Siniša Nikolić

Java Web Development kurs – Termin 04

Sadržaj

- Ključna reč static,
- Upoznavanje sa IO sistemom,
  - Rad sa fajl sistemom iz Jave – klasa File,
  - Osnovno tokovi,
  - Binarni tokovi
  - Tekstualni tokovi
  - Spežne klase,
  - Klasa Files,
  - Unos sa tastature – Klasa Scanner,

Dodatni materijal:

- Detaljno o tokovima,
- Serijalizacija objekta,
- Konvencija davanja imena.

Ključna reč static

- Definiše statičke atribute i metode

```
class StaticTest {
            int a = 1;
      static int i = 47;
      static void metoda() { i++; }
    }
```

- vezuju se za klasu, a ne za objekat klase
- Vrednost atibuta se na čuva u objektima, već se skladišti u *Field Data* prostoru koji se nalazi u *Class Data* (prostor namenjen za skladištenje metapodataka i informacija za klasu) koji pripada prostoru *Method Area* koji pripada delu memorije *Metaspace.*
- Statički atributi i metode postoje i bez kreiranje objekta zato im se treba pristupiti preko imena klase

```
StaticTest.i++;
```

- Statički atributi imaju istu vrednost u svim objektima
- Ako promenim statički atribut u jednom objektu, on će se promeniti i kod svih ostalih objekata

Ključna reč static

- Namena statičkih metoda:
  - pristup i rad sa statičkim atributima
  - opšte metode za koje nije potrebno da se kreira objekat
- Primeri upotrebe:

```
// out je staticki atribut
System.out;
Math.PI;
// ovo ostalo su staticke metode
Math.random();
Math.sin();
public static void main(String[] args) {...}
```

Statički blok

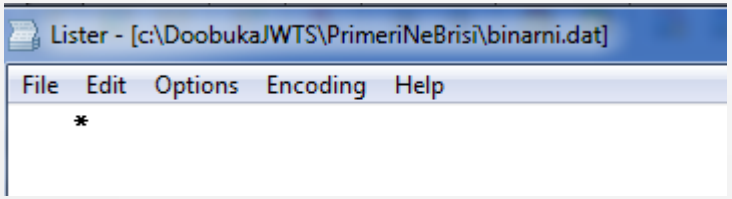
- Statički blok se izvršava samo jednom, prilikom prvog korišćenja klase
- Unutar statičkog bloka može se pristupati samo statičkim atributima i mogu se pozivati samo statičke metode

```
class Test {
  static int a;
  static int b;
         int c;
  static void f() {
    b = 6;
  }
  static {
    a = 5;
//  c = 1; //zabranjeno
    f();
  }
}
```

