VISOKA TEHNIČKA ŠKOLA
STRUKOVNIH STUDIJA NIŠ

SOFTVERSKO INŽENJERSTVO

LABORATORIJSKE VEŽBE I DEO

2019

AUTORI:
Brankica Stefanović RER 65/16
Filip Tonić RER 71/16

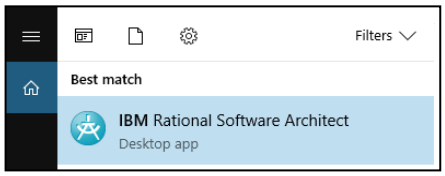
PREDMETNI PROFESOR:
dr Miloš Stojanović

VEŽBA 1

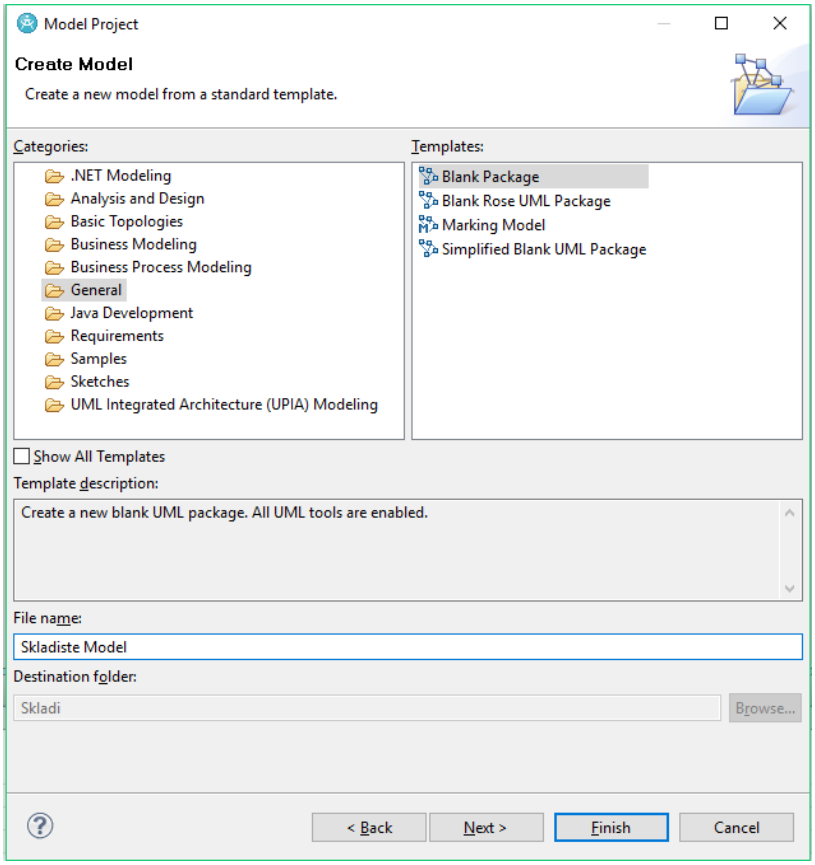
Zadatak 1: Kreirati dijagram klasa pomoću UMl-a za sledeći sistem klasa

- **Predmet** ima jedinstven, automatski generisan celobrojni identifikator kome se može pristupiti.
- **Skladište** može da sadrži zadati broj predmeta. Kreira se prazno, nakon čega se predmeti mogu dodavati i iz skladišta uzimati. Predmeti se uzimaju po redosledu stavljanja. Može se videti broj predmeta u skladištu, kao i da li je skladište puno ili prazno.
- **Proizvođač** može da napravi jedan predmet i da ga stavi u skladište koje se zadaje prilikom kreiranja proizvođača (instance klase proizvođač).
- **Potrošač** može da uzme jedan predmet iz skladišta koje se zadaje prilikom kreiranja novog potrošača.

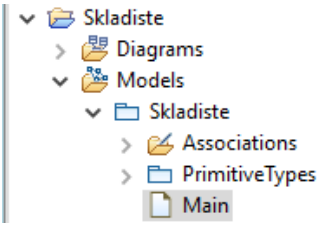
1. Pokrenite RSA



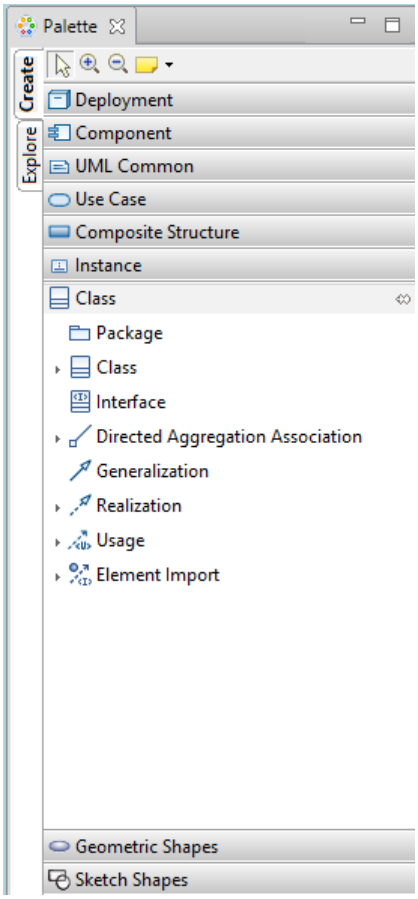
2. Iz glavnog menija izaberite **File > New Model Project**
3. Dodelite ime projektu (npr. Skladiste) i dodajte prazan projekat.
4. Klikom na Finish, kreirali ste UML projekat.



5. Otvoriti Main fajl iz project Explorera.

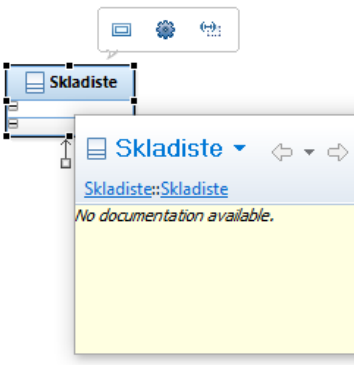


6. Iz Palette menija izabrati Class polje, i prevući u Main fajl.



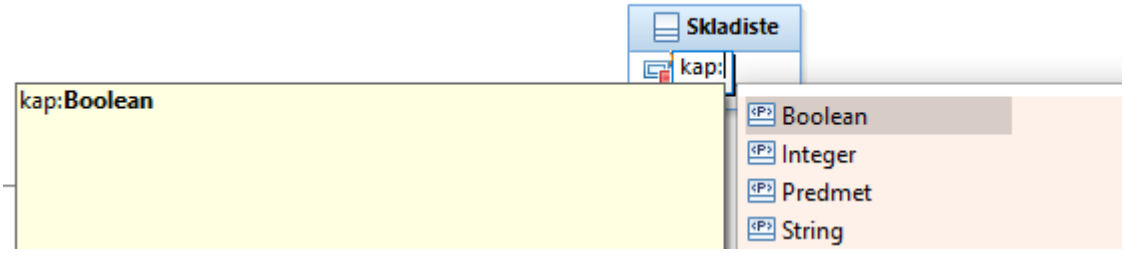
7. Dodelite ime klasi i pritiskom na Enter kreiraće se klasa sa zadatim imenom.

8. Pozicioniranjem kursora na kreirani elment, vidimo njegove informacije kao i podmeni za dodavanje atributa i funkcija datoj klasi.

