Univerzitet u Kragujevcu
Fakultet inženjerskih nauka

Seminarski rad iz predmeta
Softverski inženjering

Tema:
Aplikacija Porodično stablo i UML dijagrami

Predmetni profesor:
Prof. Dr. Nenad Filipović

Student:
Jovan Bogdanović 609/2017

Predmetni saradnik:
Tijana Šušteršič

Kragujevac 2021.

Sadržaj:

1. Postavka zadatka i opis aplikacije

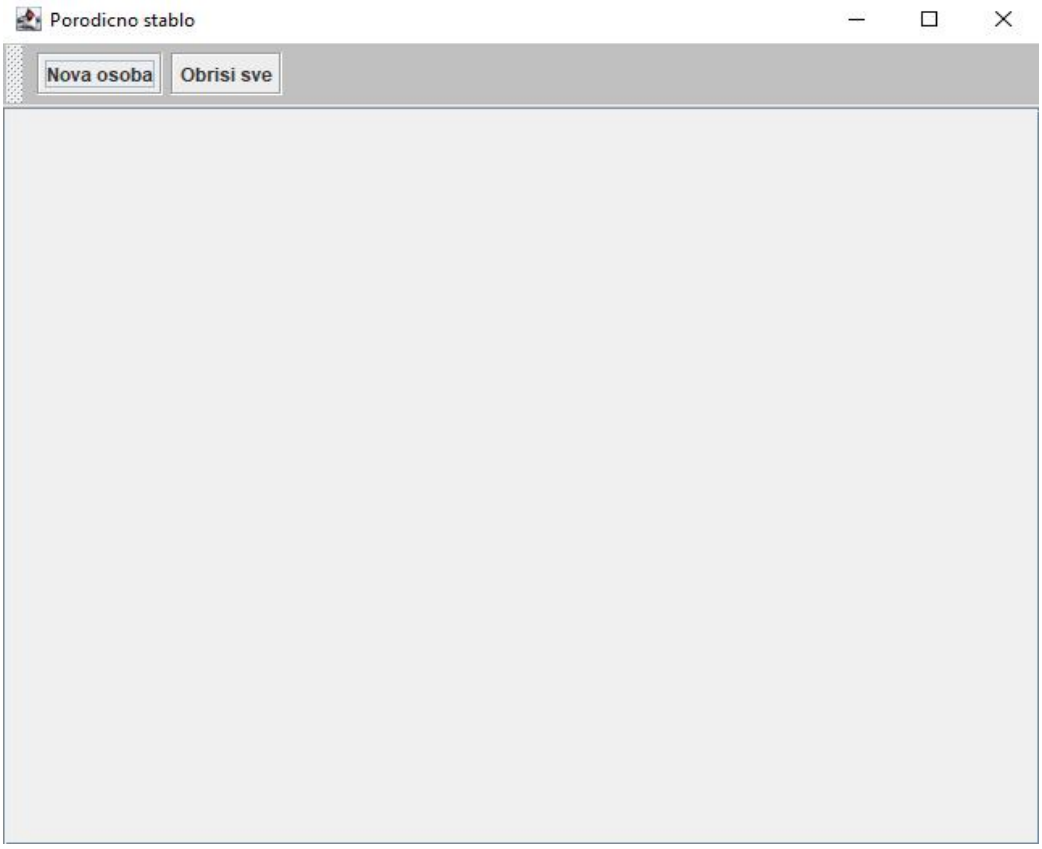
1.1 Definisanje zadatka

Potrebno je kreirati aplikaciju koja predstavlja porodično stablo. Porodično stablo treba da simulira čuvanje informacija o osobama.

Treba kreirati klase Datum i Osoba sa određenim karakteristikama. Aplikacija treba da nudi više opcija korisniku (dodavanje brata, sestre, deteta, supružnika itd.).

1.2 Uputstvo za korišćenje aplikacije

Kada se pokrene aplikacija korisnik ima praznu površinu i dva dugmeta u glavnom meniju: Nova osoba i Obriši sve.



Slika: Startovana aplikacija

Klikom na dugme Nova osoba, na praznoj površini stvara se pravougaonik bez naziva.

Klikom na dugme Obrisi sve, briše se sve na radnoj površini.

Desnim klikom na pravougaonik se ulazi u popup meni u kojem se nude sledeće opcije korisniku:

- Nova osoba (kreira nov pravougaonik na radnoj površini),
- Promeni informacije (otvara prozor u kojem se unose informacije o osobi ime, prezime, pol i datum rodjenja. Može se koristiti za ažuriranje informacija ukoliko su jednom već unete),
- Prikazi informacije (prikazuje informacije o osobi koje je korisnik predhodno uneo),
- Povezi (koristi se kada je selektovano više pravougaonika držeći dugme shift na tastaturi, za povezivanje pravougaonika u obliku stabla),
- Obrisi osobu (briše selektovane pravougaonike i sve njihove veze),
- Boja pravougaonika (otvara prozor u kojem korisnik bira boju koja se primenjuje na selektovane pravougaonike),
- Mogućnost promene veličine selektovanih pravougaonika klikom na strelicu dole/gore (ili unošenjem broja sa tastature).

Klikom na dugme Close u gornjem desnom uglu aplikacije izlazi se iz nje.

2. Objašnjenje delova programa i prikaz izvornog koda

Za izradu projektnog zadatka korišćeno je razvojno okruženje Eclipse Java 2019-06.

2.1 Klasa "Datum.java"

U ovoj klasi imamo promenljive dan, mesec i godina i njihove odgovarajuće getere i setere. Metode za validaciju datuma, proveru da li je prestupna godina, ispis unetih vrednosti u String i kreiranje datuma.

Kod:

```
package stablo;


import java.text.SimpleDateFormat;

import java.util.Calendar;


public class Datum {

	private int dan;

	private int mesec;

	private int godina;


	public int getDan() {

		return dan;

	}

	public void setDan(int dan) {

		this.dan = dan;

	}

	public int getMesec() {

		return mesec;
```

```
    }
    public void setMesec(int mesec) {
           this.mesec = mesec;
    }
    public int getGodina() {
           return godina;
    }
    public void setGodina(int godina) {
           this.godina = godina;
    }

    public String toString() {
      StringBuilder s = new StringBuilder();
      if (this.dan < 10) s.append("0");
      s.append(String.valueOf(this.dan));
      s.append("-");
      if (this.mesec < 10) s.append("0");
      s.append(String.valueOf(this.mesec));
      s.append("-");
      s.append(String.valueOf(this.godina));
      return s.toString();
  }

    public static boolean validacija(int dan, int mesec, int godina) {
           if ((dan < 1) || (dan > 31)) return false;
      if ((mesec < 1) || (mesec > 12)) return false;
      if (godina < 0) return false;
```

```
    switch (mesec) {
        case 1: return true;
        case 2: return (prestupna(godina) ? dan <= 29 : dan <= 28);
        case 3: return true;
        case 4: return dan < 31;
        case 5: return true;
        case 6: return dan < 31;
        case 7: return true;
        case 8: return true;
        case 9: return dan < 31;
        case 10: return true;
        case 11: return dan < 31;
        default: return true;
    }
}

  public static boolean prestupna(int godina) {
    if (godina % 4 != 0) {
      return false;
    } else if (godina % 400 == 0) {
      return true;
    } else if (godina % 100 == 0) {
      return false;
    } else {
      return true;
    }
  }
```

```
    public Datum() {
      Calendar currentDate = Calendar.getInstance();
      java.util.Date x = currentDate.getTime();
      SimpleDateFormat formatdd = new SimpleDateFormat("dd");
      this.dan = Integer.parseInt(formatdd.format(x));
      SimpleDateFormat formatmonth = new SimpleDateFormat("MM");
      this.mesec = Integer.parseInt(formatmonth.format(x));
      SimpleDateFormat formatyear = new SimpleDateFormat("yyyy");
      this.godina = Integer.parseInt(formatyear.format(x));
  }

    public Datum(int dan, int mesec, int godina) throws IllegalArgumentException{
      if (!validacija(dan, mesec, godina)) throw new IllegalArgumentException();
      this.dan = dan;
      this.mesec = mesec;
      this.godina = godina;
  }
}
```

2.2 Klasa "Osoba.java"

U ovoj klasi imamo promenljive za id, ime, prezime, pol i datum rodjenja.

Imamo metode koje proveravaju da li je ispravan unos podataka i metodu za kreiranje osobe i ispis njenih informacija.

