Ulaz i izlaz programa

- Računarski programi su korisni samo ako na neki način interaguju sa ostatkom sveta – ova interakcija se naziva ulaz/izlaz (engl. input/output – I/O)
- U Javi, najčešće se koristi ulaz/izlaz koji uključuje fajlove i računarske mreže putem mehanizma tokova (engl. streams) – tokovi su objekti koji podržavaju I/O naredbe
- Standardni izlaz (`System.out`) i standardni ulaz (`System.in`) su primeri tokova
- Rad sa tokovima i datotekama u Javi zahteva poznavanje mehanizma obrade grešaka pomoću izuzetaka

Rad sa tokovima

- Kada radimo sa ulazom/izlazom, razlikujemo dve osnovne kategorije podataka:
  - mašinski-formatirani podaci – sastavljeni od bajtova i
  - tekst koji mogu da čitaju ljudi – sastavljen od znakova
- Tako u Javi postoje i dve osnovne vrste tokova:
  - **Tokovi bajtova** (**byte streams**) i
  - **Tokovi znakova** (**character streams**)
- Klase za rad sa tokovima su deo paketa `java.io` koji se mora uvesti na početku programa
- Tokovi su neophodni u Java programima za rad sa fajlovima i komunikaciju preko mreže

Standardni tokovi podataka u Javi

- Java uključuje tri standardna toka podataka:

1. Standardni ulaz – tipično se koristi za učitavanje sa tastature prekos `System.in`
2. Standardni izlaz – tipično se koristi za ispis na ekran preko `System.out`
3. Standardna greška – takođe se tipično vezuje za ekran, služi za ispis grešaka preko `System.err`

Rad sa tokovima

- Objekat koji upisuje podatke u **tok bajtova** pripada nekoj od podklasa apstraktne klase `OutputStream`, dok onaj koji čita iz toka bajtova pripada nekoj od podklasa apstraktne klase `InputStream`.
- Objekat koji upisuje podatke u **tok znakova** pripada nekoj od podklasa apstraktne klase `Writer`, dok onaj koji čita iz toka znakova pripada nekoj od podklasa apstraktne klase `Reader`.
- Tokovi bajtova su korisni u mašinskoj komunikaciji, kao i za efikasno čuvanje vrlo velike količine podataka, npr. u velikim bazama podataka, ali njima se mogu obrađivati i ASCII znakovi (pošto su veličine 1 B), ali ne i UNICODE karakteri za koje je neophodan tok karaktera!

Primer 4.3 – bajt tok U/I

- Program koji učitava bajtove iz ulaznog fajla i upisuje ih u izlazni fajl – koristimo za binarne podatke i ASCII

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

public class KopiranjeBajtova {
        public static void main(String[] args) throws IOException {
                FileInputStream ulaz = null;
                FileOutputStream izlaz = null;
                try {
                        ulaz = new FileInputStream("ulazBajt.txt");
                        izlaz = new FileOutputStream("izlazBajt.txt");
                        int c;
                        while ((c = ulaz.read()) != -1) {
                                izlaz.write(c);
                        }
                }
                ...
```

Primer 4.3 – bajt tok U/I

- Program koji učitava bajtove iz ulaznog fajla i upisuje ih u izlazni fajl – koristimo za binarne podatke i ASCII

```
            ...
            catch (IOException e) {
                    System.out.println(e.getMessage());
            }
            finally {
                            if (ulaz != null) {
                                    ulaz.close();
                            }
                            if (izlaz != null) {
                                    izlaz.close();
                            }
                    }
            }
    }
```

Primer 4.4 – karakter tok U/I

- Program koji učitava bajtove iz ulaznog fajla i upisuje ih u izlazni fajl – koristiti za UNICODE karaktere

```
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class KopiranjeBajtova {
        public static void main(String[] args) throws IOException {
                FileReader ulaz = null;
                FileWriter izlaz = null;
                try {
                        ulaz = new FileReader("ulazKarakter.txt");
                        izlaz = new FileWriter("izlazKarakter.txt");
                        int c;
                        while ((c = ulaz.read()) != -1) {
                                izlaz.write(c);
                        }
                }
                ...
```

Primer 4.4 – karakter tok U/I

- Program koji učitava bajtove iz ulaznog fajla i upisuje ih u izlazni fajl – binarni za UNICODE karaktere

```
            ...
            catch (IOException e) {
                    System.out.println(e.getMessage());
            }
            finally {
                            if (ulaz != null) {
                                    ulaz.close();
                            }
                            if (izlaz != null) {
                                    izlaz.close();
                            }
                    }
            }
    }
```

Baferovani tokovi podataka

- Baferovani tokovi podataka su posebno važni prilikom rada sa velikim fajlovima
- Baferovani ulazni tok čita podatke iz dela memorije poznatog kao bafer; nativni ulazni API se poziva samo kada je bafer prazan. Slično, baferovani izlazni tok upisuje podatke u bafer i nativni izlazni API se poziva samo kada je bafer pun
- Da bi se u prethodnom primeru korisitio baferovani U/I potrebno je da se pozovu odgovarajući konstruktori:

```
ulaz = new BufferedReader(new FileReader(“unos.txt"));
izlaz = new BufferedWriter(new FileWriter(“ispis.txt"));
```

- `BufferedInputStream` i `BufferedOutputStream` kreiraju baferovane tokove bajtova, dok `BufferedReader` i `BufferedWriter` kreiraju baferovane tokove karaktera

Klasa Scanner

- Klasa `Scanner` radi kao omotač oko izvora ulaznih podataka. Izvor može biti `Reader, InputStream, String` ili `File`
- `Scanner` radi sa tokenima (najkraći smisleni niz karaktera) i delimiterima
- Primer korišćenja delimitera:

```
String ulaz = "10 caj 20 kafa 30 vocni sok";
Scanner s = new Scanner(ulaz).useDelimiter("\\s");
System.out.println(s.nextInt());
System.out.println(s.next());
System.out.println(s.nextInt());
System.out.println(s.next());
s.close();
```

Izlaz:

```
10
caj
20
kafa
```

Primer 4.5 - klasa Scanner

- Primer:

```
import java.util.Scanner;

class SkenerTest {
        public static void main(String args[]) {

                Scanner sc = new Scanner(System.in);
                System.out.println("Unesite Vas JMBG: ");
                String jmbg = sc.next();
                System.out.println("Unesite Vase ime: ");
                String ime = sc.next();
                System.out.println("Unesite Vasu platu: ");
                double plata = sc.nextDouble();
                System.out.println("JMBG:" + jmbg + " Ime:" + ime +
                                " Plata:" + plata);
                sc.close();
        }
}
```

Serijalizacija objekata prilikom U/I

- Serijalizacija objekata je postupak predstavljanja objekata kao sekvence podataka primitivnih tipova koji mogu postati elementi tokova bajtova ili karaktera. Prilikom ulaza, treba učitati serijalizovane podatke i na osnovu njih rekonsturisati kopiju originalnog objekta
- U Javi za ovu svrhu postoje gotove klase `ObjectInputStream` i `ObjectOutputStream`
- Metode za U/I rad sa objektima su `readObject()`, u `ObjectInputStream`, i `writeObject(Object obj)` u `ObjectOutputStream`. Ove metode mogu generisati IOException
- `ObjectInputStream` i `ObjectOutputStream` rade samo sa objektima klasa koje implementiraju interfejs `Serializable`

Zadatak za rad na času

- Napraviti paket `zaposleni` i u okviru njega implementirati:

1. Apstraktnu klasu `Radnik` čiji su zaštićeni podaci: ime radnika, prezime radnika, JMBG, broj tekućeg računa i koeficijent stručne spreme, a javni: metod za učitavanje podataka o radniku iz tekstualne datoteke, metod za upis imena, prezimena, broja tekućeg računa i plate radnika (za zadatu vrednost cene rada) u jedan red tekstualne datoteke i apstraktni metod za izračunavanje plate radnika.
2. klasu `AktivanRadnik` izvedenu iz apstraktne klase `Radnik,` koja kao privatni podatak sadrži varijabilni koeficijent (u skladu sa tim treba predefinisati i metod za učitavanje).
3. klasu `RadnikNaBolovanju` takođe izvedenu iz klase `Radnik`.

   U klasi `Main` kreirati platni spisak radnika jednog preduzeća na osnovu sadržaja ulazne datoteke *spisak.txt*.

Zadatak za rad na času

NAPOMENA: Platu aktivnog radnika računati po obrascu:

$$plata = (koefStrucneSpreme + varijabilniKoef) * cenaRada$$

a platu radnika na bolovanju po obrascu:

$$0.8 * koeficijentStrucneSpreme * cenaRada$$

U ulaznoj datoteci zapisana je najpre cena rada, zatim broj radnika u preduzeću, a zatim slede podaci o svim radnicima. Podaci o jednom radniku počinju linijom u kojoj je zapisan simbol + ili -. Simbol + označava da slede podaci o aktivnom radniku, a – da slede podaci o radniku koji se trenutno nalazi na bolovanju. Platni spisak ispisati u izlaznu datoteku *plate.txt*.

Napomena: za konverziju se mogu koristiti omotač klase za primitivne tipove iz `java.lang` kao što je `Integer`:

Primer za ulaz: `n = Integer.parseInt(bafUlaz.readLine());`

Primer za izlaz: `dat.write(new Double(plata).toString());`

Zadatak za vežbanje

- Napraviti paket `matematika` i u okviru njega implementirati:

1. Interfejs `Funkcija` koji sadrži metode za učitavanje parametara funkcije iz tekstualne datoteke, izračunavanje vrednosti funkcije u zadatoj tački, ispitivanje da li funkcija ima realne nule i nalaženje nula funkcije.
2. Klasu `LinearnaFunkcija` (za predstavljanje funkcija oblika $y=ax+b$) koja implementira interfejs `Funkcija`.
3. Klasu `KvadratnaFunkcija` (za predstavljanje funkcija oblika $y=ax^2+bx+c$) koja takođe implementira interfejs `Funkcija`.

U klasi `Main` definisati dve promenljive tipa interfejsa `Funkcija`, jednoj dodeliti objekat tipa `LinearnaFunkcija`, drugoj tipa `KvadratnaFunkcija`. Parametre funkcija učitati iz datoteka *funkcija1.txt* i *funkcija2.txt*, respektivno, i štampati njihove nule (ukoliko postoje) u izlazne datoteke *nule1.txt* i *nule2.txt*, respektivno.

JAVA PLATFORMA