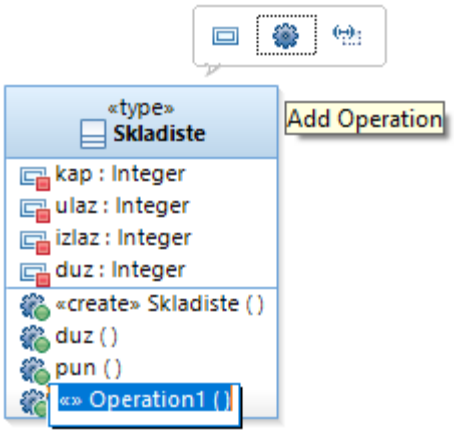


9. Selektovanjem prvog elementa u podmeniju dodajemo atribut klasi.

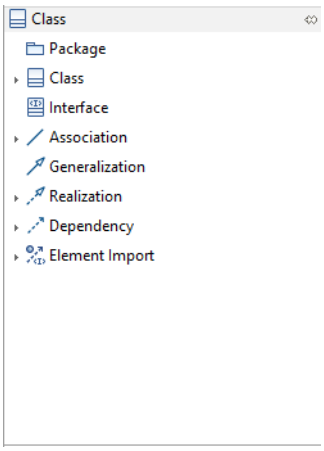
10. Dodelite ime atributu i izaberite jedan od ponuđenih ugrađenih tipova podataka.



11. Selektovanjem drugog elementa u podmeniju dodajemo metod klasi.



12. Kreiramo metod sa ili bez atributa, kako je traženo zadatkom.
13. Postupak za kreiranje klasa, kao i njihovih atributa i funkcija ponoviti za ostale klase iz zadatka.
14. Da bi smo povezali klase, potrebno je da izaberemo tip veze, kao i vrstu particije pojedinih klasa u dijagramu.
15. Iz **Class** menija komande palete biramo tip veze i prevlačenjem sa jedne klase na drugu kreiramo vezu.



Odnosi između klasa (tipovi veza):

- Asocijacija
- Agregacija
- Kompozicija

Asocijacija je veza između dve klase i prikazana je **punom linijom**. Svaki kraj linije može da ima **naziv uloge** i **multiplikativnost.**
U UML-u, asocijacija predstavlja klasu koja je deo veze između druge dve klase. Klasa koju prestavlja asocijacija možemo definisati tako da pruži dodatne informacije o vezi u kojoj posreduje i može sadržati operacije, atribute kao i druge asocijacije.

Multiplikativnost se može predstaviti na sledeće načine:

- 0.. 1 nula ili jedan
- 1 tačno jedan
- * nula ili više
- 1.. * jedan ili više

U UML-u multiplikacija predstavlja vrednost na svakoj od strana jedne veze, i predstavlja broj objekta klase koja može participirati u vezi. Takože se može specificirati i multiplikacija željenog atributa i tada zadata vrednost predstavlja broj vrednosti koje možemo da dodelimo atributu .

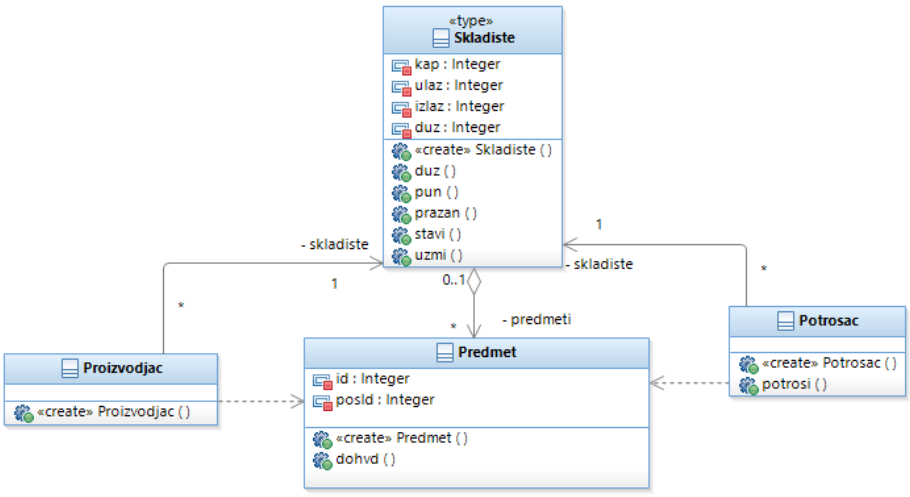
Agregacija je odnos gde je jedna klasa deo druge klase. U osnovnoj agregaciji klasa koja predstavlja deo druge klase, može postojati i bez klase koju dopunjava. Predstavlja se **simbolom romba**.

Motor je deo automobila, ali i bez automobila motor može postojati kao celina.

Kompozicija je tip agregacije, gde klasa koja predstavlja celinu, sadrži **instancu klase** koja je njen deo. U ovom slučaju instanca klase **ne može postojati** bez klase koju dopunjuje. Kompozicija se predstavlja **popunjenim simbolom romba** uz klasu koja čini celinu.

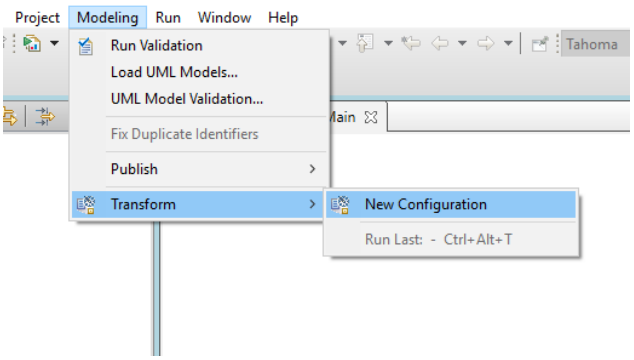
Kuća mora sadržati barem jednu sobu, a soba može postajati samo kao deo kuće.

