Programski jezik Java

Ulazno izlazni podsistem

Ulazno izlazni podsistem

- Standardna biblioteka za ulazno/izlazne operacije
- Izvorišta/odredišta:
  - tastatura/konzola

Štampanje na ekran

- System.out je izlazni tok:

```
System.out.print(“Poruka”);
System.out.println(“Poruka”);
```

- Ispis se može i formatirati:

```
System.out.printf(“format”, argumenti);
System.out.printf(“%.2f %d”, (10000.0 / 3), 5);
```

Štampanje na ekran

- Metoda **`printf`**
- Prvi parametar je specifikator formata ispisa, a ostali parametri su varijable čija se vrednost štampa.
- Specifikator formata:

`%[širina][.preciznost]tip`

printf - širina

- Definiše broj cifara

```
 [širina]¦ Kako utiče na preciznost
---------+-------------------------------------------------------
   n     ¦ Štampa zadati broj cifara.
   0n    ¦ Štampa zadati broj cifara.
         ¦ Ako broj nema toliko cifara, sa leve se popunjava nulama.
```

printf – tip - brojevi

Tip	Očekivan ulaz	Format rezultat
Brojevi		
b	Boolean	Boolean
d	Integer	Signed decimal integer
o	Integer	Unsigned octal integer
u	Integer	Unsigned decimal integer
h,x	Integer	Unsigned hexadecimal int (with a, b, c, d, e, f)
H,X	Integer	Unsigned hexadecimal int (with A, B, C, D, E, F)
f	Floating point	Signed value oblika [-]dddd.dddd.
e	Floating point	Signed value oblika [-]d.dddd or e[+/-]ddd
g	Floating point	Kraći zapis od %f i %e
E	Floating point	Isto kao e; samo ima ‘E’ za eksponent
G	Floating point	Isto kao e; samo ima ‘E’ za eksponent ako je e format

printf – tip - karakteri

Tip	Očekivan ulaz	Format rezultat
Karakteri		
c	Character	Jedno slovo
s	String	Štampa string do kraja ili do zadatog broja slova
%	Ništa	Štampa znak %

printf - primeri

```
System.out.printf("celobrojni: %d\n", c);       -->    celobrojni: 356
System.out.printf("celobrojni: %6d\n", c);     -->    celobrojni:    356
System.out.printf("celobrojni: %-6d\n", c);    -->    celobrojni: 356
System.out.printf("celobrojni: %+6d\n", c);    -->    celobrojni:   +356
System.out.printf("celobrojni: %+6d\n", -c);   -->    celobrojni:   -356

System.out.printf("razlomljeni: %f\n", 3.141);    --> razlomljeni: 3.141000
System.out.printf("razlomljeni: %6.2f\n", 3.141); --> razlomljeni:   3.14
System.out.printf("razlomljeni: %e\n", 3.141);    --> razlomljeni: 3.141000e+00
System.out.printf("razlomljeni: %6.2e\n", 3.141); --> razlomljeni: 3.14e+00
System.out.printf("razlomljeni: %g\n", 3.141);    --> razlomljeni: 3.14100
```

PrntTest.java

Unos sa tastature

- System.in je ulazni tok:

```
BufferedReader in = new BufferedReader(new
   InputStreamReader(System.in));
String s = in.readLine();
```

- Alternativa je klasa Scanner koja ne učitava samo stringove:

```
Scanner sc = new Scanner(System.in);
String s = sc.nextLine();
int i = sc.nextInt();
sc.nextLine(); // kada citamo primitivne tipove, ne uklanja se ENTER
float f = sc.nextFloat();
```

Unos drugih primitivnih tipova sa tastature

- Kraće je klasom Scanner:

```
Scanner sc = new
  Scanner(System.in);
int i = sc.nextInt();
```

- Ranije se koristila wrapper klasa i njena metoda parseXxx():

```
BufferedReader in = new BufferedReader(new
  InputStreamReader(System.in));
String s = in.readLine();
int i = Integer.parseInt(s);
```

Klasa Console

- Sistemska konzola:

```
Console c = System.console();
```

- Metode:
  - c.printf("format", promenljive)
  - c.readLine()
  - c.readLine("format", promenljive)
    - ispiše promenljive u zadatom formatu, pa učita jedan red teksta
  - c.readPassword()
    - čita šifru sa tastature, bez prikazivanja slova
  - c.readPassword("format", promenljive)
    - ispiše promenljive u zadatom formatu, pa učita šifru sa tastature, bez prikazivanja slova

ConsoleTest.java

Ulazno izlazni podsistem

- standardna biblioteka za ulazno/izlazne operacije
- izvorišta/odredišta:
  - memorija
  - fajl sistem
  - mrežne konekcije
- oslanja se na tokove (streams) i čitače/pisače (reader/writer)
- nezavisno od tokova/čitača postoji i RandomAccessFile klasa i File klasa

File klasa

- za manipulaciju datotekama i direktorijumima:
  - kreiranje datoteka i direktorijuma
  - brisanje datoteka i direktorijuma
  - pristup atributima datoteka i direktorijuma
  - modifikacija naziva i atributa datoteka i direktorijuma

File klasa

- File f = new File(".");
- File f = new File("C:\\Windows");
- f = new File(f, "pera");
- f = new File(f, "..");

- if(f.exists()) ...
- if(f.isDirectory()) ...
- f.getAbsolutePath()
- f.getCanonicalPath();

Tokovi (streams) 1/4

- bazirani na bajtovima
  - prenos jednog bajta
  - prenos niza bajtova
- omogućuju prenos podataka:
  - datoteke (FileInputStream, FileOutputStream)
  - niza bajtova (ByteArrayInputStream, ByteArrayOutputStream)
  - sekvence drugih tokova (SequenceInputStream)
  - itd.

