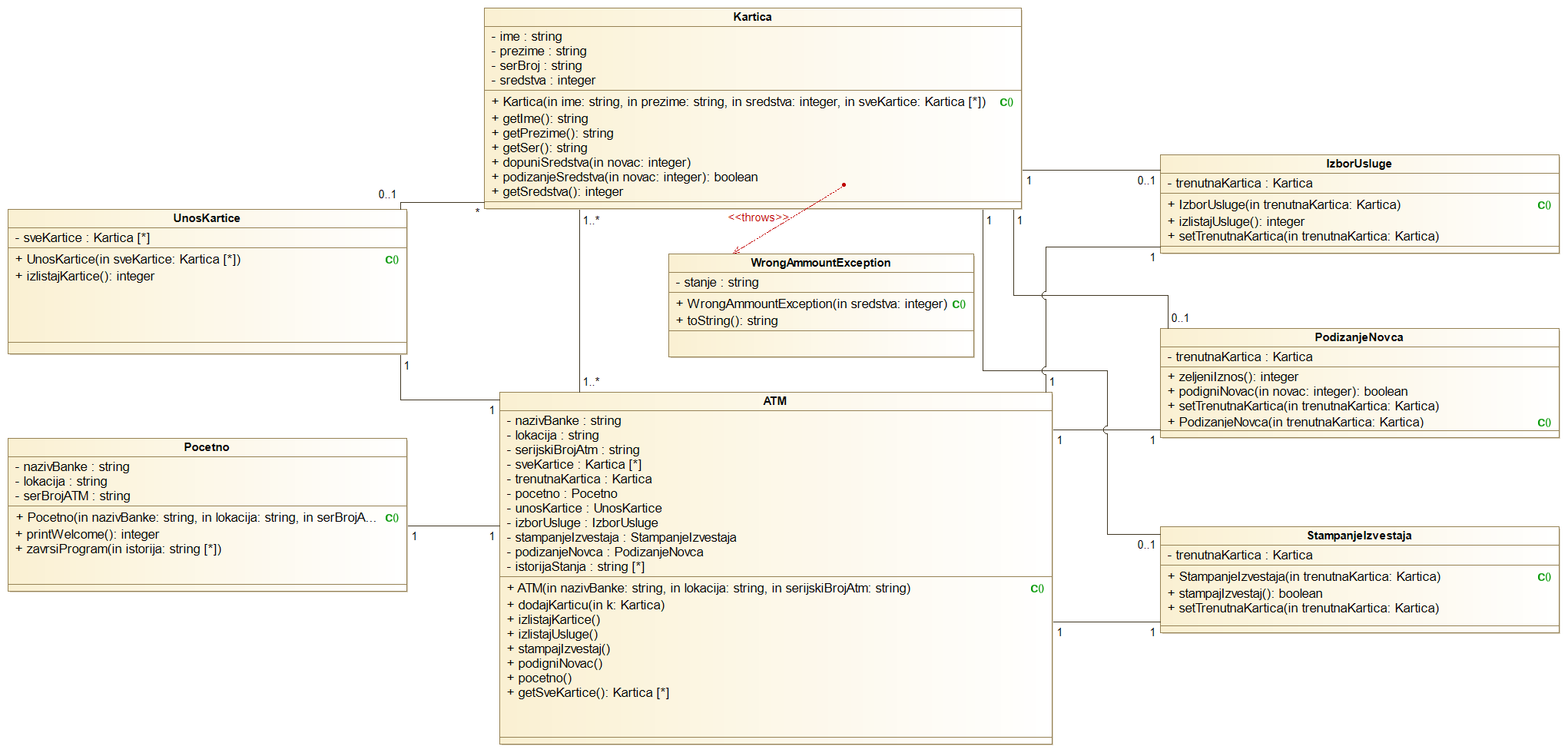
inžinjeringu, koji opisuje strukturu sastava objašnjavajući klase unutar sastava, njihove

atribute i odnose. Dijagrami klasa su najčešći u objektnom modeliranju

Elementi dijagrama klasa su:

- stvari: klasa, interfejsi, tipovi, izuzeci, šabloni, saradnje, paketi

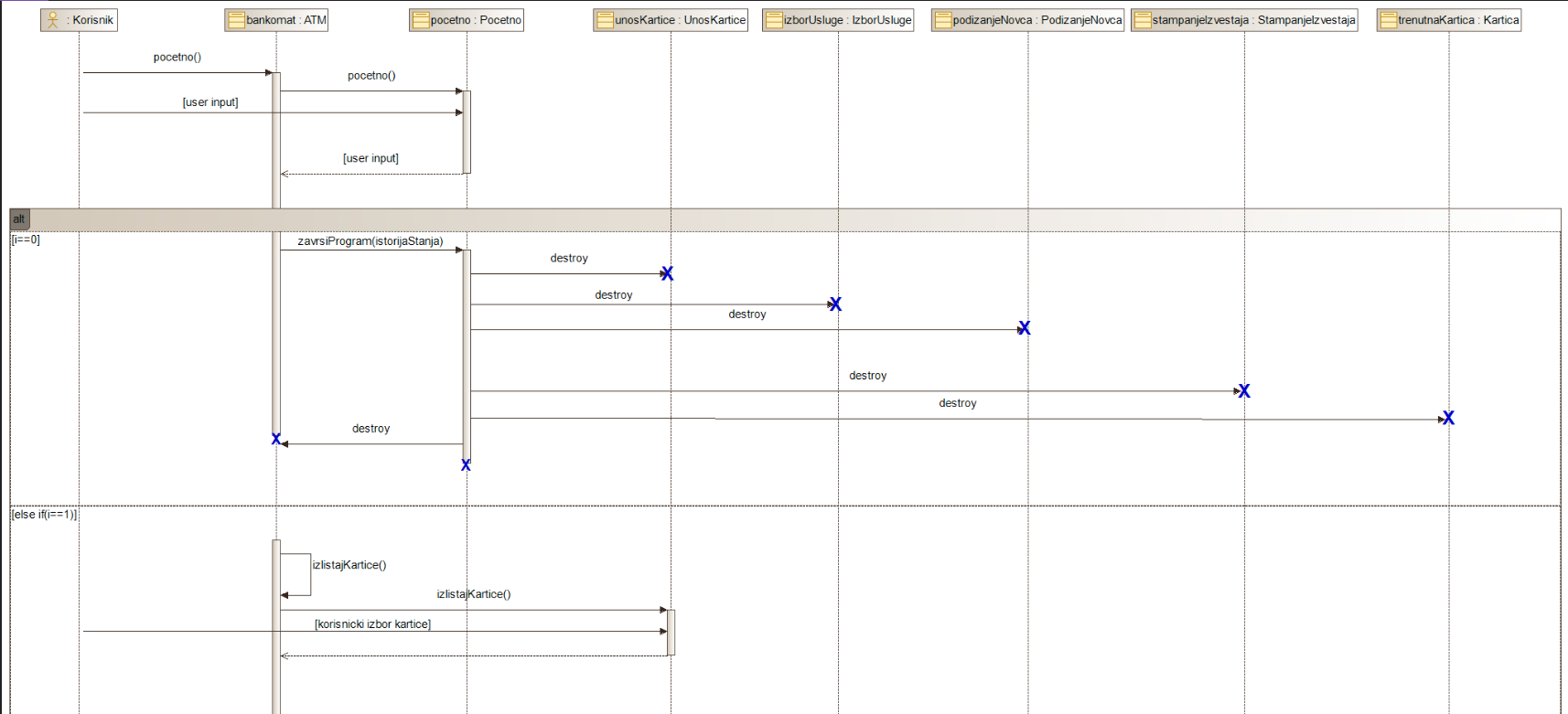
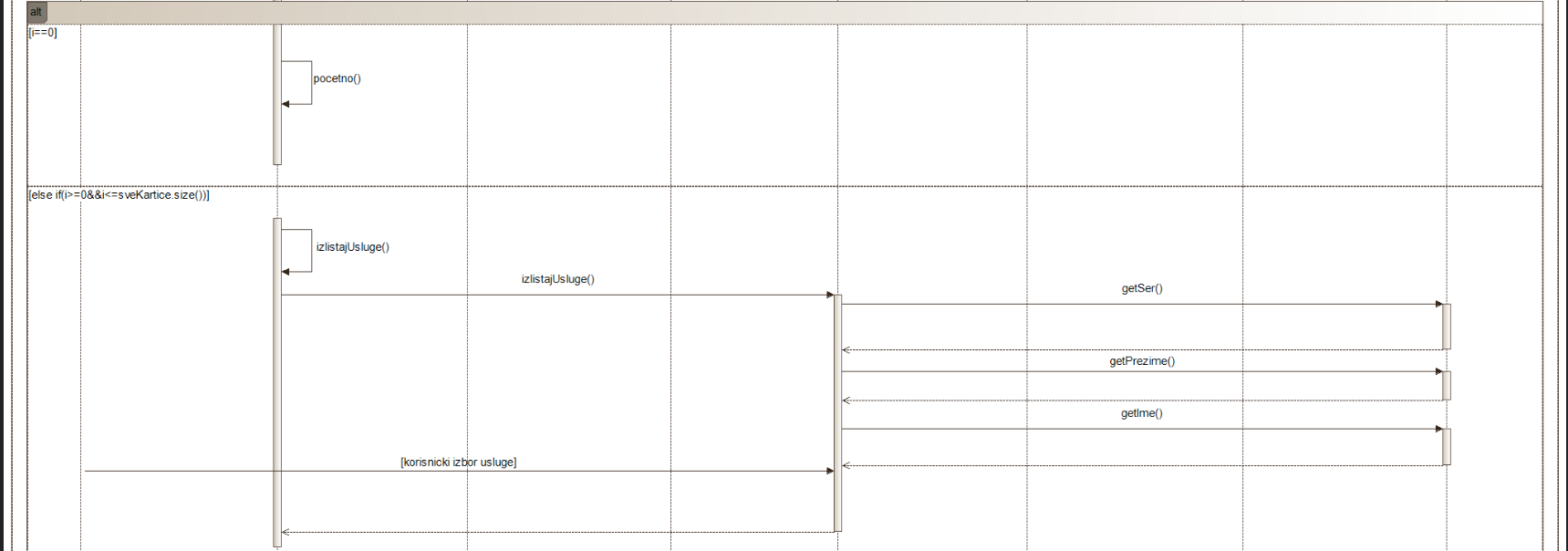