16. Kreirati dijagram kao na slici

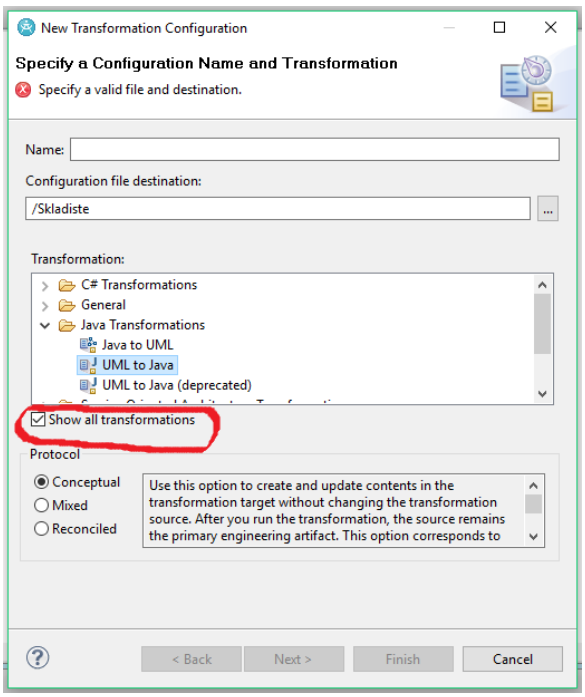


17. Nakon kreiranja dijagrama, potrebno je transformisati ga u Java, C# i C++ programski kod.

18. Izaberite **Modeling > Transform > New Configuration**

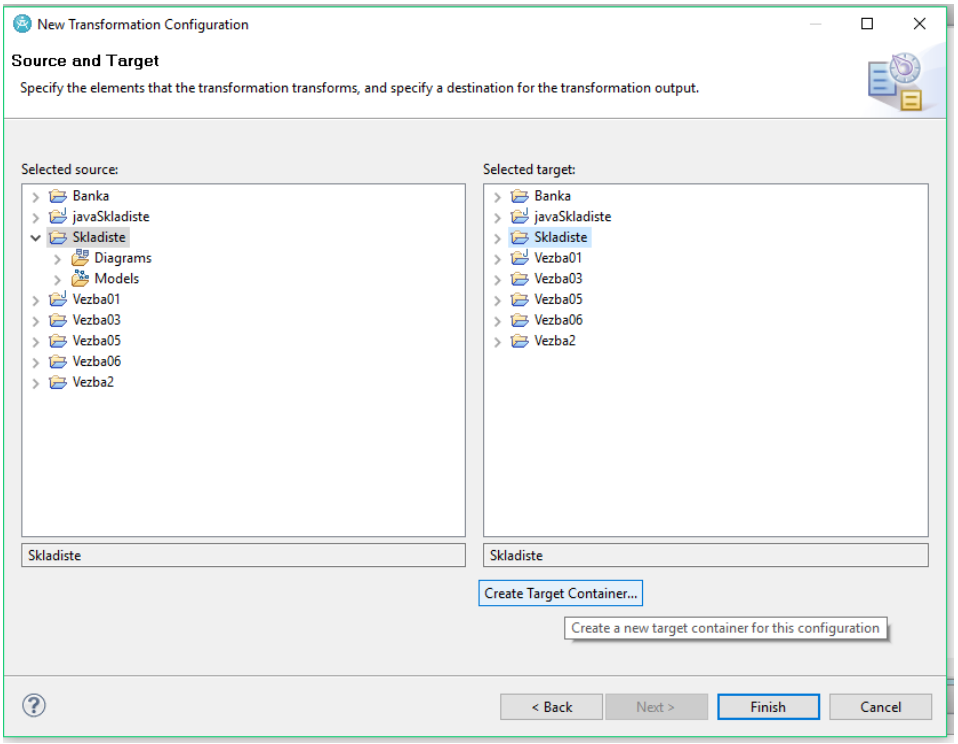


19. U sledećem prozoru čekirati opciju **Show all transformations** a zatim, iz menija Java Transformations izabrati **UML to java** . Zadajte ime koje će predstavljati ime Java projekta i kliknite Next.

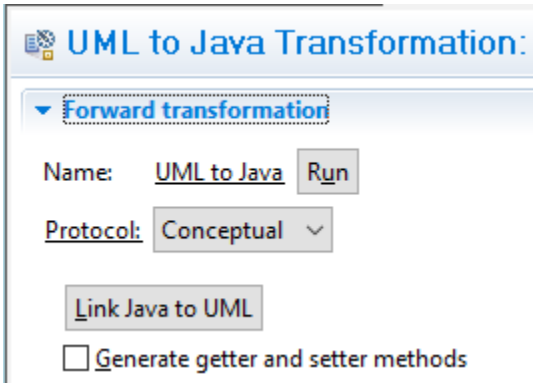


20. Dobićete listu projekta sa leve strane, gde treba odabrati projekat u kome se nalazi željeni UML dijagram, kao i listu projekata u kojima se dobijeni Java projekat može naći, sa desne strane.

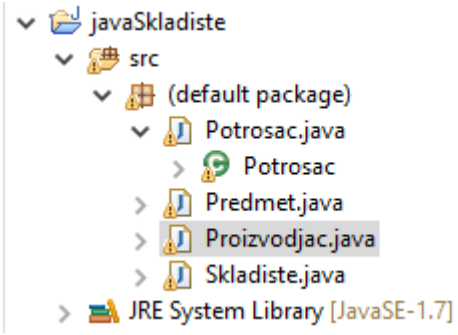
21. Kreiranje transformacije završite klikom na **Finish**.



22. Nakon što smo kreirali transformaciju, dobićemo sledeći kontekstni meni.



23. Kliknite **Run** u **Project Explorer-u** - pjaviće se novi projekat koji će sadržati java fajlove.



Otvaranjem svake klase ponaosob, možemo videti dobijeni programski kod.

Potrosac.java | Proizvodjac.java

```
/**

/**
 * <!-- begin-UML-doc -->
 * <!-- end-UML-doc -->
 * @author tonic
 * @generated "UML to Java (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
 */
public class Proizvodjac {
    /**
     * <!-- begin-UML-doc -->
     * <!-- end-UML-doc -->
     * @generated "UML to Java (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    private Skladiste skladiste;

    /**
     * <!-- begin-UML-doc -->
     * <!-- end-UML-doc -->
     * @param s
     * @generated "UML to Java (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public Proizvodjac(Skladiste s) {
        // begin-user-code
        // TODO Auto-generated constructor stub
        // end-user-code
    }
```

```
/**/

/**
 * <!-- begin-UML-doc -->
 * <!-- end-UML-doc -->
 * @author tonic
 * @generated "UML to Java (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
 */
p/**/
/**
 * <!-- begin-UML-doc -->
 * <!-- end-UML-doc -->
 * @author tonic
 * @generated "UML to Java (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
 */
public class Proizvodjac {
        /**
         * <!-- begin-UML-doc -->
         * <!-- end-UML-doc -->
         * @params x,y
         * @generated "UML to Java (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
         */
        public Skladiste skladiste {

        }
        /**
         * <!-- begin-UML-doc -->
         * <!-- end-UML-doc -->
         * @param s
         * @generated "UML to Java (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
         */
```

```
	public Proizvodjac(Skladiste s) {
		// begin-user-code
		// TODO Auto-generated constructor stub
		// end-user-code
	}

	public void proizvedi() {
	}
}
```

24. Postupak kreiranja programskog koda za ostale programske jezike, ponoviti kao za dati primer Java programskog jezika.

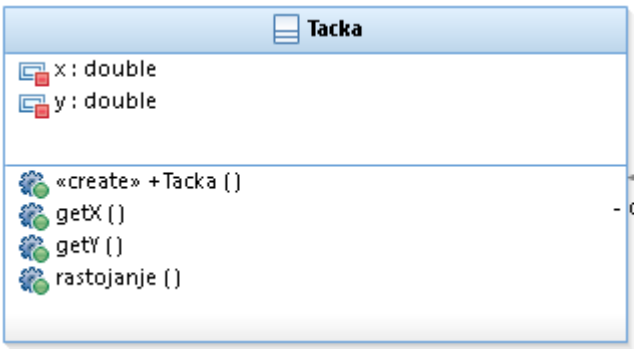
VEŽBA 2

Zadatak 1: Kreirati dijagram klasa koristeći UML, za sledeći sistem klasa (detaljan postupak je prikazan u vežbi 1)

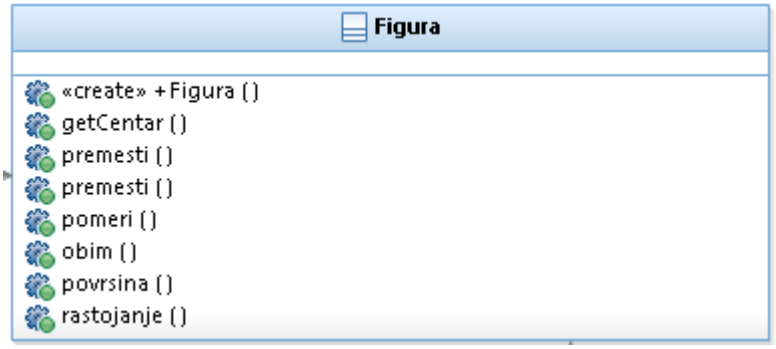
- **Tačka** u ravni zadaje se koordinatama, kojima se može pristupiti metode. Može se izračunati rastojanje do zadate tačke.
- **Figura** u ravni zadaje se tačkom koja predstavlja centar figure kojoj se može pristupiti. Obezbediti mehanizam za premeštanje figure na određeno mesto, zadavanjem koordinata centra i zadavanjem objekta klase **Tačka**, čiji konstruktor sadrži koordinate centra. Može se izračunati obim i površina figure i da se odredi rastojanje od centra figure do centra zadate figure.
- **Krug** u ravni je figura zadata poluprečnikom kome se može pristupiti odgovarajućom get( ) metodom.
- **Trougao** u ravni je figura zadata dužinama stranica kojima se može pristupiti.

