Objektno orijentisano programiranje 1

Ulazno izlazni podsistem

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("pera.txt");
- File f = new File(".");
- File f = new File("C:\\Windows");
- File f2 = new File(f1, "pera");
- File f2 = new File(f1, "..");

- if(f.exists()) ... File.separatorChar
- if(f.isDirectory()) ...
- f.createNewfile();
- f.getCanonicalPath();
- f.listRoots();
- f.listFiles();

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

- 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, 0, read);
}
```

Tokovi (streams) 3/4

- 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

Tokovi (streams) 4/4

- 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

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

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

Čitači/pisači (readers/writers) 3/3

- 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));
```

primer04

Čitanje/pisanje stringova

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

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

Sprežne klase

- InputStreamReader i OutputStreamWriter služe za ručno sprezanje tokova i čitača/pisača
- Primer:

```
BufferedReader in = new BufferedReader(
new InputStreamReader(new FileInputStream(
"testReaderWriter.dat"), "UTF-16"));
String s2;
while((s2 = in.readLine()) != null) {
  System.out.println(s2);
}
```

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 teksta

- PrintWriter može da „gađa“ konzolu ili datoteku:

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

- Ispis se može i formatirati:

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

Formatirano štampanje

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

`%[prefiks][širina][.preciznost]tip`

- Prefiks: + ili –
  - kada je +, rezultat prikazuje i znak (i pozitivan i negativan)
  - kada je -, rezultat je poravnat na levu stranu

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.
' 'n	Štampa zadati broj cifara, sa leve popunjava razmacima.

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

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(); // posle čitanja primitivnih tipova
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();
```

Klasa Scanner

- Klasa Scanner služi za unos stringova i primitivnih tipova iz tekstualnih ulaza.
- Ona radi kao jednostavan parser teksta koji je u stanju da iz tekstualnog ulaza izdvoji stringove po nekom obrascu.
- Nakon izdvajanja stringa, u stanju je da konvertuje taj string u traženi primitivni tip.

Klasa Scanner

- Primer:

```
Scanner sc = new Scanner(System.in);
System.out.print("Unesite string:");
String s = sc.nextLine();
System.out.print("Unesite int:");
int i = sc.nextInt();
// kada čitamo primitivne tipove,
// ne uklanja se ENTER
sc.nextLine();
System.out.println(s + ", " + i);
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
- Primitivni tipovi su serijalizabilni
- Većina bibliotečkih klasa je serijalizabilna

primer05

Serijalizacija objekata

- Primer:

```
public class Automobil implements java.io.Serializable {
  public Automobil() {
    upaljen = false;
  }
  public void upali() {
    upaljen = true;
  }
  public boolean radi() {
    return upaljen;
  }
  private boolean upaljen;
}
```

Serijalizacija objekata

- U Javi postoji ključna reč ***transient*** koja se može staviti uz atribut, a ona označava da vrednost atributa ne bude preneta postupkom serijalizacije.
- Ako ovu ključnu reč stavimo uz atribut koji je primitivni tip, po rekonstrukciji objekta, u njemu će biti podrazumevana vrednost za taj tip (nula za int, na primer), a ***null*** literal za reference

Rad sa arhivama

- podržan rad sa GZip i Zip formatima arhiva
- klase koje podržavaju rad sa arhivama:
  - GZipInputStream, GZipOutputStream
  - ZipInputStream, ZipOutputStream
  - ZipFile – za pojednostavljeno čitanje i ekstrakciju zip arhiva
  - ZipEntry – reprezentuje kompresovanu datoteku u arhivi

primer06

Pravljenje arhive

```
// Kreiramo arhivu "test.zip"
ZipOutputStream out = new ZipOutputStream(
          new FileOutputStream("test.zip"));
// Pripremimo se za citanje datoteke koju cemo zapakovati.
BufferedInputStream fin = new BufferedInputStream(
          new FileInputStream("ZipTest1.java"));
// Ubacivanje datoteke u arhivu pocinje metodom putNextEntry
out.putNextEntry(new ZipEntry("ZipTest1.java"));
// Posle toga moramo da datoteku iscitamo i smestimo u arhivu.
int read;
while ((read = fin.read(buffer, 0, BUFFER_LENGTH)) != -1) {
  out.write(buffer, 0, read);
}
out.close();
fin.close();
```