Kod:

```
package stablo;

public class Osoba {
```

```
	private static int count = 0;
	public int id;
	public String ime;
	public String prezime;
	public String pol;
	public Datum datRodj;
	public Familija gore;
	public Familija par;
	public Familija dole;

	public Osoba() {
		System.out.println("Kreirana je osoba sa nultim vrednostima.");
		this.id = 0;
		this.ime = "Inconnu";
		this.prezime = "Inconnu";
		this.pol = "Nepoznato";
		this.datRodj = null ;
	}

	public Osoba(String ime, String prezime, String pol, Datum datRodj) {
		this.id = ++count;
		System.out.println("Kreirana je osoba: " + ime + " " + prezime + ",
rodjena: " + datRodj + ", pol: " + pol + ", ID: " + id);
		this.ime = ime;
		this.prezime = prezime;
		this.pol = pol;
		this.datRodj = datRodj;
	}

	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public String getIme() {
		return ime;
	}

	public void setIme(String ime) {
		for (int i = 0; i < ime.length(); i++) {
			if (!Character.isLetter(ime.charAt(i))) {
				ime = "";
				System.err.println("NEVAZECE ime, sadrzi karaktere koji
nisu slova.");
				break;
			}
```

```
			}
			this.ime = ime;
		}

		public String getPrezime() {
			return prezime;
		}

		public void setPrezime(String prezime) {
			for (int i = 0; i < prezime.length(); i++) {
				if (!Character.isLetter(prezime.charAt(i))) {
					prezime = "";
					System.err.println("NEVAZECE prezime, sadrzi karaktere
koji nisu slova.");
					break;
				}
			}
			this.prezime = prezime;
		}

		public String getPol() {
			return pol;
		}

		public void setPol(String pol) {
			if (pol == "Musko" || pol == "Zensko") {
				this.pol = pol;

			} else {
				this.pol = "Nepoznato";
				System.err.println("NEVAZECI pol : vrednost :" + pol);
			}
		}

		public Datum getDatRodj() {
			return datRodj;
		}

		public void setDatRodj(Datum datRodj) {
			this.datRodj = datRodj;
		}


}
```

2.3 Klasa "Familija.java"

Ova klasa se ne koristi za korisnički interfejs već samo u test klasi "Test.java".

U ovoj klasi se kreira koren familije (majka i otac).

Kod:

```
package stablo;

import java.util.ArrayList;

public class Familija {
       public Osoba majka;
       public Osoba otac;
       public ArrayList<Osoba> partner;
       public ArrayList<Osoba> deca;

    public Familija(Osoba majka, Osoba otac) {
        this.majka = majka;
        this.otac = otac;
        this.partner = new ArrayList<Osoba>();
        this.deca = new ArrayList<Osoba>();
    }
}
```

2.4 Klasa "Veze.java"

Ova klasa se ne koristi za korisnički interfejs već samo u test klasi "Test.java".

Klasa služi za proveru rodbinskih odnosa i ispis da li postoji tražena veza ili ne.

Kod:

```
package stablo;

import java.util.HashMap;

public class Veze {
       public void getVeze(HashMap<Integer, Osoba> sviClanovi, int id, String veza){
        if(sviClanovi.containsKey(id)){
            Osoba osoba = sviClanovi.get(id);
            switch(veza){
                case "Cerka":
                    cerka(sviClanovi, osoba);
                    break;
                case "Sin":
```

```
                    sin(sviClanovi, osoba);
                    break;
                case "Majka":
                    majka(sviClanovi, osoba);
                    break;
                case "Otac":
                    otac(sviClanovi, osoba);
                    break;
                case "Brat":
                    brat(sviClanovi, osoba);
                    break;
                case "Sestra":
                    sestra(sviClanovi, osoba);
                    break;
                case "Partner":
                    partner(sviClanovi, osoba);
                    break;

                default:
                    System.out.println("Pogresan unos.");
            }
        } else
            System.out.println("Osoba sa ID: "+ id +" nije pronadjena.");
    }

    private static void cerka(HashMap<Integer, Osoba> allMembers, Osoba osoba){
        String output = "";
        if (osoba.dole!=null){  // ako ima dece
            for( Osoba dete : osoba.dole.deca)
                if (dete.pol=="Zensko") output+= osoba.ime + " ima cerku: " +
dete.ime + " " + dete.prezime + ", datum rodjenja: " + dete.datRodj + ", pol: " +
dete.pol + ", ID: " + dete.id + " ";
        }
        if(output.length()>0)
            System.out.println(output.substring(0,output.length()-1));
        else
            System.out.println(osoba.ime + " Nema cerku.");
    }


    private static void sin(HashMap<Integer, Osoba> allMembers, Osoba osoba){
        String output = "";
        if (osoba.dole!=null){  // ako ima dece
            for( Osoba dete : osoba.dole.deca)
                if (dete.pol=="Musko") output+= osoba.ime + " ima sina: " + dete.ime
+ " " + dete.prezime + ", datum rodjenja: " + dete.datRodj + ", pol: " + dete.pol +
", ID: " + dete.id + " ";
        }
        if(output.length()>0)
```

```
            System.out.println(output.substring(0,output.length()-1));
        else
            System.out.println(osoba.ime + " Nema sina.");
    }


    private static void majka(HashMap<Integer, Osoba> allMembers, Osoba osoba){
        if(osoba.gore!=null)   // ako ima roditelja
            System.out.println(osoba.ime + " ima majku: " + osoba.gore.majka.ime + "
" + osoba.gore.majka.prezime + ", datum rodjenja: " + osoba.gore.majka.datRodj + ",
pol: " + osoba.gore.majka.pol + ", ID: " + osoba.gore.majka.id);
        else
            System.out.println(osoba.ime + " Nema majku.");
    }


    private static void otac(HashMap<Integer, Osoba> allMembers, Osoba osoba){
        if(osoba.gore!=null)    // ako ima roditelja
            System.out.println(osoba.ime + " ima oca: " + osoba.gore.otac.ime + " " +
osoba.gore.otac.prezime + ", datum rodjenja: " + osoba.gore.otac.datRodj + ", pol: "
+ osoba.gore.otac.pol + ", ID: " + osoba.gore.otac.id);
        else
            System.out.println(osoba.ime + " Nema oca.");
    }


    private static void brat(HashMap<Integer, Osoba> allMembers, Osoba osoba){
        String output = "";
        if (osoba.gore != null){    // ako ima roditelja
            for( Osoba brat : osoba.gore.deca)
                if(brat!=osoba) output += osoba.ime + " ima brata: " + brat.ime + " "
+ brat.prezime + ", datum rodjenja: " + brat.datRodj + ", pol: " + brat.pol + ", ID:
" + brat.id + " ";
        }
        if(output.length()>0)
            System.out.println(output.substring(0, output.length()-1));
        else
            System.out.println(osoba.ime + " Nema brata.");
    }


    private static void sestra(HashMap<Integer, Osoba> allMembers, Osoba osoba){
        String output = "";
        if (osoba.gore != null){    // ako ima roditelja
            for( Osoba sestra : osoba.gore.deca)
                if(sestra!=osoba) output += osoba.ime + " ima sestru: " + sestra.ime
+ " " + sestra.prezime + ", datum rodjenja: " + sestra.datRodj + ", pol: " +
sestra.pol + ", ID: " + sestra.id + " ";
        }
```

```
        if(output.length()>0)
            System.out.println(output.substring(0, output.length()-1));
        else
            System.out.println(osoba.ime + " Nema sestre.");
    }

    private static void partner(HashMap<Integer, Osoba> allMembers, Osoba osoba){
       String output = "";
        if (osoba.dole != null){    // ako ima partnera
            for( Osoba partner : osoba.par.partner)
                if(partner!=osoba) output += osoba.ime + " ima partnera: " +
partner.ime + " " + partner.prezime + ", datum rodjenja: " + partner.datRodj + ",
pol: " + partner.pol + ", ID: " + partner.id + " ";
        }
        if(output.length()>0)
            System.out.println(output.substring(0, output.length()-1));
        else
            System.out.println(osoba.ime + " Nema partnera.");
    }
}
```

2.5 Klasa “PorodicnoStablo.java”

Ova klasa se ne koristi za korisnički interfejs već samo u test klasi “Test.java”.

Kada se pozove metoda PorodicnoStablo ona sama kreira koren familije koristeći klasu Familija.java.

Takođe u ovoj klasi su definisane metode za proveru rodbinske veze koristeći klasu Veze.java. Kao i metode za dodavanje deteta, supružnika i ispravno brisanje čvora (osobe).