NAPOMENA: Ukoliko trenutna instalacija RSA na vašem računaru nema u svom sastavu tip podataka „double“, specificirati da je atribut ili metoda tipa double i kada softver vrati poruku sa pitanjem da li želite da kreirate taj tip podataka, jer inicijalno ne postoji, potvrditi i u daljem radu će double tip podataka da se pojavljuje kao postojeći.

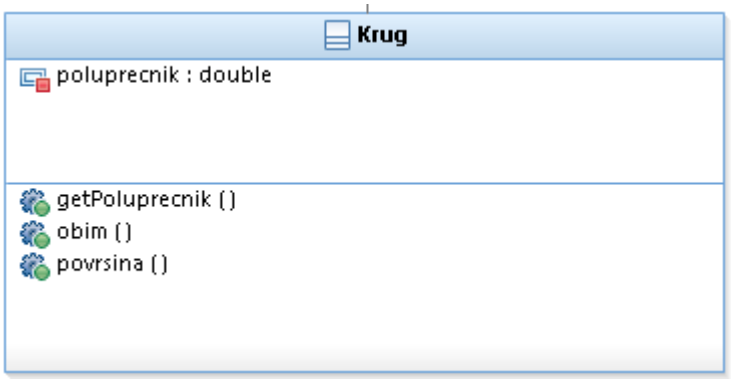
1. Kreirati novu klasu **Tačka** i dodati joj tražene atribute i metode. Klasa izgleda ovako:

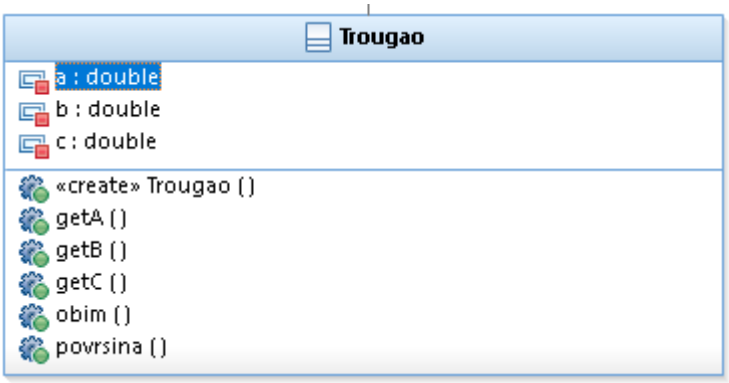


2. Kreirati klasu **Figura**, dodati joj tražene atribute i metode. Klasa izgleda ovako:

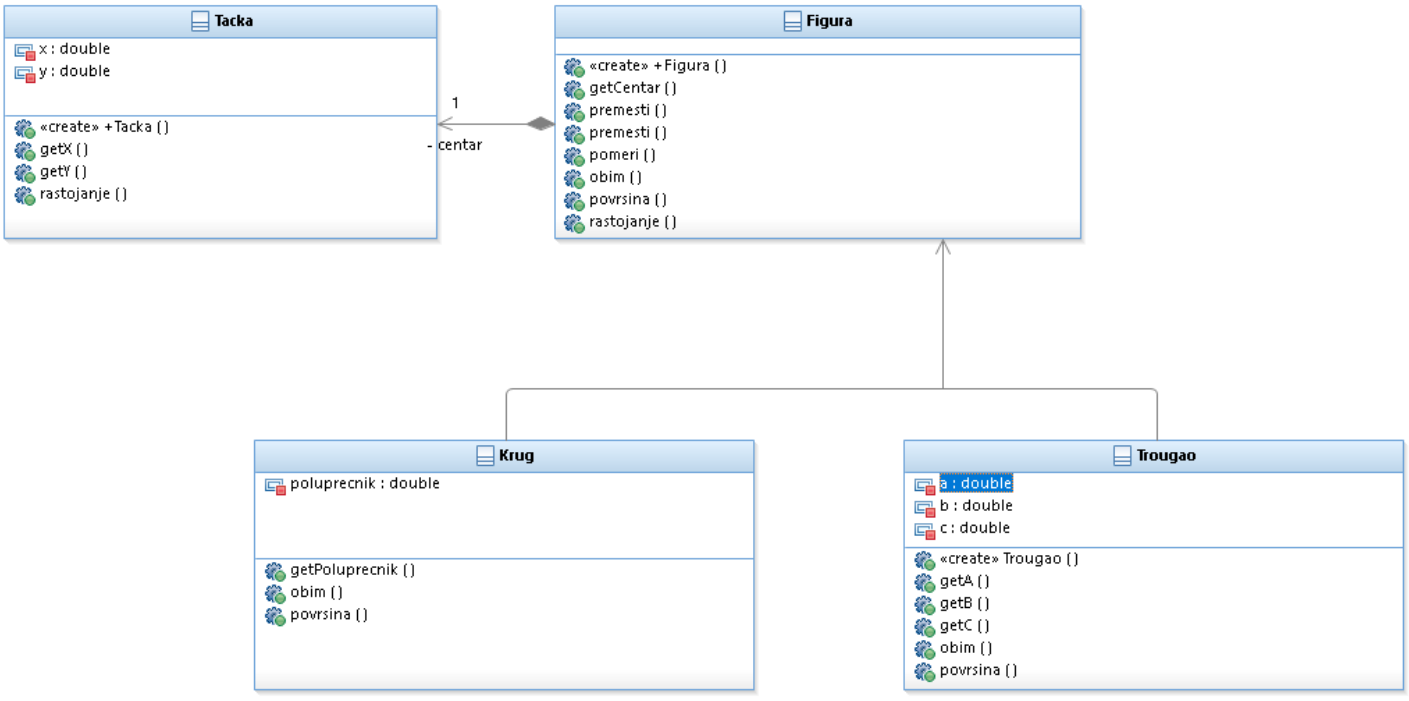


3. Izgled klasa **Krug** i **Trougao** dat je u nastavku.





4. Konačan izgleda sistema klasa dat je u nastavku:



5. Kreirani dijagram tranformisati u Java, C++ i C# programski kod.

```
/**/
/**
 * <!-- begin-UML-doc -->
 * <!-- end-UML-doc -->
 * @author tonic
 * @generated "UML to Java (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
 */
public class Tacka {
```

```
	/**
	 * <!-- begin-UML-doc -->
	 * <!-- end-UML-doc -->
	 * @generated "UML to Java (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
	 */
	private double x,y;
	/**
	 * <!-- begin-UML-doc -->
	 * <!-- end-UML-doc -->
	 * @params x,y
	 * @generated "UML to Java (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
	 */
	public Tacka(double x, double y) {
		this.x = x;
		this.y = y;
	}
	/**
	 * <!-- begin-UML-doc -->
	 * <!-- end-UML-doc -->
	 * @param s
	 * @generated "UML to Java (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
	 */
	public double getX(){
		return this.x;
	}
	/**
	 * <!-- begin-UML-doc -->
	 * <!-- end-UML-doc -->
	 * @param s
	 * @generated "UML to Java (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
	 */
	public double getY() {
		return this.y;
	}

	public double rastojanje(Tacka t){

	}
	public Proizvodjac(Skladiste s) {
		// begin-user-code
		// TODO Auto-generated constructor stub
		// end-user-code
	}

	public void proizvedi() {
	}
}
```