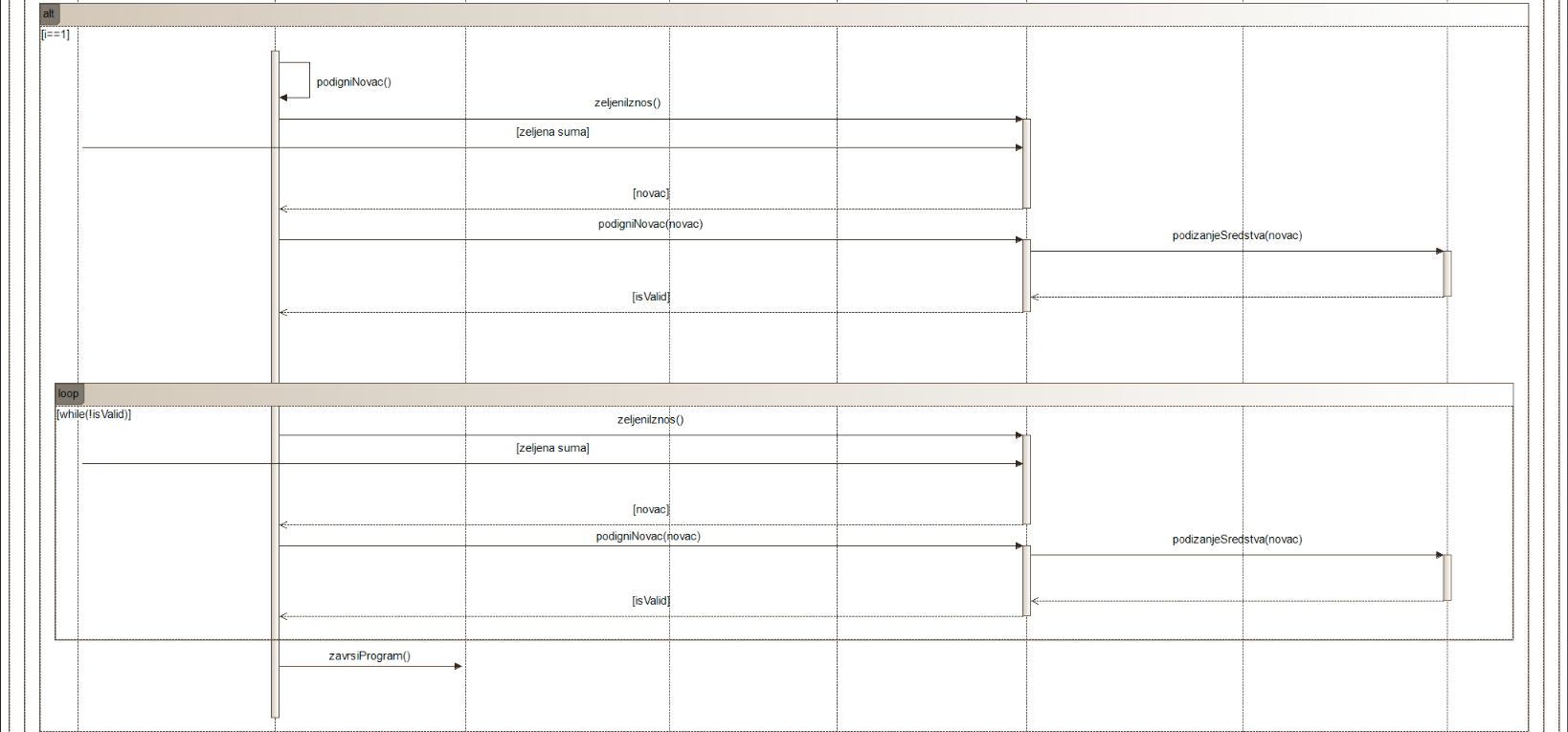
- relacije: zavisnosti, generalizacije, asocijacije, realizacije