Kod:

```
package stablo;

import java.util.HashMap;

public class PorodicnoStablo extends Veze{
    public Familija rootFamily;
    public HashMap<Integer, Osoba> sviClanovi = new HashMap<>();    // Da se lako
pristupi svakoj osobi

      public PorodicnoStablo(Osoba majka, Osoba otac) {

        this.rootFamily = new Familija(majka, otac);
```

```
        otac.dole = rootFamily;
        majka.dole = rootFamily;
        otac.par = rootFamily;
        majka.par = rootFamily;

        otac.par.partner.add(majka);
        majka.par.partner.add(otac);
        this.sviClanovi.put(otac.id, otac);
        this.sviClanovi.put(majka.id, majka);
    }


    public void getVeze(int id, String veza){
        getVeze(this.sviClanovi, id, veza);
    }

    public void displayPorodicnoStablo(){
       sviClanovi.forEach((id, osoba) -> {
            System.out.println("ID: " + id + " " + osoba.gore + " " + osoba.dole);
        });
    }


    public void addDete(int idMajke, Osoba dete, boolean silent){
        // silent je true ako se izvrsi dok se inicijalizuje stabllo
        if(this.sviClanovi.containsKey(idMajke)){
            Osoba majka = this.sviClanovi.get(idMajke);
            if (majka.pol=="Zensko" && majka.dole!=null){    // Ako ima partnera
             majka.dole.deca.add(dete);
                dete.gore = majka.dole;
                this.sviClanovi.put(dete.id, dete);
                if (!silent)    System.out.println("Dodavanje deteta uspesno.");
                return;
            }
        }
        if (!silent)    System.out.println("Dodavanje deteta neuspesno.");
    }

    public void addPartner(int postojeciPartnerId, Osoba noviPartner){
        // validacija
        if(this.sviClanovi.containsKey(postojeciPartnerId) &&
!this.sviClanovi.containsKey(noviPartner.id)){
            Osoba postojeciPartner = this.sviClanovi.get(postojeciPartnerId);
            Familija familija = postojeciPartner.dole;
            if (familija == null){        // provera da li vec ima partnera

                if(postojeciPartner.pol=="Musko" && noviPartner.pol=="Zensko") {
                    familija = new Familija(noviPartner, postojeciPartner);
                }
```

```
                else if (postojeciPartner.pol=="Zensko" && noviPartner.pol=="Musko")
{
                    familija = new Familija(postojeciPartner, noviPartner);
                }

                postojeciPartner.dole = familija;
                noviPartner.dole = familija;

                postojeciPartner.par = familija;
                noviPartner.par = familija;

                postojeciPartner.par.partner.add(noviPartner);
                noviPartner.par.partner.add(postojeciPartner);

                this.sviClanovi.put(noviPartner.id, noviPartner);
                return;
            }
        }
    }

    public void obrisiOsobu(int idOsobe){
       if(this.sviClanovi.containsKey(idOsobe)) {
              Osoba osoba = this.sviClanovi.get(idOsobe);

              if (osoba.par != null){
                  Familija familija = osoba.par;
                 if (familija != null){
                     familija.partner.remove(osoba);
                     familija = null;
                }
                 osoba.par = null;
              }

              if (osoba.dole != null){
                  Familija familija = osoba.dole;
                 if (familija != null) {
                  for (Osoba dete : osoba.dole.deca)
                        dete.gore = null;
                     osoba.dole.deca.remove(osoba);
                     familija.deca.remove(osoba);
                     familija = null;
                }
                 osoba.dole = null;
              }

              if (osoba.gore != null){
                  Familija familija = osoba.gore;
                 if (familija != null){
                  for (Osoba roditelj : osoba.gore.deca)
```

```
                        roditelj.dole = null;
                    osoba.gore.deca.remove(osoba);
                    familija.deca.remove(osoba);
                    familija = null;
                }
                    osoba.gore = null;
            }

             this.sviClanovi.remove(osoba.id, osoba);
             System.out.println("Uspesno je obrisana osoba sa ID: " + idOsobe + " " +
osoba.ime);
             osoba = null;

      }
      else {
             System.out.println("Greska pri brisanju osobe ili nepostojeci ID.");
      }
   }

}
```

2.6 Klasa "Test.java"

Ova klasa služi za testiranje svih klasa u projektu sem klase interfejsa GUI.java.

Kod:

```
package stablo;
public class Test {
      public static void main(String[] args) {
             System.out.println(Datum.prestupna(2020) ? "2020 je prestupna godina.":
"2020 nije prestupna godina.");
             Datum datum1 = new Datum(10,8,1973);
             Datum datum2 = new Datum(19,4,1973);
        Osoba otac = new Osoba("Dejan", "Jovanovic", "Musko", datum1);
        Osoba majka = new Osoba("Dragana", "Jovanovic", "Zensko", datum2);
        PorodicnoStablo stablo = new PorodicnoStablo(majka, otac);

        Datum datum3 = new Datum(7, 8, 1998);
        Osoba jovan = new Osoba("Jovan", "Jovanovic", "Musko", datum3);

        Datum datum4 = new Datum(4, 6, 2000);
        Osoba vanja = new Osoba("Vanja", "Markovic", "Zensko", datum4);

        Datum datum5 = new Datum(22, 8, 1996);
        Osoba jovana = new Osoba("Jovana", "Vuksanovic", "Zensko", datum5);
```

```
Datum datum6 = new Datum(11, 7, 1991);
Osoba petar = new Osoba("Petar", "Vuksanovic", "Musko", datum6);

Datum datum7 = new Datum(18, 9, 2019);
Osoba bogdan = new Osoba("Bogdan", "Vuksanovic", "Musko", datum7);

stablo.addDete(2, jovan, true);
stablo.addDete(2, jovana, true);

stablo.addPartner(3, vanja);

stablo.addPartner(5, petar);

stablo.addDete(5, bogdan, true);

stablo.getVeze(1, "Cerka");
stablo.getVeze(1, "Sin");
stablo.getVeze(1, "Partner");

stablo.getVeze(2, "Cerka");
stablo.getVeze(2, "Sin");
stablo.getVeze(2, "Partner");

stablo.getVeze(3, "Otac");
stablo.getVeze(3, "Majka");
stablo.getVeze(3, "Sestra");
stablo.getVeze(3, "Partner");

stablo.getVeze(4, "Partner");

stablo.getVeze(5, "Otac");
stablo.getVeze(5, "Majka");
stablo.getVeze(5, "Brat");
stablo.getVeze(5, "Partner");
stablo.getVeze(5, "Sin");

stablo.getVeze(6, "Partner");
stablo.getVeze(6, "Sin");

stablo.obrisiOsobu(4);

stablo.getVeze(3, "Partner");
stablo.getVeze(4, "Partner");

stablo.obrisiOsobu(6);

stablo.getVeze(5, "Partner");
stablo.getVeze(5, "Sin");
```

```
        stablo.obrisiOsobu(3);

        stablo.getVeze(4, "Partner");
        stablo.getVeze(3, "Partner");
        stablo.getVeze(5, "Brat");
        stablo.getVeze(2, "Sin");
        stablo.getVeze(1, "Sin");
      }
}
```

Rešenje koje se ispisuje u konzoli:

```
2020 je prestupna godina.
Kreirana je osoba: Dejan Jovanovic, rodjena: 10-08-1973, pol: Musko, ID: 1
Kreirana je osoba: Dragana Jovanovic, rodjena: 19-04-1973, pol: Zensko, ID: 2
Kreirana je osoba: Jovan Jovanovic, rodjena: 07-08-1998, pol: Musko, ID: 3
Kreirana je osoba: Vanja Markovic, rodjena: 04-06-2000, pol: Zensko, ID: 4
Kreirana je osoba: Jovana Vuksanovic, rodjena: 22-08-1996, pol: Zensko, ID: 5
Kreirana je osoba: Petar Vuksanovic, rodjena: 11-07-1991, pol: Musko, ID: 6
Kreirana je osoba: Bogdan Vuksanovic, rodjena: 18-09-2019, pol: Musko, ID: 7
Dejan ima cerku: Jovana Vuksanovic, datum rodjenja: 22-08-1996, pol: Zensko, ID: 5
Dejan ima sina: Jovan Jovanovic, datum rodjenja: 07-08-1998, pol: Musko, ID: 3
Dejan ima partnera: Dragana Jovanovic, datum rodjenja: 19-04-1973, pol: Zensko, ID: 2
Dragana ima cerku: Jovana Vuksanovic, datum rodjenja: 22-08-1996, pol: Zensko, ID: 5
Dragana ima sina: Jovan Jovanovic, datum rodjenja: 07-08-1998, pol: Musko, ID: 3
Dragana ima partnera: Dejan Jovanovic, datum rodjenja: 10-08-1973, pol: Musko, ID: 1
Jovan ima oca: Dejan Jovanovic, datum rodjenja: 10-08-1973, pol: Musko, ID: 1
Jovan ima majku: Dragana Jovanovic, datum rodjenja: 19-04-1973, pol: Zensko, ID: 2
Jovan ima sestru: Jovana Vuksanovic, datum rodjenja: 22-08-1996, pol: Zensko, ID: 5
Jovan ima partnera: Vanja Markovic, datum rodjenja: 04-06-2000, pol: Zensko, ID: 4
Vanja ima partnera: Jovan Jovanovic, datum rodjenja: 07-08-1998, pol: Musko, ID: 3
Jovana ima oca: Dejan Jovanovic, datum rodjenja: 10-08-1973, pol: Musko, ID: 1
Jovana ima majku: Dragana Jovanovic, datum rodjenja: 19-04-1973, pol: Zensko, ID: 2
Jovana ima brata: Jovan Jovanovic, datum rodjenja: 07-08-1998, pol: Musko, ID: 3
Jovana ima partnera: Petar Vuksanovic, datum rodjenja: 11-07-1991, pol: Musko, ID: 6
Jovana ima sina: Bogdan Vuksanovic, datum rodjenja: 18-09-2019, pol: Musko, ID: 7
Petar ima partnera: Jovana Vuksanovic, datum rodjenja: 22-08-1996, pol: Zensko, ID: 5
Petar ima sina: Bogdan Vuksanovic, datum rodjenja: 18-09-2019, pol: Musko, ID: 7
Uspesno je obrisana osoba sa ID: 4 Vanja
Jovan Nema partnera.
Osoba sa ID: 4 nije pronadjena.
Uspesno je obrisana osoba sa ID: 6 Petar
Jovana Nema partnera.
Jovana ima sina: Bogdan Vuksanovic, datum rodjenja: 18-09-2019, pol: Musko, ID: 7
Uspesno je obrisana osoba sa ID: 3 Jovan
Osoba sa ID: 4 nije pronadjena.
Osoba sa ID: 3 nije pronadjena.
Jovana Nema brata.
Dragana Nema sina.
Dejan Nema sina.
```

2.7 Klasa “GUI.java”

U ovoj klasi se kreira kompletan korisnički interfejs.

