JavaBeans Component Technology in Java

Прва лекција садржи примере који се односе на рад са JavaBean компонентама које се третирају као локалне компоненте у софтверским апликацијама на Java платформи.

Обрађене теме су:

1. Креирање JavaBean компоненте (класе)

2. Конфигурисање свих потребнихи карактеристика JavaBean компоненте.

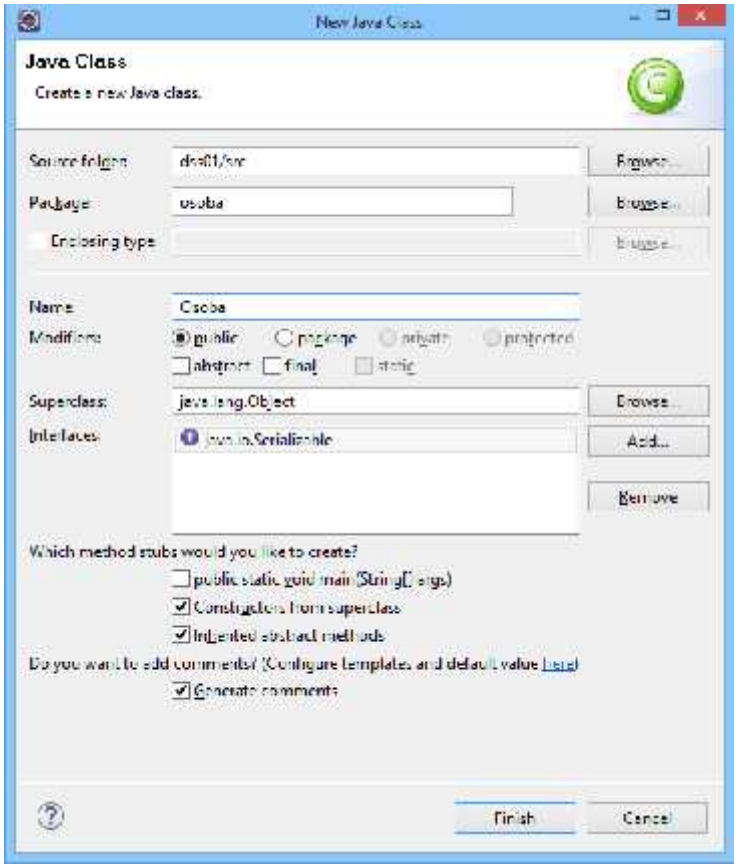
3. Серијализација JavaBean компонентаме помоћу интерфејса **`java.io.Serializable`** и помоћне класе **`SerializationUtility`**.

4. Серијализација JavaBean компонентаме помоћу интерфејса **`java.io.Externalizable`**.

5. Употреба класа за рад са динамичким структурама података, нпр. **`java.util.ArrayList`**.

Решени пример

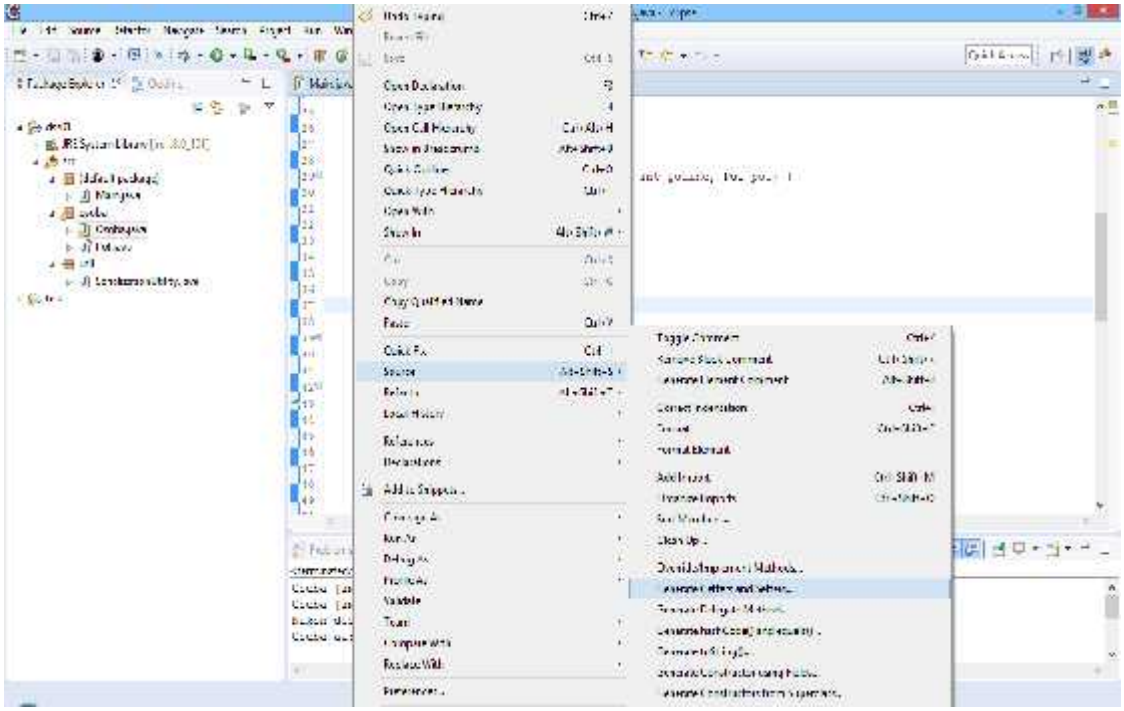
Креирати класу **Osoba** која треба да буде JavaBean. Приликом креирања класе треба обавезно назначити да класа имплементира интерфејс **java.io.Serializable**.



