Dist 1.2

Normal

Serijalizacija h1

- Proces pretvaranja objekta u neki standardni oblik kako bi mogao da se učita na izvornom ili nekom drugom sistemu. Takav oblik može da se sačuva i prenosi.
- Čuvaju se informacije koje su potrebne za rekonstrukciju objekta
  - tip objekta
  - unutrašnje informacije
  - reference na druge objekte
- Objekat koji se serijalizuje mora da implementira interfejs `Serializable`
- Sve promenljive koje nisu statičke i `transient` se serijalizuju
  - primer `transient` promenljive
    - ```java
      public class T implements Serializable {
          transient int a;
          int b;
      }
      ```
  - Serijalizuju se i `super` promenljive
- Čuva se kompletan graf objekata, uz eliminaciju petlji
- Veoma je preporucljivo da se doda `private static final long serialVersionUID = 1L;` unutar svake klase koja se serijalizuje
  - `serialVersionUID` predstavlja vrednost na osnovu koje onaj ko prima serijalizovan objekat može da ustanovi da li je verzija njegove klase odgovarajuća
  - > ⚠ ako se ne navede eksplicitno onda onda će se preračunati, ali to može izazvati nekozistentnosti jer različiti kompajleri drugačije to rade pa ista klasa na jednom sistemu / okruženju može da ima drugačiju vrednost na drugom
  - Pri promeni članica klase `serialVersionUID` treba ažurirati. Ukoliko se negde drugde deserijalizuje serijalizovani objekat stare klase (gde kao takav ima stari `serialVersionUID`) onda će doći do izuzetka (exception-a) jer se vrednosti `serialVersionUID` ne poklapaju
  - Detaljniji opis: https://stackabuse.com/what-is-the-serialversionuid-in-java

Serijalizacija u binarnom obliku h1

- ```java
  ObjectOutputStream oos = null;

  try {
      Student s1 = new Student("20/2020", "Pera Peric");

      FileOutputStream fos = null;
      fos = new FileOutputStream("SerializedObj.ser");
      oos = new ObjectOutputStream(fos);

      oos.writeObject(s1);
  }
  catch (Exception e) {
      e.printStackTrace();
  }
  finally {
  ```

```
        try {
          if (oos != null)
              oos.close();
        }
        catch (Exception e) {
            e.printStackTrace();
        }
    }
```

Serijalizacija u XML fajl h1

- Koristi se klasa `XMLEncoder`
- Generisani XML je
  - prenosiv
  - nezavistan od verzije
  - strukturno kompaktan
  - otporan na greške
- Da bi objekat mogao da se serijalizuje potrebno je da bude instanca Java Bean-a
- **Java Bean** je klasa koja
  - implementira interfejs `Serializable`
  - ima javni konstruktor bez argumenata
  - ima privatna polja sa javnim geterima i seterima
    - setter
      - mora da ima prefiks `set`
      - mora da prihvata neki argument (ne može da bude bez argumenata)
      - mora da ima kao povratni tip `void`
    - getter
      - mora da ima prefiks `get`
      - ne sme da prihvata neki argument
      - mora da ima kao povratni tip nešto što nije `void`
    -

      > Postoji opcija da se polja ne definišu kao privatna, ali je ipak dobra praksa da budu (zbog enkapsulacije)