Klasa u sebi sadrži ugnežđene klase tipa private i to:

- MouseHandler (proverava sta je selektovano, u ovoj klasi je omogućeno selektovanje više pravougaonika držanjem dugmeta shift na tastaturi),
- MouseMotionHandler (za pomeranje pravougaonika prevlačenjem),
- ControlPanel (u ovoj klasi se kreiraju glavni meni na vrhu i popup meni koji se otvara klikom na desni klik miša),
- ClearAction (u ovoj klasi se definiše šta se događa kada se klikne dugme Obriši sve, iz glavnog menija),
- ColorAction (ova klasa se koristi za promene boja u programu),
- ConnectAction (u ovoj klasi se definiše šta se događa kada je selektovano više pravougaonika i potom kliknuto na Poveži iz popup menija),
- DeleteAction (ova klasa služi za brisanje selektovanih pravougaonika i veza sa njima),
- NameAction (u ovoj klasi se kreira prozor u kojem se unose informacije kada se klikne na Promeni informacije u popup meniju i unose se informacije o kreiranim osobama u listu),
- InfoAction (u ovoj klasi se prikazuju informacije selektovane osobe/pravougaonika iz liste osoba koje su kreirane),
- NewNodeAction (u ovoj klasi se kreira nov čvor/pravougaonik kada se klikne na dugme Nova osoba iz glavnog ili popup menija),
- Edge (ova klasa se koristi za povezivanje čvorova/pravougaonika),
- Node (klasa ze kreiranje čvorova i rukovanje njima),
- ColorIcon (klasa za rukovanje ikonicama boja iz palete boja).

Interfejs pruža mogućnost kreiranja stabla u kojem se logički mogu zaključiti rodbinske veze samim pozicioniranjem pravougaonika (osoba) i linijama kojima su povezani.