Након креирање класе у жељеном пакету у пројекту, треба класи додати приватне атрибуте (променљиве) , при чему не треба додавати поља која нису страни кључеви у случају повезивања са другим ентитетима. Сви атрибути класе треба да буду приватни (*private*).

```
8  public class Osoba implements Serializable {
9
10     private String ime;
11     private String prezime;
12     private int godine;
13     private Pol pol;
14
15     /**
16      * Podrazumevani konstruktor
17      */
18     public Osoba() {
19         super();
20     }
```

Следећи корак је додавање get/set метода за све приватне атрибуте којима се контролише приступ атрибутима. get/set методе се додају тако што се старује приручни мени након клика десним тастером миша било где у едитору кода за одабрану класу.

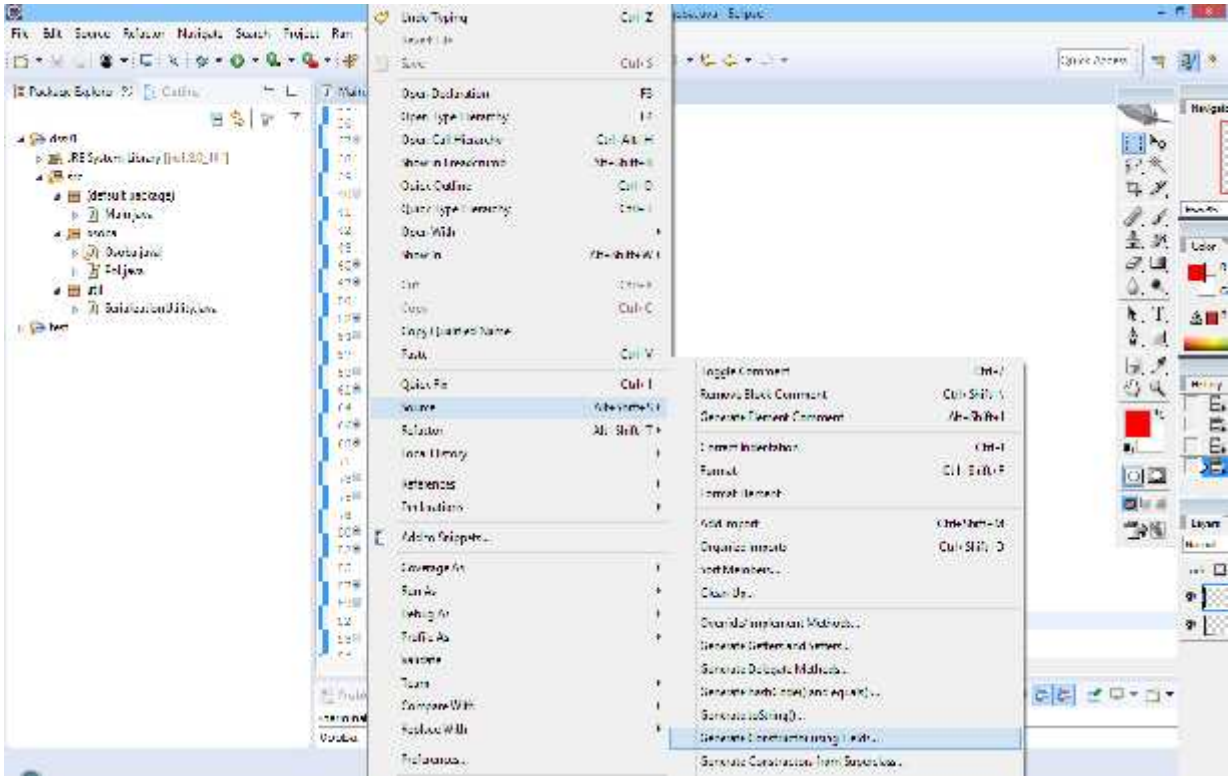


Треба одабрати креирање get/set метода за све променљиве у класи. Методе треба креирати као јавно доступне (*public*).

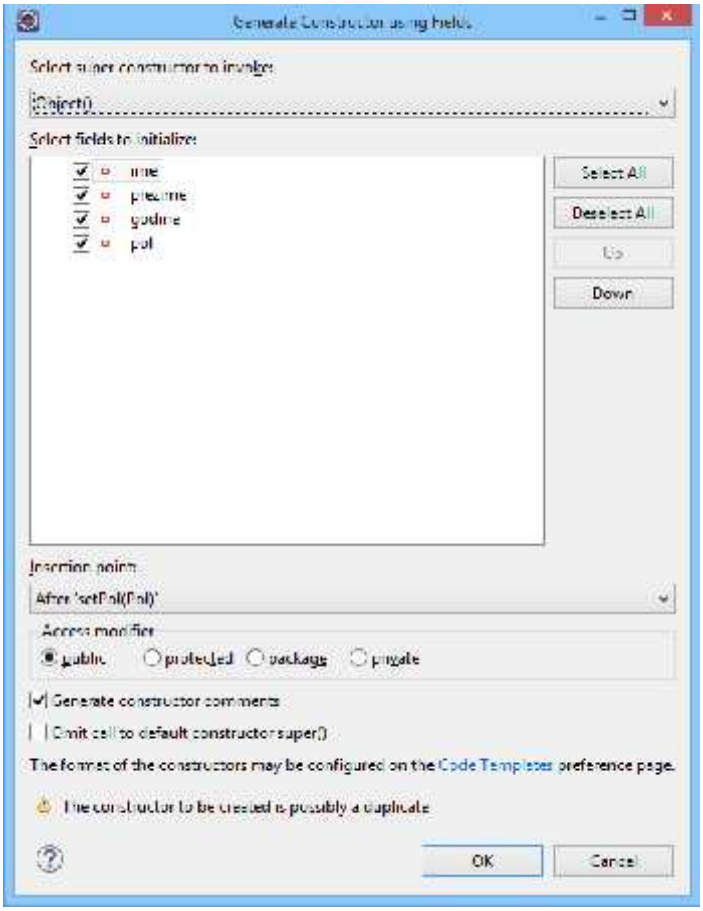
У код класе се додају get/set методе за све промењљиве. Исечак кода је приказан на следећој слици.

```
    /**
     * @return the ime
     */
    public String getIme() {
        return ime;
    }

    /**
     * @param ime the ime to set
     */
    public void setIme(String ime) {

    /**
     * @return the prezime
     */
    public String getPrezime() {

    /**
     * @param prezime the prezime to set
     */
    public void setPrezime(String prezime) {

    /**
     * @return the godine
     */
    public int getGodine() {

    /**
     * @param godine the godine to set
     */
    public void setGodine(int godine) {

    /**
     * @return the pol
     */
    public Pol getPol() {

    /**
     * @param pol the pol to set
     */
    public void setPol(Pol pol) {
```

Overriding конструктора тако да се обезбеди креирање објекта са иницијализацијом свих атрибута. Конструктор са атрибутима се додаје тако што се старује приручни мени након клика десним тастером миша било где у едитору кода за одабрану класу, и одабере опција *Generate Constructor using Fields ...*