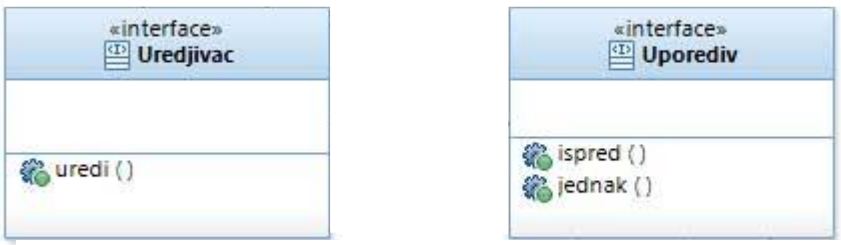
VEŽBA 3

Zadatak 1: Sortiranje uporedivih stvari

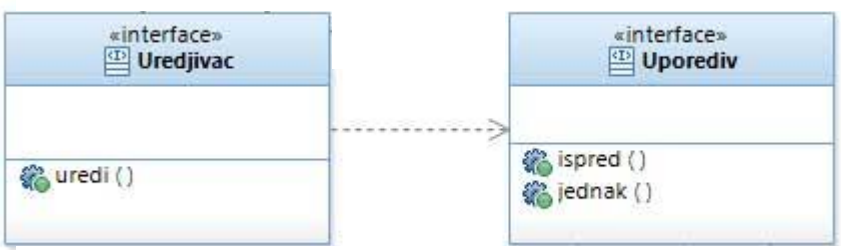
Kreirati dijagram klasa na jeziku UML sledećeg sistema tipova:

- Za apstraktnu uporedivu stvar može da se ispita da li se nalazi ispred i da li je jednaka drugoj uporednoj stvari.
- **Krug** , **datum** i **ugao** su uporedne stvari.
- Apstraktan uređivač može da uredi niz uporedivih stvari.
- Metoda **izbora** , metoda **umetanja** i metoda **podele** su uređivači.

1. Za detaljan prikaz kreiranja projekta osvrnite se vežbu 1.
2. Potrebno je da kreiramo dva interfejsa tako što iz palette menija biramo Class/Interface. Dodeljujemo im nazive **Uredjivac** i **Uporediv**.

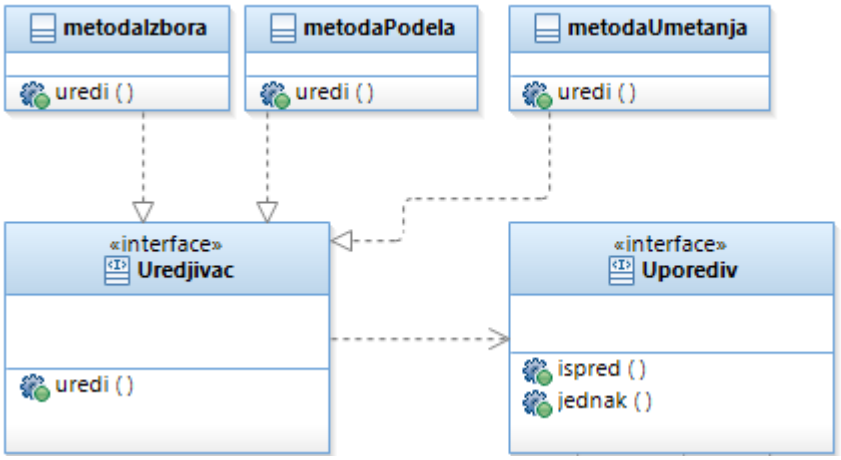


3. Neophodno je kreirati vezu izmedju ova dva interfejsa, biranjem **Dependency** veze iz palete. Dependency(zavisnost) Povezje stvari kod kojih izmena nezavisne stvari utiče na ponašanje zavisne stvari. Zavisna stvar koristi nezavisnu stvar. Grafička notacija: Uređivač zavisi od intefrejsa Uporediv.



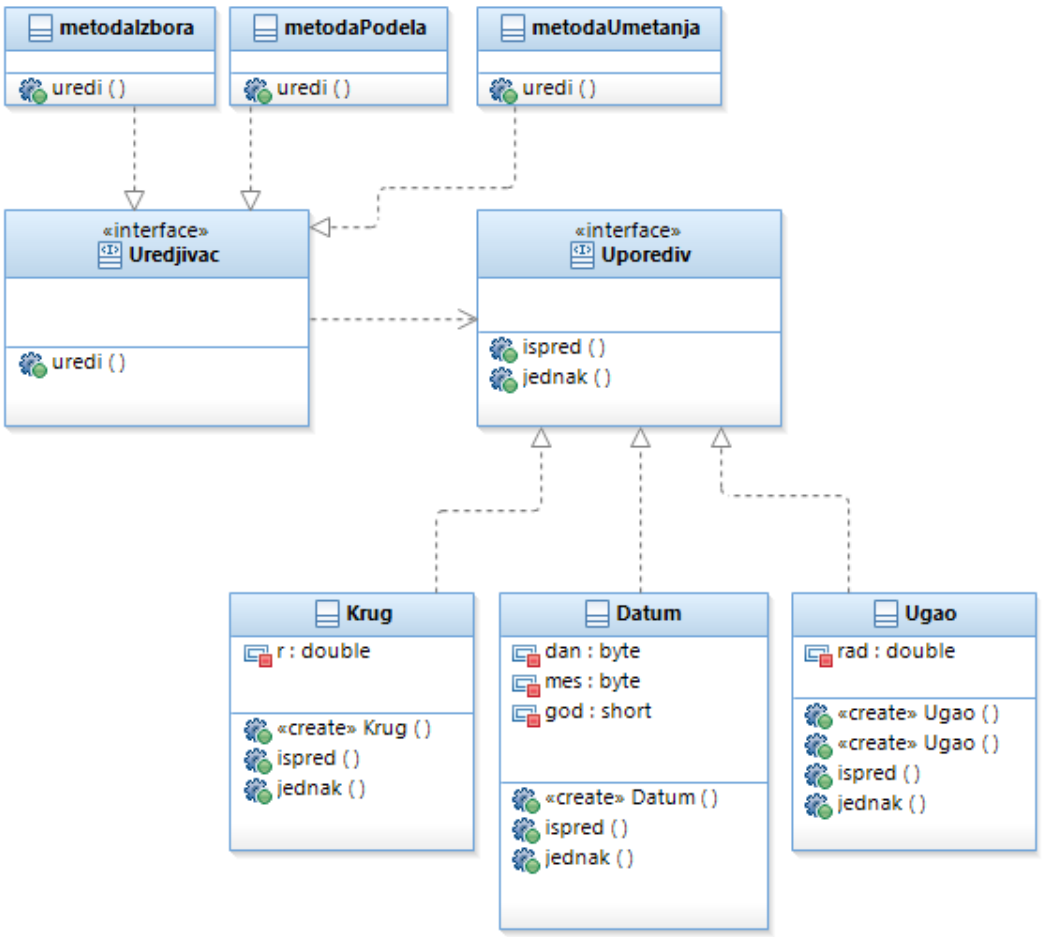
4. Interfejsi treba da sadrže metode uredi(), ispred() i jednak(). (Kreiranje metoda je prikazano u vežbama 1 i 2

5. Neophodno je napraviti 3 klase: **metodaIzbora, metodaPodela** i **metodaUmetanja** , koje sadrže metodu uredi().



6. Povežite klase sa interfejsom **Uredjivac** pomoću veze koju možete pronaći u paleti pod nazivom **Realization** . Realizacijom , jedan element garantuje da će izvršiti ono što se očekuje od drugog elementa.