Primer01

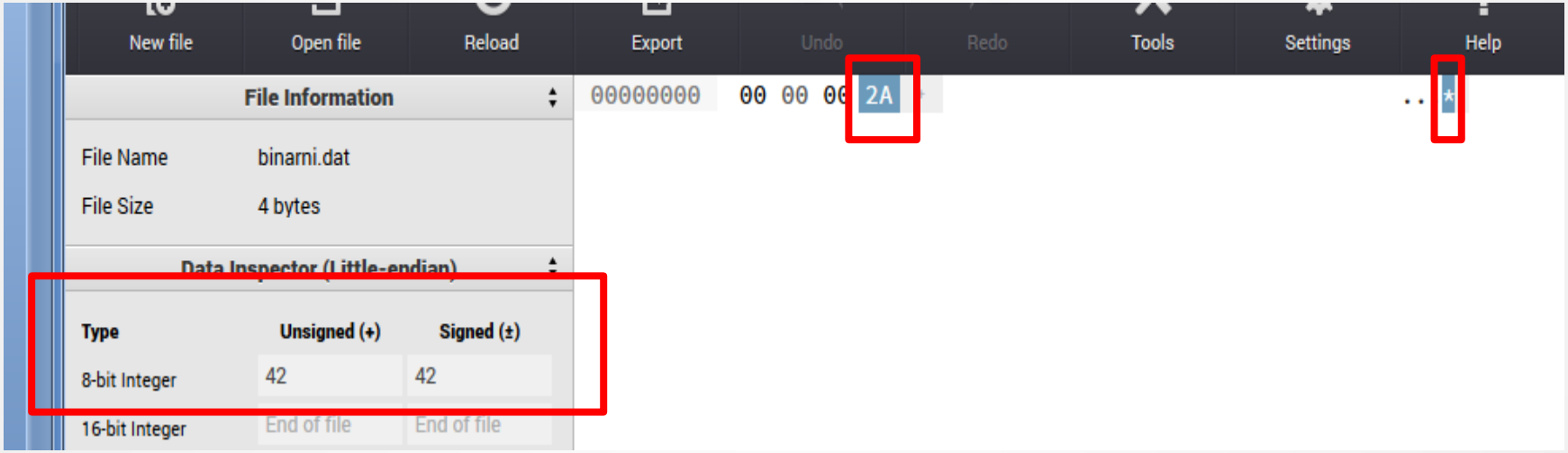
Binarni fajlovi

- Služe za čuvanje podataka u binarnom obliku.
- Binarni fajlovi su u odnosu na tekstulane brži za procesiranje i zauzimaju manje prostora.
- mana: nisu ljudski čitljivi ('human readable'), tj. ne možemo da ih otvorimo u tekstalnom editoru i razumemo njihov sadržaj (eventalno neki Heksadecimalni editor, ali i dalje nemamo garancije da ćemo razumeti sadržaj fajla)
- Primer: pisanje celog broja 42 u fajl binarni.dat
  - sadržaj 00101010, jedan bajt (= 42 u decimalnom zapisu)
  - binarni.dat otvoren u notepad programu



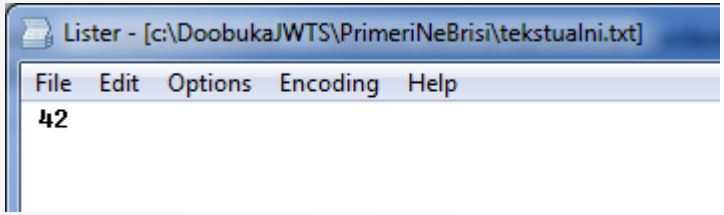
Binarni fajlovi

- Binarni fajl binarni.dat otvoren u online **hexed** programu



Tekstualni fajlovi

- Služe za čuvanje podataka u tekstualnom obliku.
- ljudski čitljivi su (‘human readable‘)
- Primer: pisanje celog broja 42 u fajl tekstualni.txt
  - sadržaj je tekst “42”



Poređenje binarnih i tekstualni fajlova

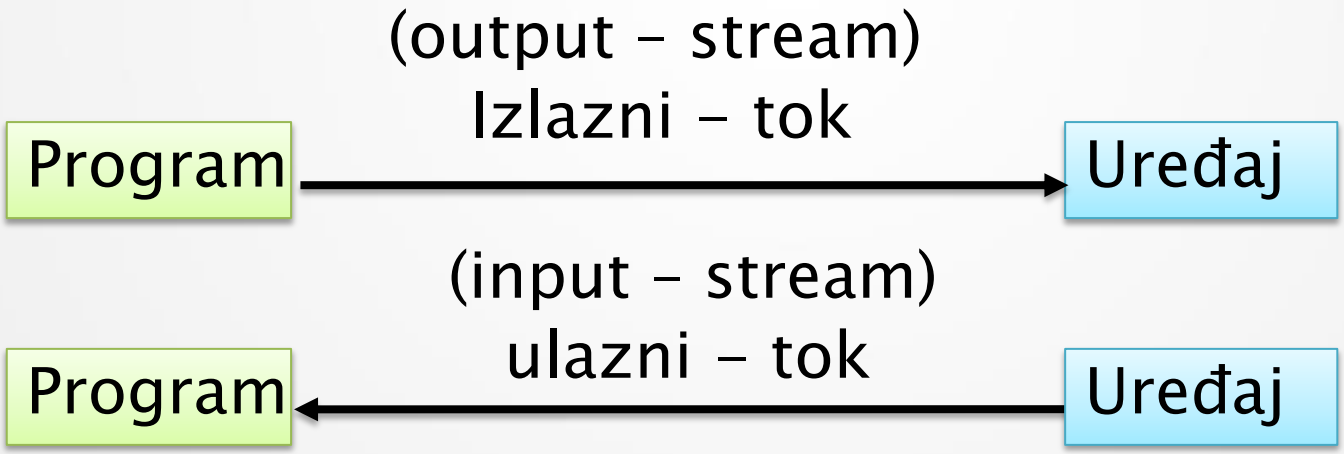
	za	protiv
Binarni	Efikasni za procesiranje i zauzimaju manje prostora	Podaci nisu ljudski čitljivi
Tekstualni	Podaci su ljudski čitljvi	Nisu efikasni za rad