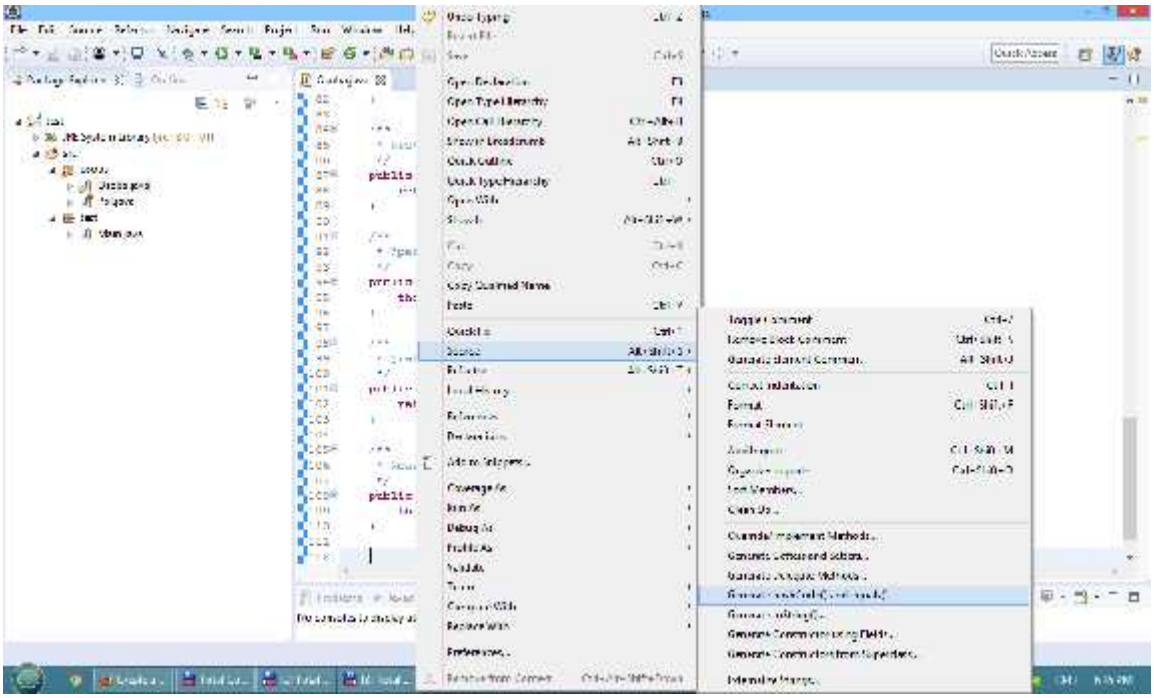
Након тога се приказује дијалог за озбор атрибута које треба укључити у конструктор.

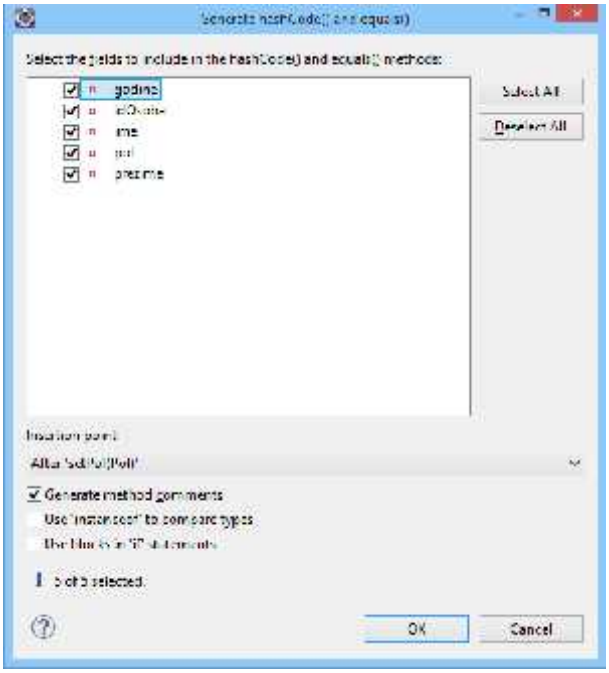


Креирани конструктор је приказан на следећој слици.

```
22    /**
23     * @param ime
24     * @param prezime
25     * @param godine
26     * @param pol
27     * @param idOsoba
28     */
29    public Osoba(String ime, String prezime, int godine, Pol pol) {
30        super();
31        this.ime = ime;
32        this.prezime = prezime;
33        this.godine = godine;
34        this.pol = pol;
35    }
```

Override метода *equals()* и *hashCode()* тако што се кликне десним тастером миша у едитор кода одабране класе и одабере опција *Generate hashCode() and equals()...*





Hashcode је број који се генерише из сваког објекта и омогућује објектима да буду коришћени у оквиру **Hashtable**. **Hashcode** Java објекта је 32-битни цео број који омогућује руковање тим објектом у оквиру **hash-based** структура података. **Hashcode** је је за сваки објекат јединствен у оквиру JVM. У оквиру класе треба имплементирати методу **hashCode()** тако да ако су два објекта једнака након поређења методом **equals()** они морају вратити исти **hashcode**.