7. Trebamo dodati još 3 klase (**krug , datum** i **ugao**) koje će biti povezane sa interfejsom **Uporediv** pomoću veze Realization.



8. Kreirani dijagram transformisati u Java, C++ i C# programski kod.

```
/**/
import java.util.Set;
/**
 * @author student
 *
 * @generated "UML to Java (deprecated) (com.ibm.xtools.transform.uml2.java.internal.UML2JavaTransform)"
 */
public interface Uredjivac {
	/**
	 * @param niz
	 * @generated "UML to Java (deprecated) (com.ibm.xtools.transform.uml2.java.internal.UML2JavaTransform)"
	 */
	public void uredi(Set niz);
}
```

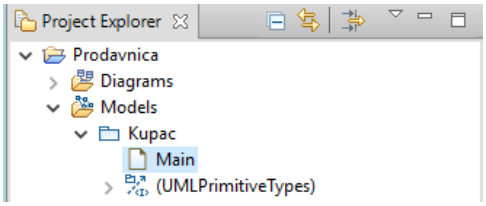
```
/**/
/**
 * @author student
 *
 * @generated "UML to Java (deprecated) (com.ibm.xtools.transform.uml2.java.internal.UML2JavaTransform)"
 */
public interface Uporediv {
	/**
	 * @param u
	 * @return
	 * @generated "UML to Java (deprecated) (com.ibm.xtools.transform.uml2.java.internal.UML2JavaTransform)"
	 */
	public Boolean ispred(Uporediv u);

	/**
	 * @param u
	 * @return
	 * @generated "UML to Java (deprecated) (com.ibm.xtools.transform.uml2.java.internal.UML2JavaTransform)"
	 */
	public Boolean jednak(Object u);
}
```

VEŽBA 4

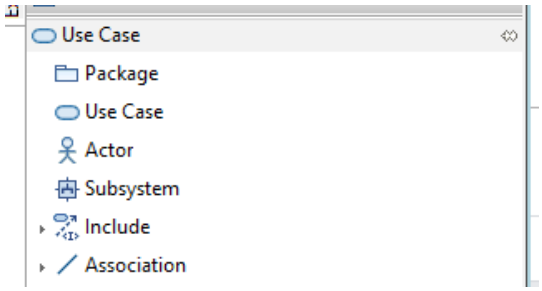
Zadatak 1: Dijagrami slučajeva korišćenja

Kreirati na jeziku UML dijagrame slučajeva korišćenja koji opisuju kupovinu u prodavnici.

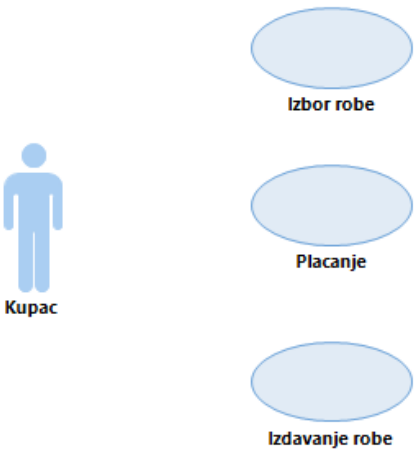
1. Pokrenite Rational Software Architect kao što je prikazano u vežbi 1.
2. Kreirajte novi UML projekat.
3. Dodelite ime projektu (Prodavnica), i izaberite prazan paket.
4. Otvorite Main fajl iz Project Explorera.



5. Iz Pallete izabrat **Use Case / Actor.**



6. Dodelite naziv **Kupac**.
7. Nakon toga je potrebno dodati slučajeve korišćenja uz pomoć **Use Case** opcije unutar palete.



Potrebno je napraviti slučajeve korišćenja koji će nam definisati **Izbor robe, Plaćanje** i **Izdavanje robe.**

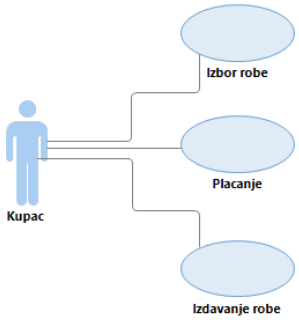
Slučaj korišćenja je zapravo opis skupa sekvenci akcija, uključujući varijante, koje subjekat obavlja da bi proizveo rezultat od vrednosti za pojedinog aktera.

Slučaj korišćenja specificira šta subjekat radi, a ne kako radi. Akter može biti čovek (korisnik) ili neki sistem. Akter je standardni stereotip klase sa posebnim grafičkim simbolom.

Relacija komunikacije

Prikazuje se **punom linijom**. Komunikaciju može inicirati akter ili slučaj korišćenja. Realizacija je dozvoljena između aktera i slučaja korišćenja i dva slučaja korišćenja koja se ne odnose na isti subjekat.

8. Sledeći korak koji treba da preduzmemo je pravljanje relacije.
9. Relacija se pravi tako što se iz palete bira **Use Case / Association** .



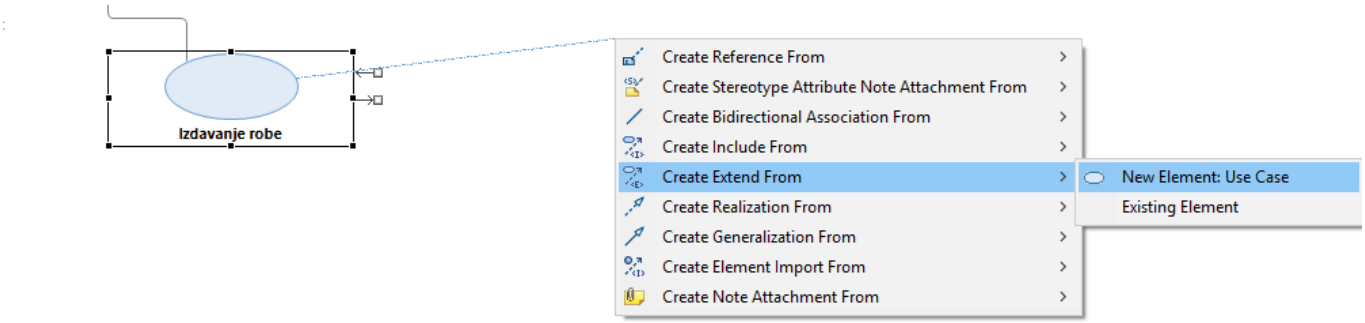
Relacija proširivanja

Prikazuje se **isprekidanom linijom** sa strelicom i natpisom **<< extend >>** . Označava proširivanje slučaja korišćenja.