Čitanje iz arhive

```
ZipInputStream in =
  new ZipInputStream(
    new FileInputStream("test.zip"));
ZipEntry zipEntry;
// Prolazimo kroz sve datoteke u arhivi.
while ((zipEntry = in.getNextEntry()) != null) {
  System.out.println("Extracting file: " +
                     zipEntry.getName());
  int read;
  while ((read = in.read(buffer, 0, BUFFER_LENGTH)) != -1) {
    String s = new String(buffer, 0, read, "UTF8");
    System.out.println(s);
  }
}
in.close();
```

Klasa ZipFile

```
ZipFile zf = new ZipFile("test.zip");
ZipEntry e = zf.getEntry("ZipTest.java");
InputStream in = zf.getInputStream(e);
int read;
byte[] buffer = new byte [BUFFER_LENGTH];
while ((read = in.read(buffer, 0, BUFFER_LENGTH)) != -1) {
  String s = new String(buffer, 0, read, "UTF8");
  System.out.println(s);
}
in.close();
zf.close();
```

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

primer07

CASE STUDY

Case Study – INI datoteke

- INI datoteke – Windows sistem za smeštanje parametara programa
- tekstualne datoteke
- struktura:
  - sekcije
    - parovi (ključ, vrednost)
  - komentari-redovi koji počinju znakom ‘;’

primer08

INI datoteke - primer

```
[General]
; naslov prozora
Title = Naslov

; x koordinata prozora
x = 100
```

Test aplikacija

```
public class TestIni {
  public TestIni() {
    INIFile ini = new INIFile("test.ini");
    String title = ini.getString("General",
   "Title","nemaaaaa");
    int x = ini.getInt("General", "x", -1);
    System.out.println("Title: " + title);
    System.out.println("x: " + x);
  }

  public static void main(String[] args) {
    new TestIni();
  }
}
```

INIFile klasa

```
import java.io.*;
import java.net.URL;
import java.util.*;

public class INIFile {

 /**  Hash mapa koja sadrzi kategorije (sekcije). Hash kljuc je
   naziv kategorije (string), a vrednost je hash mapa koja
   sadrzi parove (parametar, vrednost). */
   private HashMap<String, HashMap<String, String>> categories
   = new HashMap<String, HashMap<String, String>>();

  public INIFile(String filename) {
    BufferedReader in = null;
    try {
      in = new BufferedReader(new FileReader(filename));
      readINI(in);
      in.close();
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
```

```
 private void readINI(BufferedReader in) {
    String line, key, value; StringTokenizer st; int pos;
    String currentCategory = "default";
    HashMap<String, String> currentMap = new HashMap<String,
   String>();
    categories.put(currentCategory, currentMap);
    try {
      while ((line = in.readLine()) != null) {
        line = line.trim();
        if (line.equals("") || line.indexOf(';') == 0)
          continue;
        if (line.charAt(0) == '[') {
          currentCategory = line.substring(1, line.length()-1);
          currentMap = new HashMap<String, String>();
          categories.put(currentCategory, currentMap);
        } else {
          String[] keyValue = line.split("=");
          if (keyValue.length > 0) {
            key = keyValue[0].trim();
            value = keyValue[1].trim();
            currentMap.put(key, value);
}         }
      }
    } catch (Exception ex) {
      ex.printStackTrace();
    }
  }
```

```
/**  Vraca vrednost datog parametra u obliku stringa.
   *  @param category Kategorija (sekcija) u kojoj se nalazi parametar
   *  @param key Naziv parametra
   *  @return String koji sadrzi vrednost parametra
   */
 public String getString(String category, String key, String defaultValue) {
 HashMap<String, String> hm = categories.get(category);
 if (hm == null)
 return defaultValue;
 else {
 String value = hm.get(key);
 if (value != null)
 return value;
 else
 return defaultValue;
 }
 }
```

```
/**  Vraca vrednost datog parametra u obliku integera.
     *  @param category Kategorija (sekcija) u kojoj se nalazi
   parametar
     *  @param key Naziv parametra
     *  @return Integer vrednost parametra
     */
   public int getInt(String category, String key, int defaultValue)
    {
            HashMap<String, String> hm = categories.get(category);
            if (hm == null)
                    return defaultValue;
            else {
                    String value = hm.get(key);
                    if (value == null) {
                            return defaultValue;
                    }
                    try {
                            return (Integer.parseInt(value));
                    } catch (NumberFormatException ex) {
                            ex.printStackTrace();
                            return defaultValue;
                    }

            }
    }
```