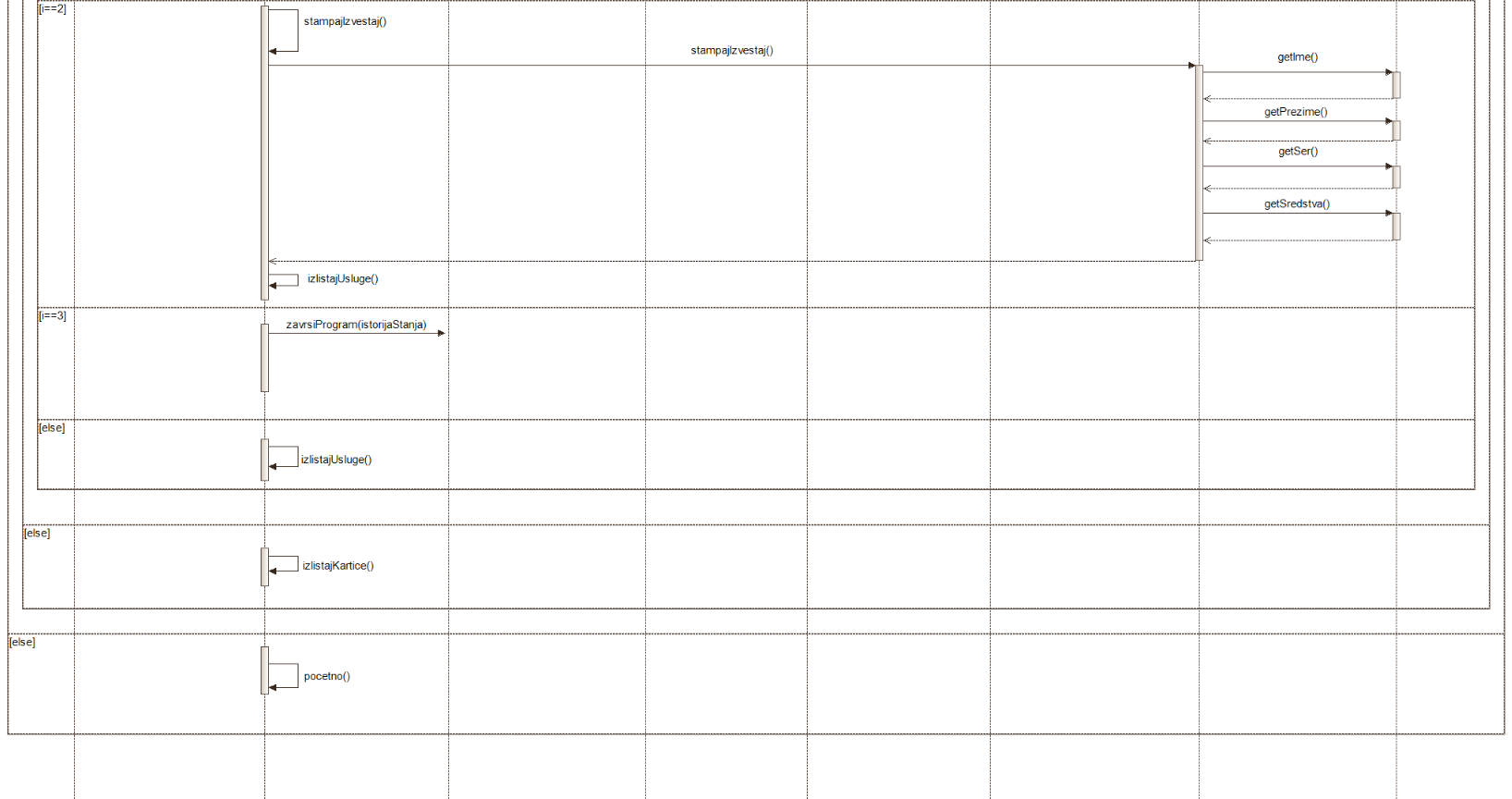


Slika 2 Dijagram klasa

* 1. **Dijagram sekvenci**

Dijagram sekvenci je jedan od četiri tipa dijagrama interakcije. Dijagram sekvenci prikazuje komunikaciju između skupa objekata, koja se ostvaruje porukama koje objekti međusobno razmenjuju u cilju ostvarivanja očekivanog ponašanja. Detaljno opisuje kako se operacije izvode i koje poruke se šalju i kada. Dijagrami sekvenci mogu da sadrže aktere, objekte i poruke. Vertikalna osa prikazuje interakciju, a horizontalna osa hronološki red izvršavanja.

 Slika 3 Dijagram sekvenci

* 1. **Dijagram aktivnosti**

Dijagrami aktivnosti(activity diagram) su namenjeni modeliranju dinamičkih aspekata (ponašanja) sistema.

Dijagram aktivnosti prikazuje:

- tok aktivnosti koju izvršavaju objekti;

- eventualno i tok objekata između koraka aktivnosti.