Tokovi (streams) 2/4

- osmišljeni kao mehanizam koji omogućuje unificiran pristup podacima
- isti kôd se koristi za čitanje/pisanje iz, na primer, datoteke ili mrežne konekcije
- koncept filtera - donose dodatnu funkcionalnost tokovima:
  - prenos primitivnih tipova na mašinski nezavisan način (DataInputStream, DataOutputStream)
  - baferizovan prenos podataka (BufferedInputStream, BufferedOutputStream)
  - prenos objekata (ObjectInputStream, ObjectOutputStream)
  - formatiranje podataka (PrintStream)
  - održavanje čeksuma nad podacima (CheckedOutputStream) CRC32 , Adler32

FileStreamTest primer02

Tokovi (streams) 3/4

- Metode u tokovima:
  - read() – čita jedan bajt iz toka
  - read(byte[]) – čita niz bajtova
  - skip(long n) – preskače zadati broj bajtova
  - available() – vraća broj raspoloživih bajtova iz toka koji se mogu pročitati pre blokiranja sledećeg čitanja
  - close() – zatvara tok

Tokovi (streams) 4/4

- Primer upotrebe – kopiranje sadržaja datoteke:

```
byte[] buffer = new byte[BUFFER_LENGTH];
while((read=in.read(buffer, 0,BUFFER_LENGTH)) != -1) {
   // obrada učitanog niza bajtova
   out.write(buffer);
}
```

CopyFile primer03

Čitači/pisači (readers/writers) 1/3

- ispravljaju problem sa tokovima – slabu podršku Unicode rasporedu:
  - tokovi ne prenose dobro Unicode stringove
  - poseban problem predstavljaju različite hardverske platforme (*little-endian*, *big-endian*)
- čitači/pisači ne zamenjuju tokove – oni ih dopunjuju
- čitači/pisači se koriste kada je potrebno preneti Unicode stringove ili karaktere – u ostalim situacijama koriste se tokovi

Čitači/pisači (readers/writers) 2/3

- omogućuju prenos karaktera iz/u:
  - datoteke (FileReader/FileWriter)
  - druge nizove karaktera (CharArrayReader/CharArrayWriter)
  - stringove (StringReader/StringWriter)
- Klase za spregu tokova i čitača/pisača – InputStreamReader, OutputStreamWriter:

```
BufferedReader in = new BufferedReader(
  new InputStreamReader(System.in));
```

FileReaderWriterTest primer04

Čitači/pisači (readers/writers) 3/3

- Metode u čitačima:
  - read() – čita jedan karakter iz toka
  - read(char[]) – čita niz karaktera
  - skip(long n) – preskače zadati broj karaktera
  - close() – zatvara čitač

Čitanje/pisanje stringova

- Koriste se klase BufferedReader i PrintWriter
- BufferedReader ima metodu readLine
- PrintWriter ima metodu println
- Primer:

```
PrintWriter out = new PrintWriter(new
   FileWriter("testReaderWriter.dat"));
out.println("Ovo je tekst");
out.close();
BufferedReader in = new BufferedReader(
new FileReader("testReaderWriter.dat"));
String s2;
while((s2 = in.readLine()) != null) {
  System.out.println(s2);
}
in.close();
```

Zaključak

- podaci se u čitaju iz ulaznih tokova, a pišu u izlazne tokove
- iz programa se retko radi direktno sa bajtovima
  - zato se tokovi ugrađuju u Filter klase koje imaju odgovarajuće metode za čitanje/pisanje
  - zato imamo tokove objekata, tokove primitivnih tipova itd.
- ako radimo sa karakterima/stringovima, koristimo čitače i pisače

Štampanje na ekran

- System.out je izlazni tok:

```
System.out.print(“Poruka”);
System.out.println(“Poruka”);
```

- Ispis se može i formatirati:

```
System.out.printf(“format”, argumenti);
System.out.printf(“%.2f %d”, (10000.0 / 3), 5);
```

Unos sa tastature

- System.in je ulazni tok:

```
BufferedReader in = new BufferedReader( new
   InputStreamReader(System.in));
String s = in.readLine();
```

- Alternativa je klasa Scanner koja ne učitava samo stringove:

```
Scanner sc = new Scanner(System.in);
String s = sc.nextLine();
int i = sc.nextInt();
sc.nextLine();
float f = sc.nextFloat();
sc.close();
```

Unos drugih primitivnih tipova sa tastature

- Koristi se wrapper klasa i njena metoda parseXxx():

```
BufferedReader in = new BufferedReader( new
  InputStreamReader(System.in));
String s = in.readLine();
int i = Integer.parseInt(s);
```

- Kraće je klasom Scanner:

```
Scanner sc = new Scanner(System.in);
int i = sc.nextInt();
sc.close();
```

Serijalizacija objekata

- serijalizacija objekta – prevođenje objekta u niz bajtova i njegova rekonstrukcuja iz niza u “živ” objekat
- serijalizovan niz bajtova se može snimiti u datoteku ili poslati preko mreže – i jedno i drugo upotrebom tokova
- prilikom serijalizacije, serijalizuju se osim samog objekta i njegovi atributi – stablo serijalizovanih objekata
- da bi se neki objekat serijalizovao:
  - potrebno je da implementira **`java.io.Serializable`** interfejs
  - da su atributi i parametri metoda takođe serijalizabilni

Rad sa arhivama

- podržan rad sa GZip i Zip formatima arhiva
- klase koje podržavaju rad sa arhivama:
  - GZipInputStream, GZipOutputStream
  - ZipInputStream, ZipOutputStream
  - ZipFile