Kod:

```
package stablo;
```

```
import java.awt.*;
import java.awt.event.*;
import java.awt.geom.Rectangle2D;
import java.util.ArrayList;
import java.util.List;
import java.util.ListIterator;
import javax.swing.*;
import javax.swing.event.*;

public class GUI extends JComponent {

    private static final int WIDE = 640;
    private static final int HIGH = 480;
    private static final int RADIUS = 75;
    private ControlPanel control = new ControlPanel();
    private int radius = RADIUS;
    private List<Node> nodes = new ArrayList<Node>();
    private List<Node> selected = new ArrayList<Node>();
    private List<Edge> edges = new ArrayList<Edge>();
    private List<Osoba> osobe = new ArrayList<>();
    private Point mousePt = new Point(WIDE / 2, HIGH / 2);
    private Rectangle mouseRect = new Rectangle();
    private boolean selecting = false;

    public static void main(String[] args) throws Exception {
        EventQueue.invokeLater(new Runnable() {
```

```
        public void run() {
            JFrame f = new JFrame("Porodicno stablo");
            f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            GUI gp = new GUI();
            f.add(gp.control, BorderLayout.NORTH);
            f.add(new JScrollPane(gp), BorderLayout.CENTER);
            f.getRootPane().setDefaultButton(gp.control.defaultButton);
            f.pack();
            f.setLocationByPlatform(true);
            f.setVisible(true);
        }
    });
}

public GUI() {
    this.setOpaque(true);
    this.addMouseListener(new MouseHandler());
    this.addMouseMotionListener(new MouseMotionHandler());
}

@Override
public Dimension getPreferredSize() {
    return new Dimension(WIDE, HIGH);
}

//ova metoda se poziva kada je napisano "repaint()",
@Override
```

```
public void paintComponent(Graphics g) {
    g.setColor(new Color(0x00f0f0f0));
    g.fillRect(0, 0, getWidth(), getHeight());
    // poziva se metoda draw() iz Egde
    for (Edge e : edges) {
        e.draw(g);
    }
    // poziva se metoda draw() iz Node i crta se node
    for (Node n : nodes) {
        n.draw(g);
    }
    if (selecting) {
        g.setColor(Color.darkGray);
        g.drawRect(mouseRect.x, mouseRect.y, mouseRect.width, mouseRect.height);
    }
}

private class MouseHandler extends MouseAdapter {

    @Override
    public void mouseReleased(MouseEvent e) {
        selecting = false;
        mouseRect.setBounds(0, 0, 0, 0);
        if (e.isPopupTrigger()) {
            showPopup(e);
        }
        e.getComponent().repaint();
```

```
    }

    @Override
    public void mousePressed(MouseEvent e) {
        mousePt = e.getPoint();
        if (e.isShiftDown()) {
            Node.selectToggle(nodes, mousePt);
        } else if (e.isPopupTrigger()) {
            Node.selectOne(nodes, mousePt);
            showPopup(e);
        } else if (Node.selectOne(nodes, mousePt)) {
            selecting = false;
        } else {
            Node.selectNone(nodes);
            selecting = true;
        }
        e.getComponent().repaint();
    }

    private void showPopup(MouseEvent e) {
        control.popup.show(e.getComponent(), e.getX(), e.getY());
    }
}

private class MouseMotionHandler extends MouseMotionAdapter {

    Point delta = new Point();
```

```
    @Override
    public void mouseDragged(MouseEvent e) {
        if (selecting) {
            mouseRect.setBounds(
                    Math.min(mousePt.x, e.getX()),
                    Math.min(mousePt.y, e.getY()),
                    Math.abs(mousePt.x - e.getX()),
                    Math.abs(mousePt.y - e.getY()));
            Node.selectRect(nodes, mouseRect);
        } else {
            delta.setLocation(
                    e.getX() - mousePt.x,
                    e.getY() - mousePt.y);
            Node.updatePosition(nodes, delta);
            mousePt = e.getPoint();
        }
        e.getComponent().repaint();
    }
}

public JToolBar getControlPanel() {
    return control;
}

// popup meni i meni na vrhu
private class ControlPanel extends JToolBar {
```

```
// kreiraju se dugmici za menije
private Action newNode = new NewNodeAction("Nova osoba");
private Action clearAll = new ClearAction("Obrisi sve");
private Action name = new NameAction("Promeni informacije");
private Action info = new InfoAction("Prikazi informacije");
private Action color = new ColorAction("Boja pravougaonika");
private Action connect = new ConnectAction("Povezi");
private Action delete = new DeleteAction("Obrisi osobu");
private JButton defaultButton = new JButton(newNode);
private JComboBox kindCombo = new JComboBox();
private ColorIcon hueIcon = new ColorIcon(Color.green);
private JPopupMenu popup = new JPopupMenu();

ControlPanel() {
    this.setLayout(new FlowLayout(FlowLayout.LEFT));
    this.setBackground(Color.lightGray);

    //ove dve stvari se dodaju na glavni meni gore
    this.add(defaultButton);
    this.add(new JButton(clearAll));

    JSpinner js = new JSpinner();
    js.setModel(new SpinnerNumberModel(RADIUS, 10, 100, 5));
    js.addChangeListener(new ChangeListener() {

        @Override
```

```
            public void stateChanged(ChangeEvent e) {
                JSpinner s = (JSpinner) e.getSource();
                radius = (Integer) s.getValue();
                Node.updateRadius(nodes, radius);
                GUI.this.repaint();
            }
        });

        // ubacivanje elemenata u meni koji se otvara na desni klik
        popup.add(new JMenuItem(newNode));
        popup.add(new JMenuItem(name));
        popup.add(new JMenuItem(info));
        popup.add(new JMenuItem(connect));
        popup.add(new JMenuItem(delete));
        popup.add(new JMenuItem(color));
        popup.add(js);
    }
}

// kad se klikne dugme Obrisi sve, ovo se pokrene
private class ClearAction extends AbstractAction {

    public ClearAction(String name) {
        super(name);
    }

    public void actionPerformed(ActionEvent e) {
```

```
            nodes.clear();
            edges.clear();
            osobe.clear();
            repaint();
        }
    }

    // za promene boja
    private class ColorAction extends AbstractAction {

        public ColorAction(String name) {
            super(name);
        }

        public void actionPerformed(ActionEvent e) {
            Color color = control.hueIcon.getColor();
            color = JColorChooser.showDialog(
                    GUI.this, "Choose a color", color);
            if (color != null) {
                Node.updateColor(nodes, color);
                control.hueIcon.setColor(color);
                control.repaint();
                repaint();
            }
        }
    }
```

```
// kada se klikne Povezi, povezuje sve selektovane elemente
private class ConnectAction extends AbstractAction {

    public ConnectAction(String name) {
        super(name);
    }

    public void actionPerformed(ActionEvent e) {
        Node.getSelected(nodes, selected);
        if (selected.size() > 1) {
            for (int i = 0; i < selected.size() - 1; ++i) {
                Node n1 = selected.get(i);
                Node n2 = selected.get(i + 1);
                edges.add(new Edge(n1, n2));
            }
        }
        repaint();
    }
}

// za brisanje selektovanih elemenata
private class DeleteAction extends AbstractAction {

    public DeleteAction(String name) {
        super(name);
    }
```

```
    public void actionPerformed(ActionEvent e) {
        ListIterator<Node> iter = nodes.listIterator();
        while (iter.hasNext()) {
            Node n = iter.next();
            if (n.isSelected()) {
                deleteEdges(n);
                iter.remove();
            }
        }
        repaint();
    }

    private void deleteEdges(Node n) {
        ListIterator<Edge> iter = edges.listIterator();
        while (iter.hasNext()) {
            Edge e = iter.next();
            if (e.n1 == n || e.n2 == n) {
                iter.remove();
            }
        }
    }
}

// ovo se pokrece kada se klikne na Promeni informacije u popup meniju
private class NameAction extends AbstractAction {

    public NameAction(String name) {
```

```
        super(name);
    }

    public void actionPerformed(ActionEvent e) {
        String ime = "";
        String prezime = "";
        String tekst = "Unesite sve podatke";
        Pol pol = Pol.Musko;
        int day = 1;
        int month = 1;
        int year = 1990;
        boolean again = true;

        while (again) {
            // inicijacizacija elementa za prozor gde se unose podaci o osobi
            JLabel txt = new JLabel("");
            txt.setText(tekst);
            JTextField fieldIme = new JTextField(12);
            fieldIme.setText(ime);
            JTextField fieldPrezime = new JTextField(12);
            fieldPrezime.setText(prezime);

            JComboBox polCombo = new JComboBox();
            for (Pol p : Pol.values()){
                polCombo.addItem(p);
            }
            polCombo.setSelectedItem(pol);
```

```
JComboBox dayCombo = new JComboBox();
for (int i = 1; i < 32; i++){
    dayCombo.addItem(i);
}
dayCombo.setSelectedItem(day);

JComboBox monthCombo = new JComboBox();
for (int i = 1; i < 13; i++){
    monthCombo.addItem(i);
}
monthCombo.setSelectedItem(month);

JComboBox yearCombo = new JComboBox();
for (int i = 1920; i < 2022; i++){
    yearCombo.addItem(i);
}
yearCombo.setSelectedItem(year);

// ubacivanje elemenata u panel
JPanel myPanel = new JPanel();
myPanel.add(txt);
myPanel.add(Box.createVerticalStrut(1));
myPanel.add(new JLabel("Ime:"));
myPanel.add(fieldIme);
myPanel.add(new JLabel("Prezime:"));
myPanel.add(fieldPrezime);
```

```
                myPanel.add(new JLabel("Pol:"));
                myPanel.add(polCombo);
                myPanel.add(new JLabel("Dan rodjenja:"));
                myPanel.add(dayCombo);

                myPanel.add(new JLabel("Mesec rodjenja:"));
                myPanel.add(monthCombo);
                myPanel.add(new JLabel("Godina rodjenja:"));
                myPanel.add(yearCombo);

                myPanel.setLayout(new GridLayout(0, 1));

                int result = JOptionPane.showConfirmDialog(null, myPanel, "Osoba",
JOptionPane.OK_CANCEL_OPTION);

                    // ako se kline ok, proverava se da li je uneto ime ili prezime,
ako nije uneseno pokrece ponovo panel sve dok  se ne unesu ili se klikne CANCEL
                if (result == JOptionPane.OK_OPTION) {
                    if (!fieldIme.getText().equals("") &&
!fieldPrezime.getText().equals("")) {

                        // preuzimanje vrednosti  iz panela
                        ime = fieldIme.getText();
                        prezime = fieldPrezime.getText();
                        pol = (Pol) polCombo.getSelectedItem();
                        day = (int) dayCombo.getSelectedItem();
                        month = (int) monthCombo.getSelectedItem();
```

```
                    year = (int) yearCombo.getSelectedItem();
                    String name = fieldIme.getText() + " " +
fieldPrezime.getText();

                    // kreiranje osobe
                    Datum datum = new Datum(day, month, year);
                    Osoba osoba = new Osoba(ime, prezime, pol.toString(), datum);
                    int id = osoba.getId();
                    osobe.add(osoba);

                    // promena imena jednog cvora
                    Node.updateName(nodes, name, id);
                    repaint();
                    again = false;
                } else {
                    // cuvanje unesenih vrednosti
                    ime = fieldIme.getText();
                    prezime = fieldPrezime.getText();
                    pol = (Pol) polCombo.getSelectedItem();
                    day = (int) dayCombo.getSelectedItem();
                    month = (int) monthCombo.getSelectedItem();
                    year = (int) yearCombo.getSelectedItem();
                    tekst = "Morate uneti ime i prezime osobe!";
                }
            } else if (result  == JOptionPane.CANCEL_OPTION) {
                again = false;
            }
```

```
        }
    }
}

private enum Pol {
    Musko, Zensko;
}

// prikaz informacija o selektovanom cvoru
private class InfoAction extends AbstractAction {

    public InfoAction(String name) {
        super(name);
    }

    public void actionPerformed(ActionEvent e) {

            JPanel myPanel = new JPanel();
            int id = -1;
            // trazi selektovani cvor i uzima njegov id
            for(Node n : nodes){
                if(n.isSelected()){
                    id = n.getId();
                }
            }

            // ukoliko cvor ima zadato ime id mu je razlicito od -1
```

```
                if(id != -1){
                    id--;
                    // uzimanje podataka o osobi i prikaz
                    Osoba osoba = osobe.get(id);
                    Datum datum = osoba.getDatRodj();
                    myPanel.add(new JLabel(osoba.ime + " " + osoba.prezime));
                    myPanel.add(new JLabel("Pol: " + osoba.pol));
                    myPanel.add(new JLabel("Datum rodjenja: " + datum));
                    myPanel.add(new JLabel("ID: " + osoba.id));

                    myPanel.setLayout(new GridLayout(0, 1));
                } else {
                    myPanel.add(new JLabel("..."));
                }
                JOptionPane.showConfirmDialog(null, myPanel, "Informacije o osobi",
JOptionPane.OK_CANCEL_OPTION);
        }
    }

    // kreiranje novog cvora klikom na Nova osoba
    private class NewNodeAction extends AbstractAction {

        public NewNodeAction(String name) {
            super(name);
        }

        public void actionPerformed(ActionEvent e) {
```

```
            Node.selectNone(nodes);
            Point p = mousePt.getLocation();
            Color color = control.hueIcon.getColor();
            String name = "...";
            Node n = new Node(p, radius, color, name);
            n.setSelected(true);
            nodes.add(n);
            repaint();
        }
    }

    // kreiranje veza (linije) izmedju cvorova
    private static class Edge {

        private Node n1;
        private Node n2;

        public Edge(Node n1, Node n2) {
            this.n1 = n1;
            this.n2 = n2;
        }

        public void draw(Graphics g) {
            Point p1 = n1.getLocation();
            Point p2 = n2.getLocation();
            g.setColor(Color.darkGray);
            g.drawLine(p1.x, p1.y, p2.x, p2.y);
```

```
    }
}

// kreiranje cvorova i metode za rukovanje cvorom
private static class Node {

    private int id = -1;
    private Point p;
    private int r;
    private Color color;
    private String name;
    private boolean selected = false;
    private Rectangle b = new Rectangle();

    public Node(Point p, int r, Color color, String name) {
        this.p = p;
        this.r = r;
        this.color = color;
        this.name = name;
        setBoundary(b);
    }

    private void setBoundary(Rectangle b) {
        b.setBounds(p.x - r, p.y - r, 2 * r, 2 * r);
    }

    public void draw(Graphics g) {
```

```
        g.setColor(this.color);
        // boji se pozadina cvora
        g.fillRect(b.x, b.y, b.width, 30);

        if (selected) {
            // boji se ivica ukoliko je selektovan cvor
            g.setColor(Color.red);
            g.drawRect(b.x, b.y, b.width, 30);
        }

        Font font = new Font("Arial", Font.BOLD, 14);
        g.setFont(font);
        g.setColor(Color.white);

        FontMetrics fm = g.getFontMetrics();
        Rectangle2D rect = fm.getStringBounds(this.name, g);
        g.drawString(this.name, (int) (b.x + b.width/2 - rect.getWidth()/2),
                (int) (b.y + 30/2 + rect.getHeight()/2));

    }
    public void setId(int i) {
        this.id = i;
    }
    public int getId() {
        return  this.id;
    }
```

```
public Point getLocation() {
    Point out = new Point((int) p.getX(),(int) Math.abs(p.getY()  + (25 - r)));
    return out;
}

public boolean contains(Point p) {
    return b.contains(p);
}

public boolean isSelected() {
    return selected;
}

public void setSelected(boolean selected) {
    this.selected = selected;
}

public static void getSelected(List<Node> list, List<Node> selected) {
    selected.clear();
    for (Node n : list) {
        if (n.isSelected()) {
            selected.add(n);
        }
    }
}
```

```
public static void selectNone(List<Node> list) {
    for (Node n : list) {
        n.setSelected(false);
    }
}

public static boolean selectOne(List<Node> list, Point p) {
    for (Node n : list) {
        if (n.contains(p)) {
            if (!n.isSelected()) {
                Node.selectNone(list);
                n.setSelected(true);
            }
            return true;
        }
    }
    return false;
}

public static void selectRect(List<Node> list, Rectangle r) {
    for (Node n : list) {
        n.setSelected(r.contains(n.p));
    }
}

public static void selectToggle(List<Node> list, Point p) {
    for (Node n : list) {
```

```
            if (n.contains(p)) {
                n.setSelected(!n.isSelected());
            }
        }
    }

    public static void updatePosition(List<Node> list, Point d) {
        for (Node n : list) {
            if (n.isSelected()) {
                n.p.x += d.x;
                n.p.y += d.y;
                n.setBoundary(n.b);
            }
        }
    }

    public static void updateRadius(List<Node> list, int r) {
        for (Node n : list) {
            if (n.isSelected()) {
                n.r = r;
                n.setBoundary(n.b);
            }
        }
    }

    public static void updateColor(List<Node> list, Color color) {
        for (Node n : list) {
```

```
            if (n.isSelected()) {
                n.color = color;
            }
        }
    }

    public static void updateName(List<Node> list, String name, int id) {
        for (Node n : list) {
            if (n.isSelected()) {
                n.name = name;
                n.id = id;
            }
        }
    }
}

private static class ColorIcon implements Icon {

    private static final int WIDE = 20;
    private static final int HIGH = 20;
    private Color color;

    public ColorIcon(Color color) {
        this.color = color;
    }

    public Color getColor() {
```

```
            return color;
        }

        public void setColor(Color color) {
            this.color = color;
        }

        public void paintIcon(Component c, Graphics g, int x, int y) {
            g.setColor(color);
            g.fillRect(x, y, WIDE, HIGH);
        }

        public int getIconWidth() {
            return WIDE;
        }

        public int getIconHeight() {
            return HIGH;
        }
    }
}
```