Aktivnost je specifikacija parametrizovanog ponašanja koje se izražava kroz tok izvršenja

preko sekvenciranja i konkurisanja podaktivnosti. Dijagram aktivnosti je graf koji sadrži

čvorove i grane.

Grane:

- prelazi (tranzicije) između akcija;

- tok objektač.

Čvorovi:

- akcije i aktivnosti;

- objekti;

- slanje signala (send signal) ;

- prihvatanja događaja (accept event) ;

- prihvatanja vremenskog događaja (accept time event) ;

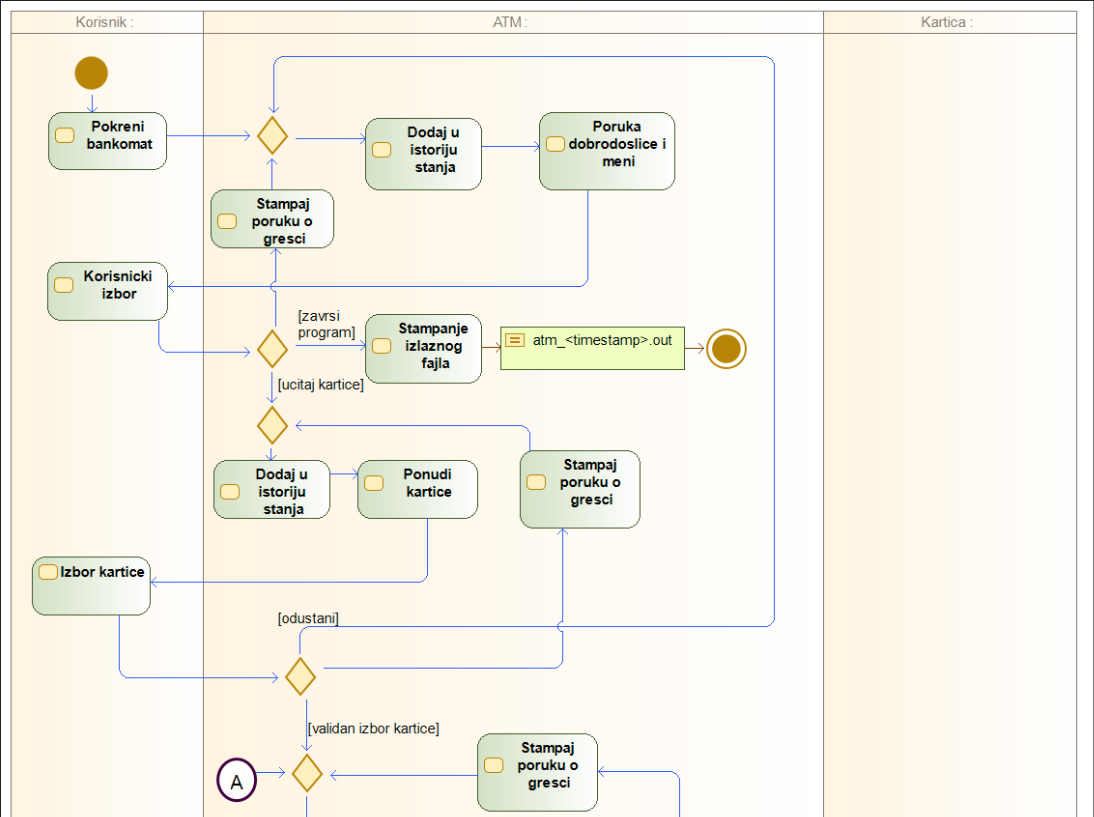
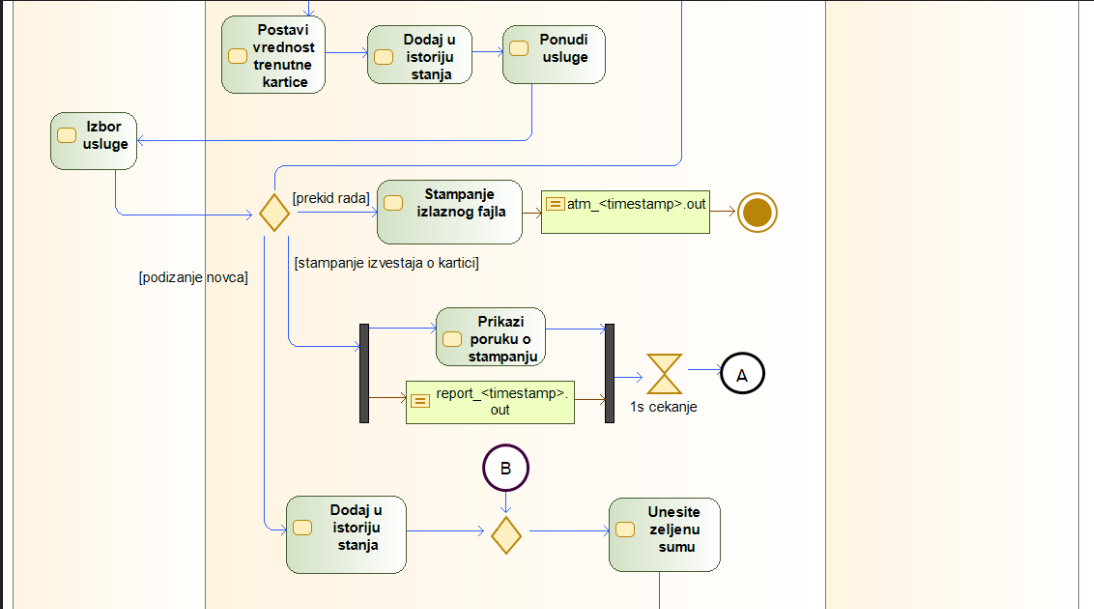
- kontrolni čvorovi:

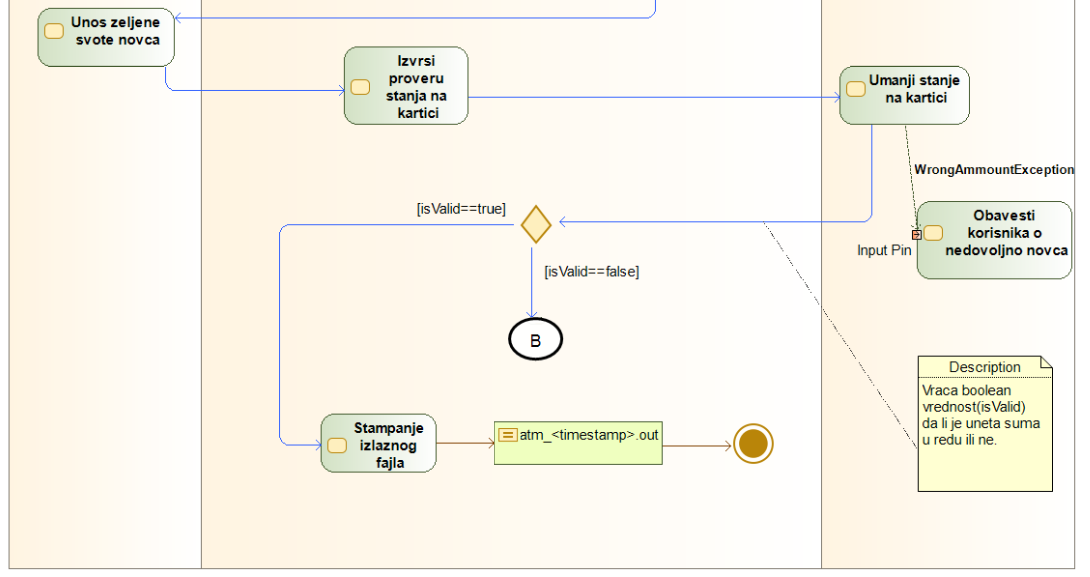
- sekvencijalna grananja i spajanja u toku kontrole (decision i merge) ;

- konkurentna grananja i spajanja u toku kontrole (fork i join) ;

- pseudočvorovi: početni, završni i kraj toka ;

- konektori.