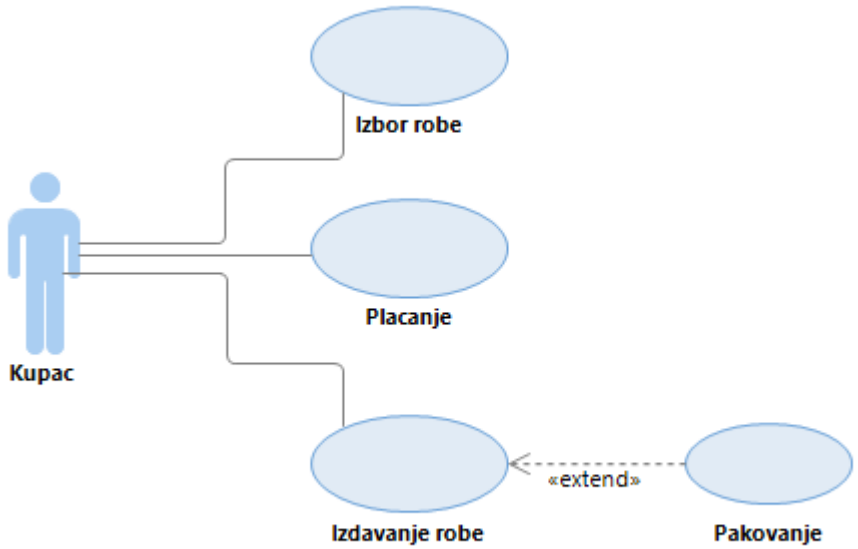
Pakovanje prema slučaju korišćenja izdavanje robe , ukazuje da izdavanje robe može obuhvatiti ponašan je specificirano u pakovanju.

Praktično, izdavanje robe može da se proširi i ispolji celokupno ponašanje opisano u Pakovanju. Koristi se da se izrazi opciono ponašanje osnovnog slučaja korišćenja.

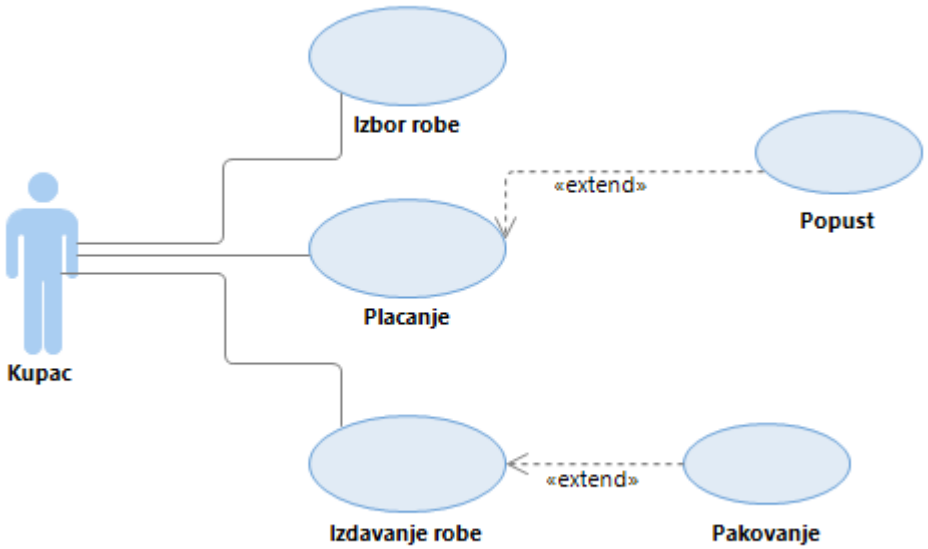
10. Relaciju proširivanja pravimo tako što izaberemo kockicu koja se pojavljuje sa strane zatim biramo **create extend form -> New Element Use Case**



11. Dodelićemo joj naziv pakovanje.



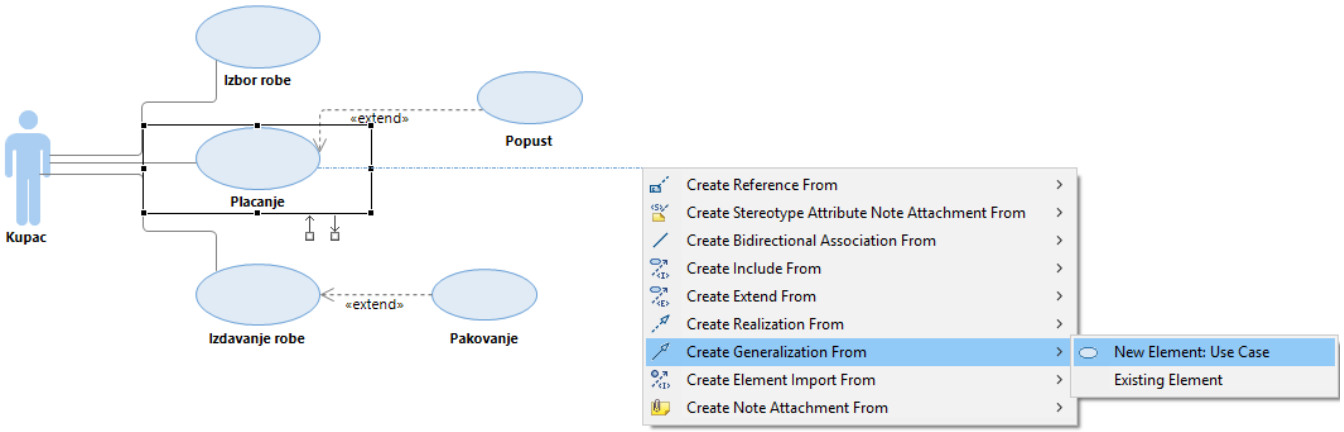
12. Takođe je neophodno dodeliti još jednu relaciju koja je vezana za plaćanje. To ćemo uraditi na isti način. Jako je bitno voditi računa o tome u kom pravcu je okrenuta strelica.



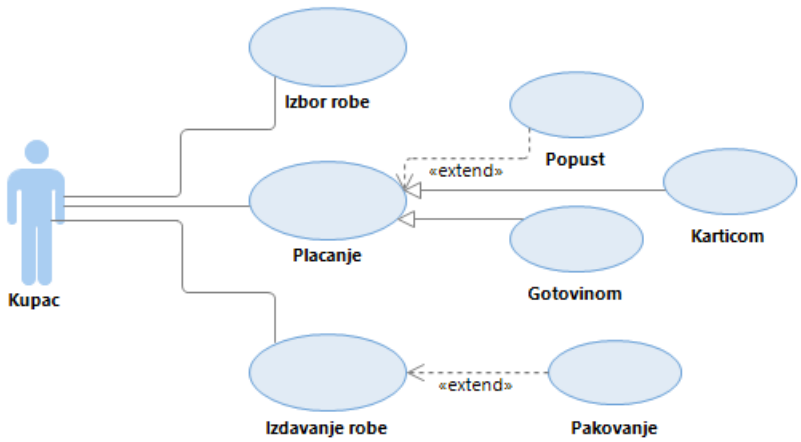
Relacija generalizacije

Prikazuje se **punom linijom sa trougaonom strelicom.** Relacija generalizacije od slučaja korišćenja Karticom koja je usmerena prema slučaju korišćenja, Plaćanje pokazuje da je slučaj korišćenja Karticom specifičan slučaj opširnijeg slučaja Plaćanje.