Погледати: https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html

Серијализација применом интерфејса ***java.io.Serializable***

Креирана је помоћна класа **SerializationUtility** са статичким методама које врше серијализацију и десеријализацију објеката у датотеке. Ова помоћна класа се користи за класе које имплементирају интерфејс **java.io.Serializable** који нема метода. Методе као параметре примају објекте и називе датотека у које се врши серијализација. Серијализација и десеријализација нису под контролом објекта на којој се примењују пошто се користи помоћна класа која има јавне методе за серијализацију и десеријализацију.

```
/**
 * Serialize the given object and save it to file
 * @param obj
 * @param fileName
 * @throws IOException
 */
public static void serialize(Object obj, String fileName) throws IOException {
    FileOutputStream fos = new FileOutputStream(fileName);
    ObjectOutputStream oos = new ObjectOutputStream(fos);
    oos.writeObject(obj);
    fos.close();
}

/**
 * Deserialize to Object from the given file
 * @param fileName
 * @return
 * @throws IOException
 * @throws ClassNotFoundException
 */
public static Object deserialize(String fileName) throws IOException, ClassNotFoundException {
    FileInputStream fis = new FileInputStream(fileName);
    ObjectInputStream ois = new ObjectInputStream(fis);
    Object obj = ois.readObject();
    ois.close();
    return obj;
}
```

Позив метода за серијализацију је приказан на следећој слици.

```
// Serijalizacija osobe u datoteku
try {
    SerializationUtility.serialize(o, "o.ser");
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

// Deserijalizacija osobe iz datoteke
Osoba os = new Osoba();
try {
    os = (Osoba)SerializationUtility.deserialize("o.ser");
} catch (ClassNotFoundException | IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

Серијализација применом интерфејса ***java.io.Externalizable***

Дефинисана је класа која имплементира интерфејс **java.io.Externalizable** и која мора имплементирати методе за серијализацију и десеријализацију. Заглавље класе је

public class Zaposleni extends Osoba implements Externalizable

На овај начин класа сама контролише начин серијализације/десеријализације тако што имплементира методе:

`void writeExternal(ObjectOutput out) throws IOException`

`void readExternal(ObjectInput in) throws IOException, ClassNotFoundException`

Применом ових метода класа може заштити садржај објеката приликом серијализације тако што ће применити неки алгоритам за измену садржаја приликом уписа у фајл, и потом инверзни алгоритам за рестаурирање оригиналног садржаја, као што је приказано на следећој слици.

```
    /**
     * example of restoring parameter during deserialization (godineStaza + 18.)
     */
    @Override
    public void readExternal(ObjectInput oi) throws IOException, ClassNotFoundException {
        this.setIme((String) oi.readObject());
        this.setPrezime((String) oi.readObject());
        this.setGodine(oi.readInt());
        this.setPol((Pol)oi.readObject());
        this.godineStaza = oi.readInt();
        this.setPozicija((String)oi.readObject());
    }

    /**
     * Example of changing parameter during serialization (godineStaza + 818)
     */
    @Override
    public void writeExternal(ObjectOutput oo) throws IOException {
        oo.writeObject(this.getIme());
        oo.writeObject(this.getPrezime());
        oo.writeInt(getGodine());
        oo.writeObject(this.getPol());
        oo.writeInt(this.godineStaza + 818);
        oo.writeObject(this.pozicija);
    }
```

Позив метода за контролисану серијализацију применом **`java.io.Externalizable`** интерфејса је приказан на следећој слици.

```
        // serijalizacija
        String datoteka = "zaposleni.ser";
        FileOutputStream fos;
        try {
            fos = new FileOutputStream(datoteka);
            ObjectOutputStream oos = new ObjectOutputStream(fos);
            oos.writeObject(zap);
            oos.close();
        } catch (IOException e) {
            // TODO Auto generated catch block
            e.printStackTrace();
        }

        // deserijalizacija
        try {
            FileInputStream fis = new FileInputStream(datoteka);
            ObjectInputStream ois = new ObjectInputStream(fis);
            zap = (Zaposleni)ois.readObject();
            ois.close();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

Задаци за самостални рад

Задатак 1

Креирати класе **`Zadatak`** и **`Izvrsilac`** које описује извршавање задатака у предузећу и обезбедити да класе буду JavaBean-ови. Задатак треба описати пољима: наслов, опис, почетак (датум/време), рок (датум), трајање. Извршиоца задатка треба описати помоћу поља: име, презиме, стручна спрема, позиција, године искуства. Сваки задатак извршава стриктно један извршилац. Један извршилац може извршити више задатака.