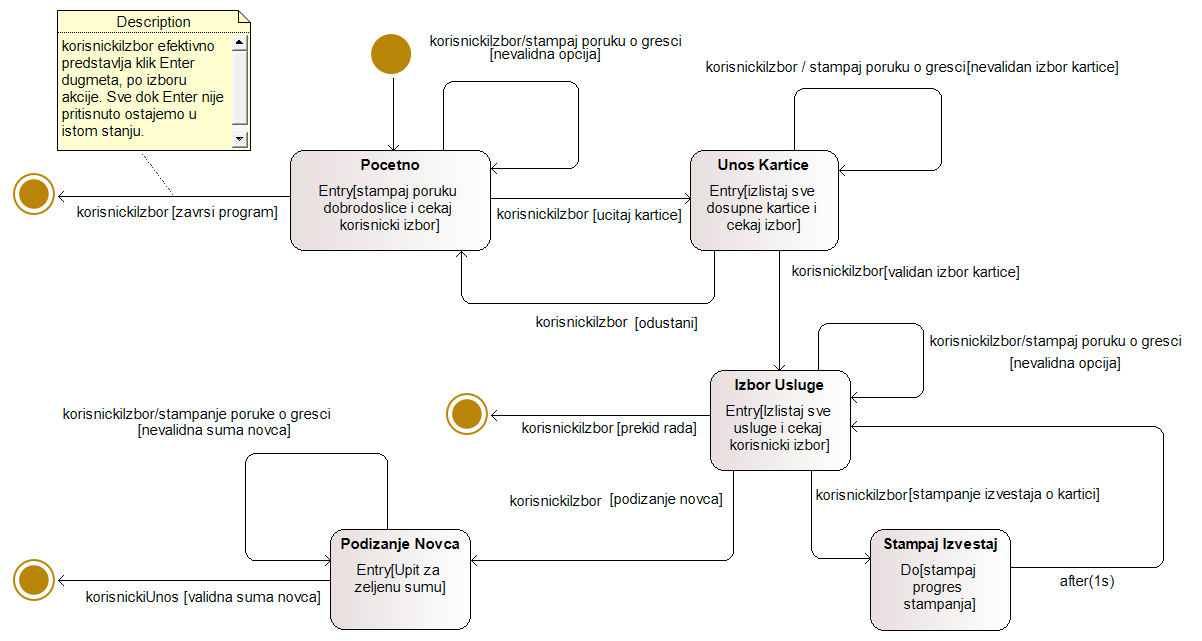
 



Slika 4 Dijagram Aktivnosti

* 1. **Dijagram stanja**

Dijagrami stanja (state machine diagram) se koriste za opisivanje ponašanja sistema. Dijagram stanja je graf koji prikazuje automat stanja: čvorovi su stanja, a grane su prelazi. Dijagrami stanja se fokusiraju na tok aktivnosti. Dijagrami stanja se fokusiraju na događajima vođeno ponašanje. Dijagrami stanja se kreiraju za entitete koji pokazuju bitno dinamičko ponašanje.



Slika 5 Dijagram stanja

1. **Literatura**

[1] E – učenje, Računarska tehnika i softversko inženjerstvo, VI semestar, Softverski inženjering

<http://moodle.fink.rs/course/view.php?id=978>

[2] Finite-state machine, <https://en.wikipedia.org/wiki/Finite-state_machine>

[3] State design pattern, <https://www.geeksforgeeks.org/state-design-pattern/>

[4] How to show Labels in State machine, <https://www.modelio.org/forum/16-feature-requests/4441-labels-in-a-state-machine.html>