3. 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, a to su:

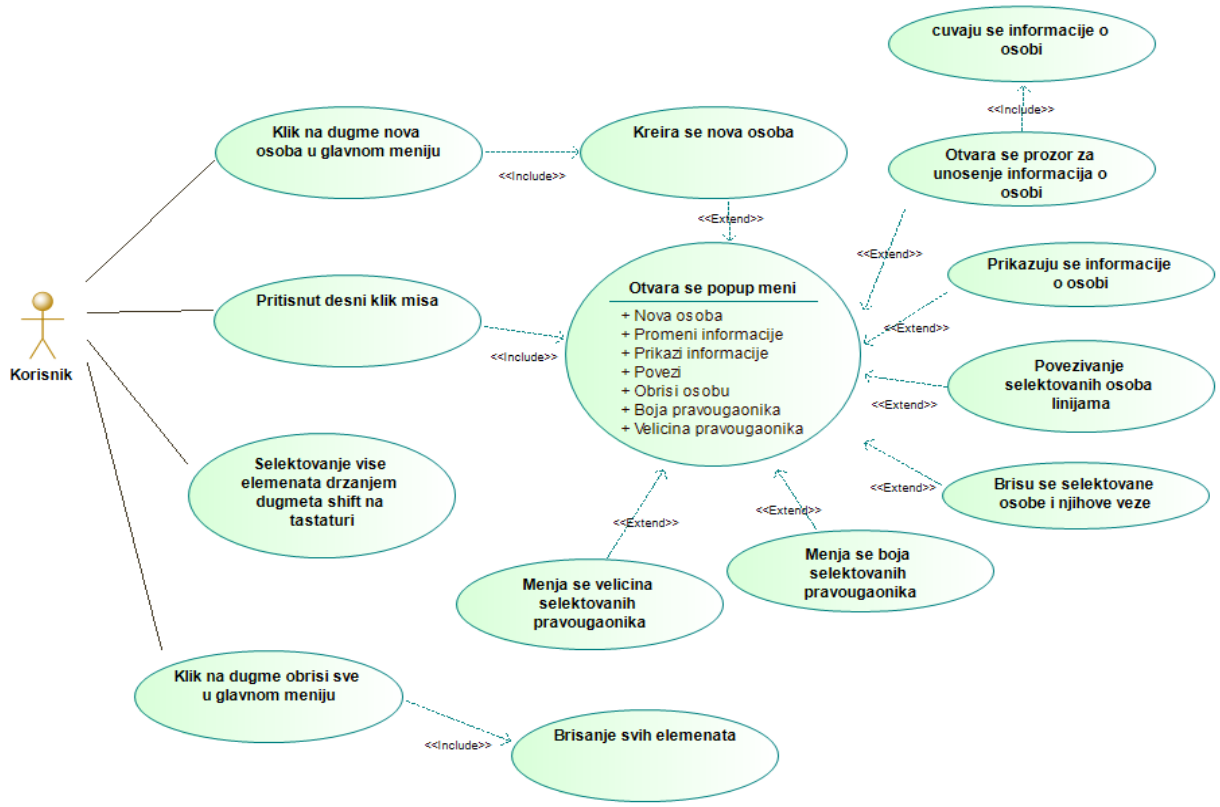
- Dijagram slučajeva korišćenja,
- Dijagram klasa,
- Dijagram sekvenci,
- Dijagram aktivnosti,
- Dijagram stanja.

3.1 Dijagram slučajeva korišćenja

Dijagram slučajeva korišćenja (engl. use case diagram) prikaz je interakcije korisnika sa sistemom koji pokazuje odnos između korisnika i različitih slučajeva korišćenja u kojima je korisnik uključen.

Slučajevi korišćenja predstavljeni su krugovima ili elipsama, a korisnici(akteri) predstavljeni su čovečuljcima.

Sledi slika dijagrama za projekat Porodično stablo.



Slika: Dijagram slučajeva korišćenja (Porodično stablo)