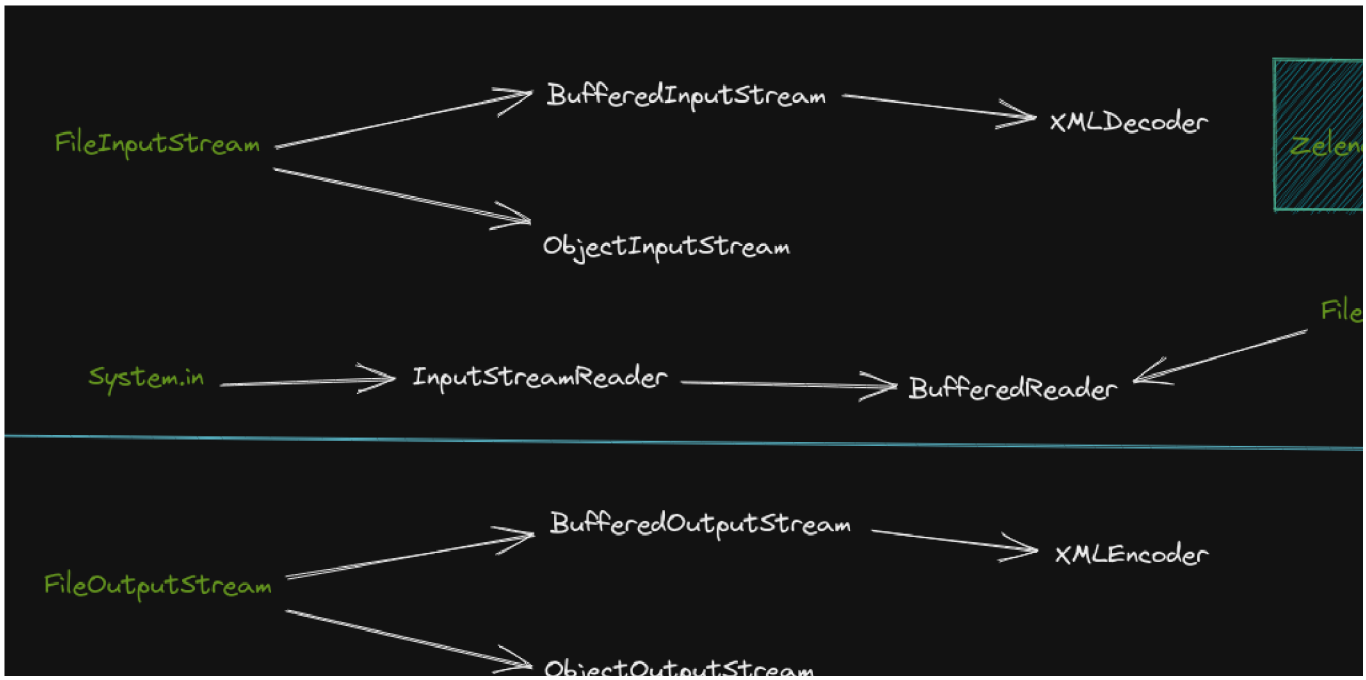
- Primer pisanja
  - java
    ```
    XMLEncoder coder = null;

    try {
        Student s = new Student("20/2020", "Pera Peric");

        FileOutputStream fos = new FileOutputStream("SerializedObj.xml");
        BufferedOutputStream bos = new BufferedOutputStream(fos);

        coder = new XMLEncoder(bos);
        coder.writeObject(s);
    }
    catch (Exception e) {
        System.out.println(e);
    }
    finally {
        coder.close();
    }
    ```

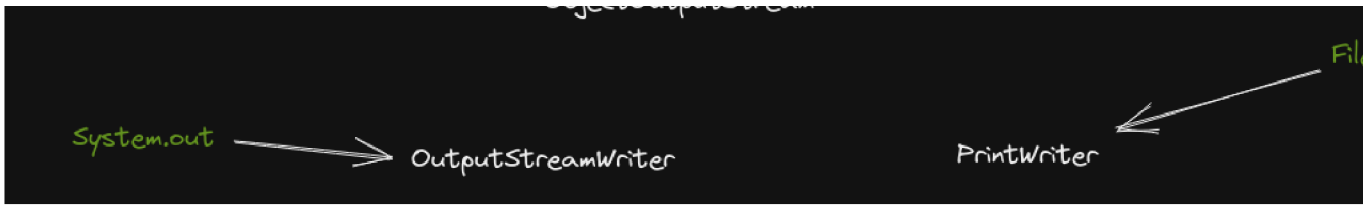
- Primer čitanja
  - java
```
XMLDecoder decoder = null;

try {
    FileInputStream fis = new FileInputStream("SerializedObj.xml");
    BufferedInputStream bis = new BufferedInputStream(fis);

    decoder = new XMLDecoder(bis);
    Student s = (Student) decoder.readObject();
    decoder.close();

    System.out.println("Student: " + s);
}
catch (Exception e) {
    System.out.println(e);
}
finally {
    decoder.close();
}
```

- Sve spomenute IO klase i njihova primena u primerima
  - Input / Read
    - java
```
// FileInputStream   - citanje fajla i pretvaranje u InputStream
// FileReader        - citanje teksta
// InputStreamReader - konvertovanje bajt toka u karakter tok (arg za BufferedReader)
// BufferedReader    - citanje cele linije / teksta
// ObjectInputStream - konvertovanje / citanje InputStream-a u objekat (deserijalizacija)
```

  - Output / Write
    - java
```
// FileOutputStream     - pisanje bajtova u fajl
// FileWriter           - pisanje teksta
// OutputStreamWriter   - konvertovanje izlaznog toka karaktera u bajt tok
// PrintWriter          - slicno kao FileWriter, sa dodacima za formatiranje i bez izuzetka
// ObjectOutputStream   - pisanje InputStream-a objekta u fajl (serijalizacija)
// BufferedOutputStream - prosledjuje se XMLEncoder-u
```

  - Wide Mode (ON) Zen Mode (ON) View Mode (ON) Grid Mode (OFF) Edit Block




System.out
OutputStreamWriter
PrintWriter