Обезбедити следеће функционалности:

1. Кеирати неколико објеката типа **`Zadatak`** и **`Izvrsilac`**. Написати програм који проверава функционисање свих метода у класама.

2. Обезбедити серијализациу за објекте типа **`Zadatak`** применом интерфејса **`java.io.Externalizable`**. Написати код који тестира серијализацију објеката типа **`Zadatak`**.

3. У класи извршилац треба обезбедити да се сви задаци које извршава групишу у листу задатака, за шта се може користити параметризована класа **`java.util.ArrayList`**.

4. Написати програм који креира мени за управљање задацим за одабраног извршиоца. Мени садржи ставке: (1) додавање задатка, (2) брисање задатка, (3) измена задатка, (4) преглед свих задатака, и (5) снимање листе свих задатака у датотеку (серијализација).

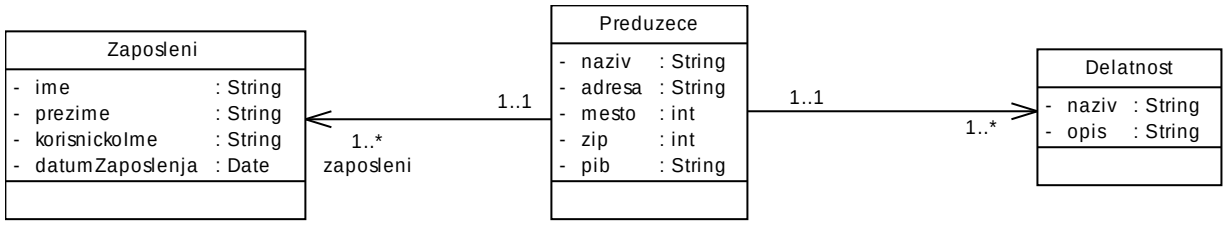
Линкови:

https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html

https://docs.oracle.com/javase/6/docs/api/java/io/Externalizable.html

Задатак 2

Креирати класе **`Preduzece`**, **`Zaposleni`** и **`Delatnost`** и успоставити релације као што је приказано на слици.



Обезбедити следеће функционалности:

1. Кеирати неколико објеката типа **`Preduzece`**, **`Zaposleni`** и **`Delatnost`**. Написати програм који проверава функционисање свих метода у класама.

2. Реализовати серијализациу за објекте типа **`Zaposleni`** применом интерфејса **`java.io.Externalizable`**. Приликом серијализације обезбедити заштитини механизам који врши енкрипцију вредности поља **`korisnickoIme`**. Енкрипција користи алгоритам који мења распоред знакова у корисничком имену тако што инвертује распоред знакова у стрингу (први на последње месту, други на претпоследње, ..., последњи на прво). Написати код који тестира серијализацију објеката типа **`Zaposleni`**.

3. У класи **`Preduzece`** треба обезбедити да се сви запослени и све делатности предузећа групишу у две независне листе, за шта се може користити параметризована класа **`java.util.ArrayList`**.

4. Написати програм који креира почетни мени за управљање основним подацима о предузећу. Основни мени садржи ставке за избор руковања: (1) подацима у предузећу, (2) запосленима, и (3) делатностима.

Када се одабере опција за основне податке о предузећу (1) тада се могу изменити подаци који су имплементирани као поља у класи **`Preduzece`**.

Када се одабере опција за руковање запосленима (2) тада се креира нови мени за управљање запосленима (**`Zaposleni`**). Мени садржи ставке: (1) додавање запосленог, (2) брисање запосленог, (3) измена података о запосленом, (4) преглед свих запослених, и (5) снимање листе свих запослених у датотеку (серијализација).

Када се одабере опција за руковање делатностима (3) тада се креира нови мени за управљање делатностима (**`Delatnost`**). Мени садржи ставке: (1) додавање делатности, (2) брисање делатности, (3) измена података о делатности, (4) преглед свих делатности, и (5) снимање листе свих делатности у датотеку (серијализација).

Линкови:

https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html

https://docs.oracle.com/javase/6/docs/api/java/io/Externalizable.html