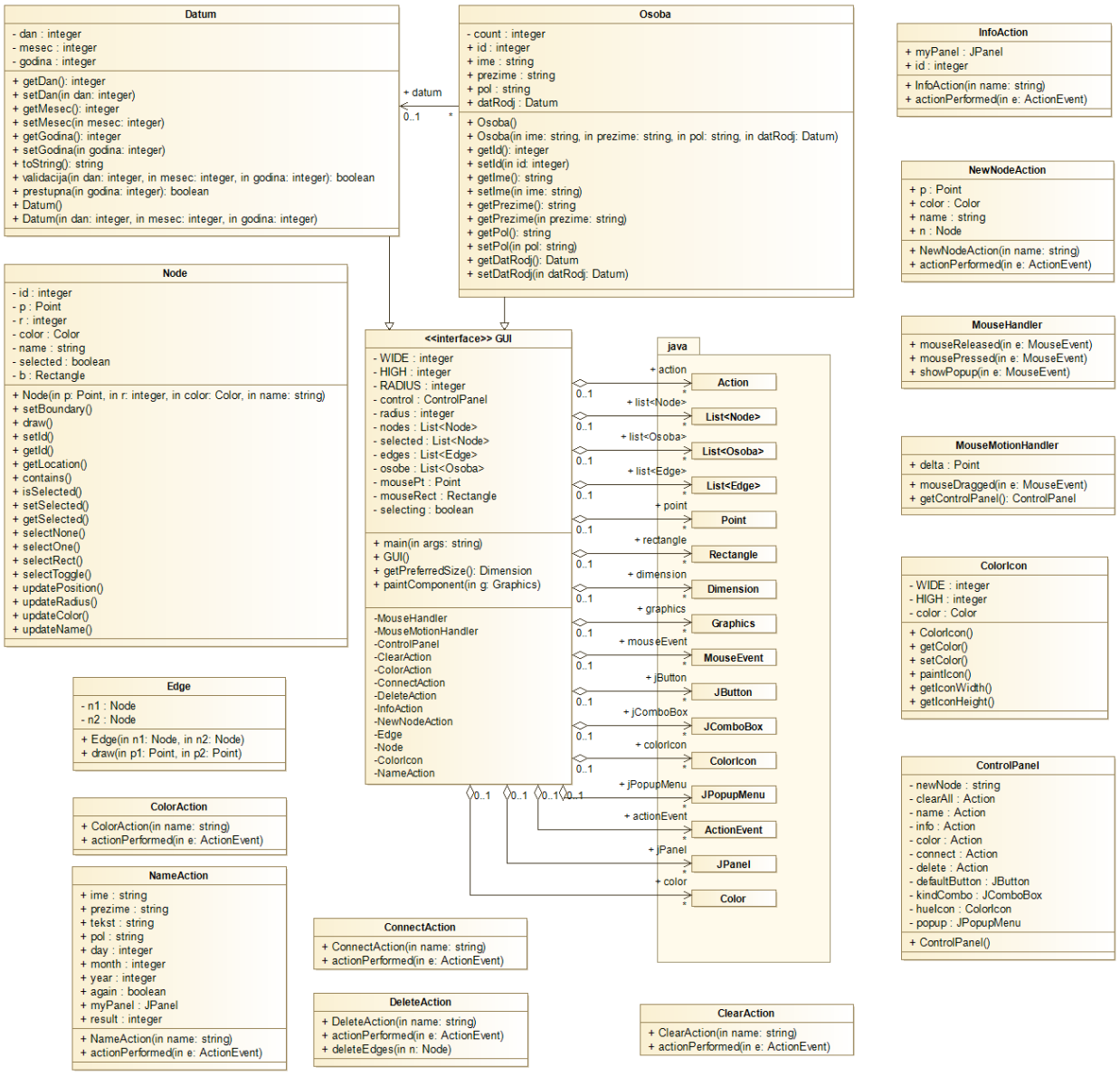
3.2 Dijagram klasa

Dijagram klasa (engl. class diagram) je vrsta strukturnog dijagrama u softverskom inžinjeringu, koji opisuje strukturu sastava objašnjavajući klase unutar sastava, njihove atribute i odnose.

Elementi dijagrama klasa su:

- stvari: klasa, interfejsi, tipovi, izuzeci, šabloni, saradnje, paketi
- relacije: zavisnosti, generalizacije, asocijacije, realizacije

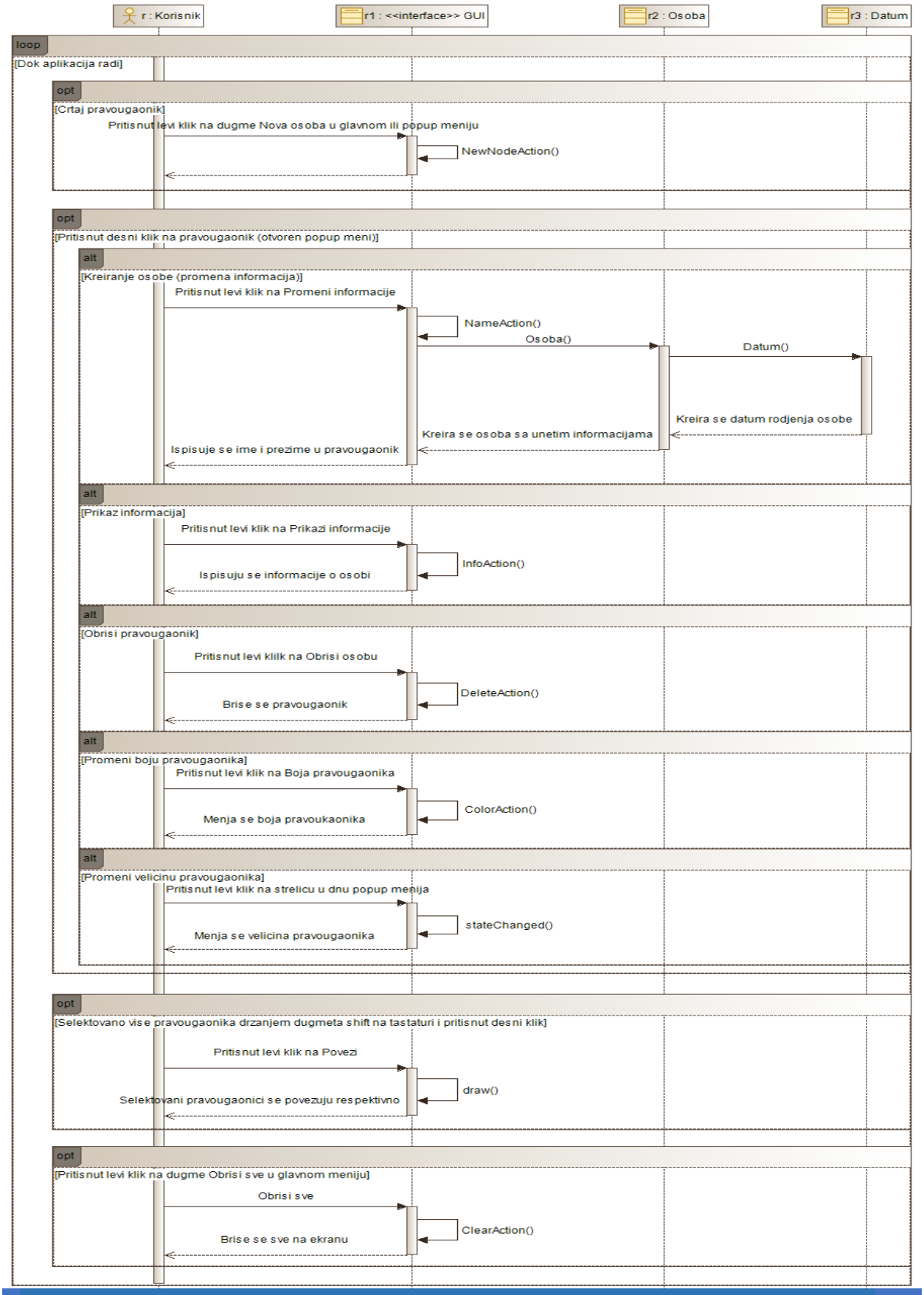
Simbol klase je pravougaonik podeljen horizontalnim linijama u odeljke(naziv klase, atributi, operacije, odgovornosti).
Sledi slika dijagrama za projekat Porodično stablo.



Slika: Dijagram klasa (Porodično stablo)

3.3 Dijagram sekvenci

Dijagram sekvenci (engl. sequence diagram) prikazuje komunikaciju između skupa objekata, koja se ostvaruje porukama koje objekti međusobno razmenjuju u cilju ostvarivanja očekivanog ponašanja. Dijagram sekvenci može da sadrži aktere, objekte i poruke. Sledi slika dijagrama za projekat Porodično stablo.


r : Korisnik
r1 : <<interface>> GUI
r2 : Osoba
r3 : Datum
loop
[Dok aplikacija radi]
opt
[Crtaj pravougaonik]
Pritisnut levi klik na dugme Nova osoba u glavnom ili popup meniju
NewNodeAction()
opt
[Pritisnut desni klik na pravougaonik (otvoren popup meni)]
alt
[Kreiranje osobe (promena informacija)]
Pritisnut levi klik na Promeni informacije
NameAction()
Osoba()
Datum()
Kreira se datum rodjenja osobe
Kreira se osoba sa unetim informacijama
Ispisuje se ime i prezime u pravougaonik
alt
[Prikaz informacija]
Pritisnut levi klik na Prikazi informacije
InfoAction()
Ispisuju se informacije o osobi
alt
[Obrisi pravougaonik]
Pritisnut levi klilk na Obrisi osobu
DeleteAction()
Brise se pravougaonik
alt
[Promeni boju pravougaonika]
Pritisnut levi klik na Boja pravougaonika
ColorAction()
Menja se boja pravoukaonika
alt
[Promeni velicinu pravougaonika]
Pritisnut levi klik na strelicu u dnu popup menija
stateChanged()
Menja se velicina pravougaonika
opt
[Selektovano vise pravougaonika drzanjem dugmeta shift na tastaturi i pritisnut desni klik]
Pritisnut levi klik na Povezi
draw()
Selektovani pravougaonici se povezuju respektivno
opt
[Pritisnut levi klik na dugme Obrisi sve u glavnom meniju]
Obrisi sve
ClearAction()
Brise se sve na ekranu

Slika: Dijagram sekvenci (Porodično stablo)

3.4 Dijagram aktivnosti

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

Dijagram aktivnosti prikazuje:

- tok aktivnosti koju izvršavaju objekti,
- eventualno i tok objekata između koraka aktivnosti.