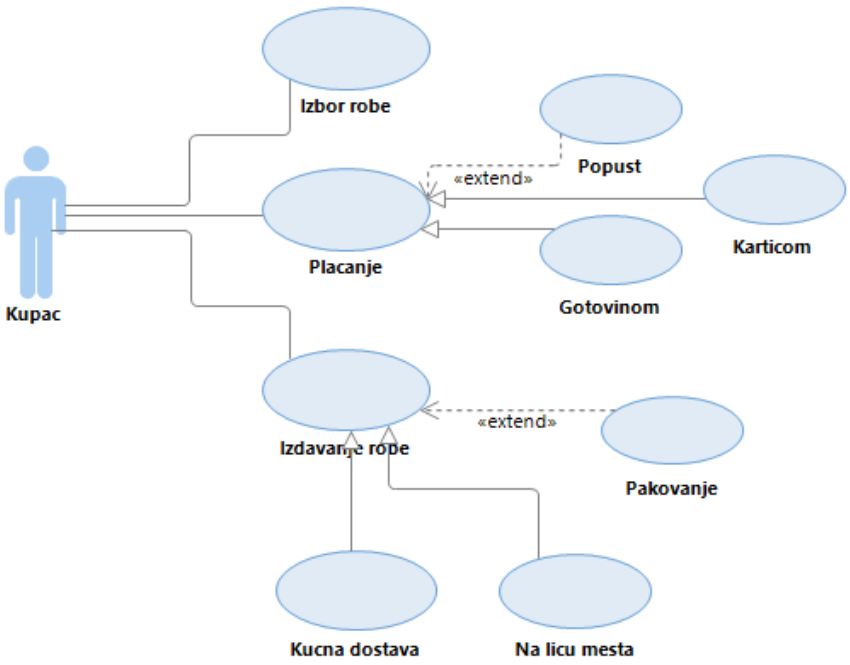
13. Neophodno je da dodamo dva slučaja korišćenja vezana za način plaćanja a to su **karticom i gotovinom** na sledeći način .



14. Nnakon kreiranja relacije generalizacije treba da dobijete sledeći prikaz.



15. Takođe treba dodati još dve relacije generalizacije vezane za izdavanje robe (**na licu mesta** , **kućna dostava**)

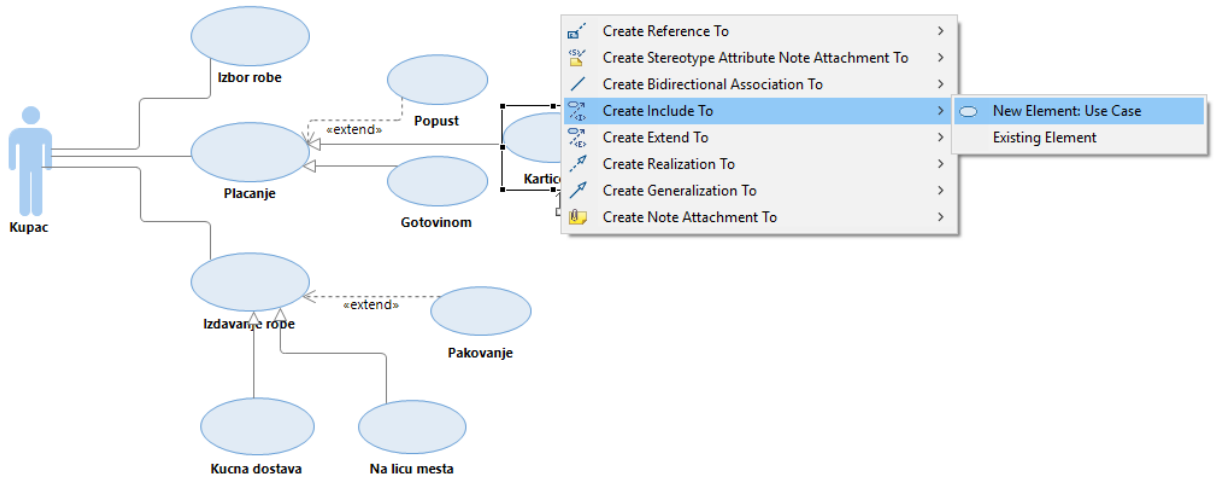


16. Dodavanje **provere identiteta** prilikom plaćanja karticom, u ovom slučaju nam je neohodna **relacija uključivanja** .

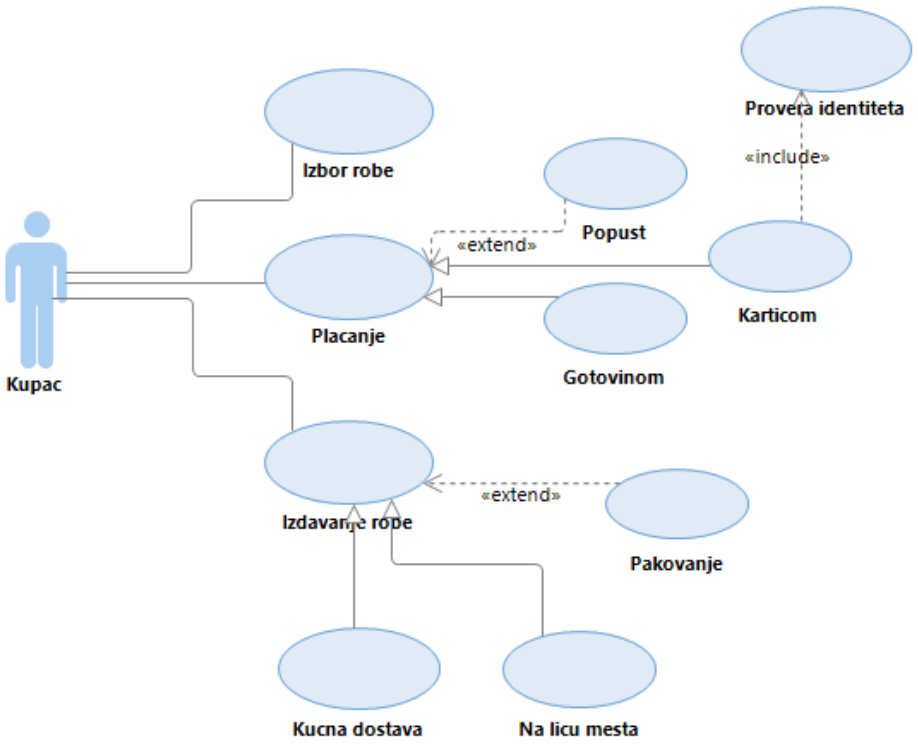
Relacija uključivanja

Prikazuje se **isprekidanom linijom** sa strelicama i natpisom **<< include >>.** Relacija uključivanja od slučaja korišćenja Karticom, prema slučaju korišćenja Provera identiteta, ukazuje da će slučaj korišćenja Karticom uključiti i ponašanje slučaja korišćenja Provera identiteta. Ponašanje opisano u slučaju korišćenja Provera identiteta je obavezno i za slučaj korišćenja Kartica.

Koristi se da opiše zajedničko ponašanje između više slučajeva korišćenja.



17. Kreiranjem relacije dobijamo sledeći prikaz.



Literatura:

[1] Laslo Kraus : Projektovanje softvera, Akademska misao, Beograd, 2013