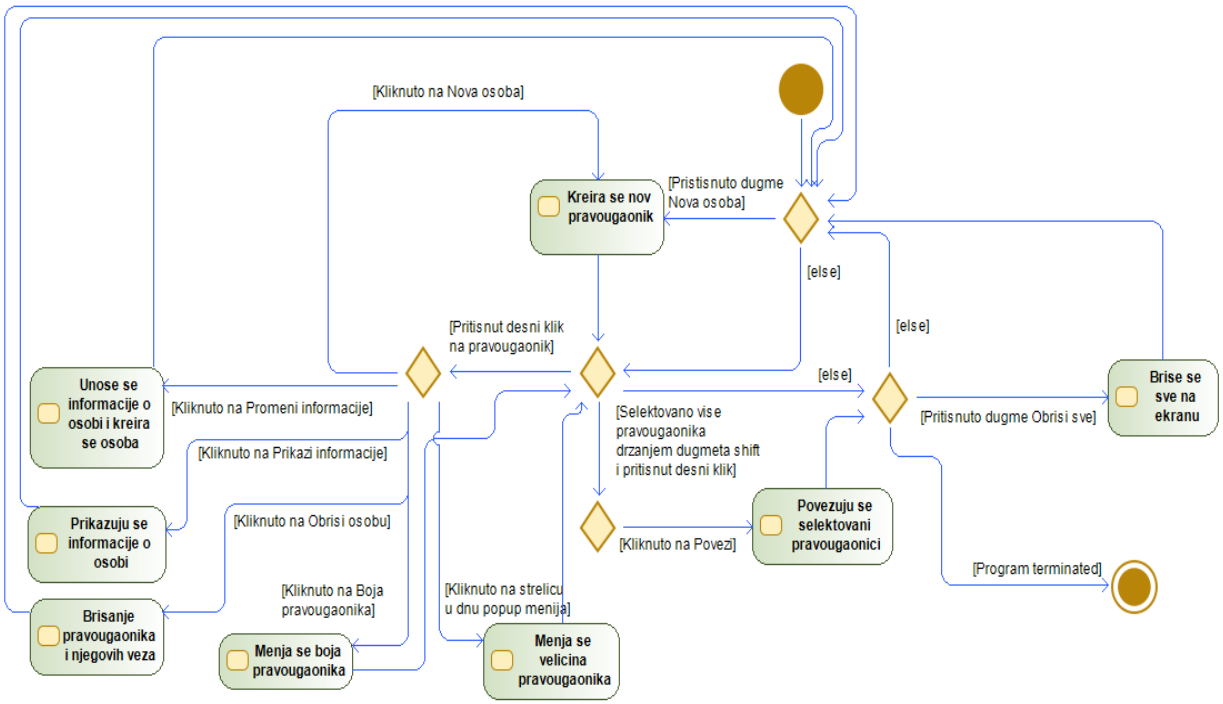
Aktivnost je specifikacija parametrizovanog ponašanja koje se izražava kroz tok izvršenja preko sekvenciranja i konkurisanja podaktivnosti. Dijagram aktivnosti je graf koji sadrži čvorove i grane.

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: Dijagram aktivnosti (Porodično stablo)

3.5 Dijagram stanja

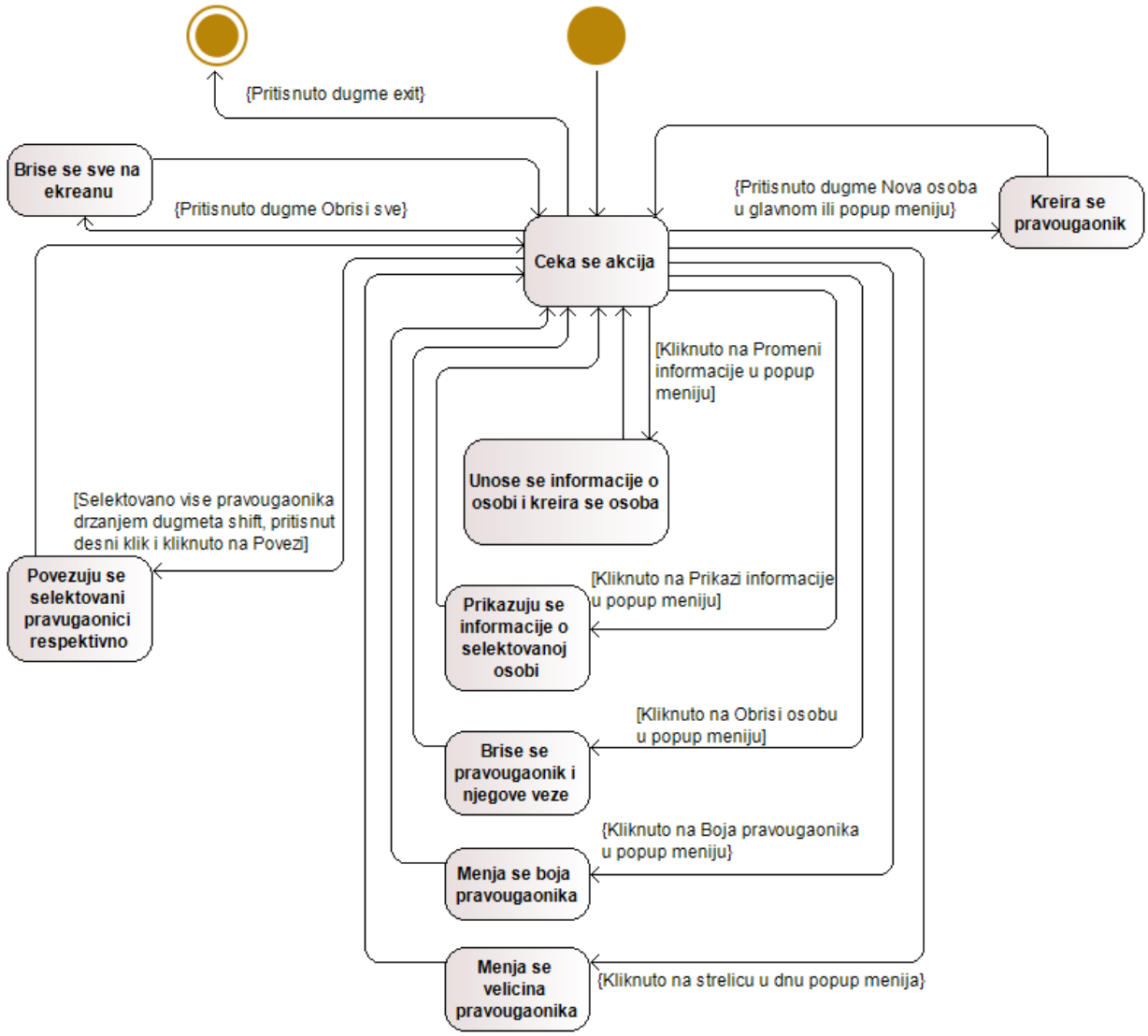
Dijagrami stanja (engl. state machine diagram) se koriste za opisivanje ponašanja sistema.

On može da opiše moguća stanja objekta kako se događaji pojavljuju.

Svaki dijagram obično predstavlja objekte jedne klase i prati različita stanja tih objekata kroz sistem.

Dijagram stanja se može upotrebiti da grafički predstavi automate konačnih stanja.

Stanje se označava pravougaonikom sa zaobljenim ivicama.



Slika: Dijagram stanja (Porodično stablo)

4. Literatura

Moodle FIN portal – Kurs Softverski inženjering godina III, semestar VI:

http://moodle.fink.rs/pluginfile.php/20007/mod_resource/content/4/UML-Uvod.pdf 28.01.2021. – 12:07

http://moodle.fink.rs/pluginfile.php/20008/mod_resource/content/3/Vezbe%201%20-%20Use%20case.pdf 28.01.2021. – 12:49

http://moodle.fink.rs/pluginfile.php/20009/mod_resource/content/3/Vezbe%202%20-%20Sequence%20diagram.pdf 29.01.2021. – 15:34

http://moodle.fink.rs/pluginfile.php/20011/mod_resource/content/5/Vezbe%203%20-%20Activity%20diagram.pdf 30.01.2021. – 11:03

http://moodle.fink.rs/pluginfile.php/20013/mod_resource/content/4/Vezbe%205%20-%20State%20machine%20diagram.pdf 31.01.2021. – 19:25

http://moodle.fink.rs/pluginfile.php/20015/mod_resource/content/3/Vezbe%206%20-%20Class%20diagram.pdf 01.01.2021. – 14:38

http://moodle.fink.rs/pluginfile.php/20018/mod_resource/content/3/Vezbe%207%20-%20Class%20diagram%20part%202.pdf 01.01.2021. – 15:51