Upoznavanje sa ulaz/izlaz IO sistemom

- *java.io* – Standardna biblioteka za ulazno/izlazne operacije
- Uobičajena namena: skladištenje i učitavanje podataka sa raznih uređaja npr. hard disk
- Izvorišta/odredišta:
  - memorija
  - fajl sistem
  - mrežne konekcije

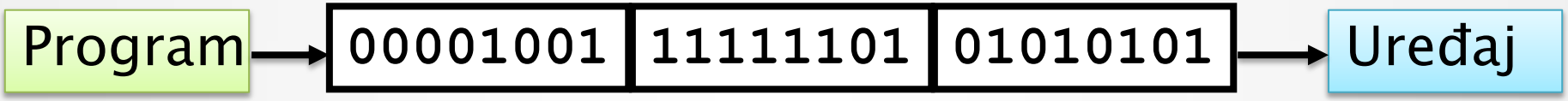
Upoznavanje sa ulaz/izlaz IO sistemom

- Prenos podataka sa izvorišta na odredište se oslanja na tokove koji prenose bajtove (*OutputStream/InputStream*) i tokove koji prenose karaktere(*Reader/Writer*)
- Nezavisno od tokova postoji i File klasa

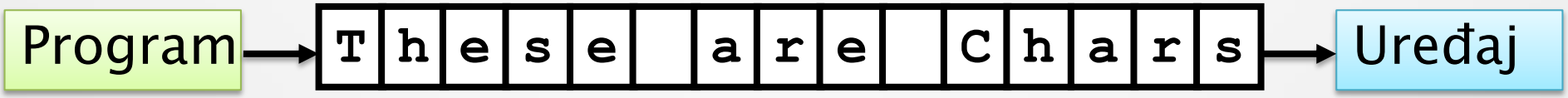


Upoznavanje sa ulaz/izlaz IO sistemom

- Postoje dve vrste tokova u Javi:
- Binarni, tokovi koji prenose bajtove – 8 bita



- Tekstualni – tokovi koji prenose karaktere



Klasa File

- Klasa File omogućuje sistemski nezavisan apstraktni pogled ka datotekama i direktorijumima.
- Služi 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
- Ne omogućuje čitanje/pisanje podataka iz datoteka!
- Sve se bazira na putanjama koje mogu biti apsolutne ili relativne.
  - Svaka putanja sadrži nazive direktorijuma koji su razdvojeni separatorom.
  - Svi nazivi u putanji osim zadnjeg su imena direktorijuma, poslednje ime može označavati direktorijum ili datoteku.

Klasa File

- Apsolutna putanja u Linux OS započinje "/" dok u Microsoft OS započinje imenom slovom čvrstog diska, iza čega sledi simbol : i separator putanje, npr. "c:\" ili "d:\").
- Relativna putanja je putanja koja se interpretira u skladu sa informacijama o nekoj drugoj putanji npr. u odnosu na trenutnu poziciju ". " .
- Primeri upotrebe:

```
File f = new  File("."); //za eklipsu bi to bio folder
   projekat u kome pokreće aplikacija
File f = new
   File("C:\\Workspace\\Termin04\\materijali");
File f = new File("
   C:\\Workspace\\Termin04\\materijali\\studenti.csv");
```

Klasa File

Funkcije

Function	Description
.exists()	proverava da li datoteka ili direktorijum na koju ukazuje putanja postoji ili ne u sistemu
.isDirectory()	određuje da li objekat klase fajl pokazuje na direktorijum, vraća true/false
.getName()	pribavlja ime datoteke ili direktorijuma na koga pokazuje objekat kalse file
.listFiles()	pribavlja sadržaj direktorijuma na koga pokazuje objekat kalse file
.delete()	briše datoteku ili direktorijum na koju ukazuje apstrakna putanja objekat kalse file
.mkdir()	kreira novi direktorijum na koji ukazuje zadata apstrakna putanja
.isFile()	određuje da li objekat klase fajl pokazuje na datoteku, vraća true/false
.createNewFile()	kreira novu praznu datoteku na koji ukazuje zadata apstrakna putanja
.getParentFile()	pribavlja direktorijuma (ukoliko postoji) koji sadrži posamtrani diretorijum ili datoteku
.getAbsolutePath()	vraća apsolutnu putanju
.getPath()	vraća relativnu putanju

Primer02-01

Binarni tokovi (Input Stream/OutputStream)

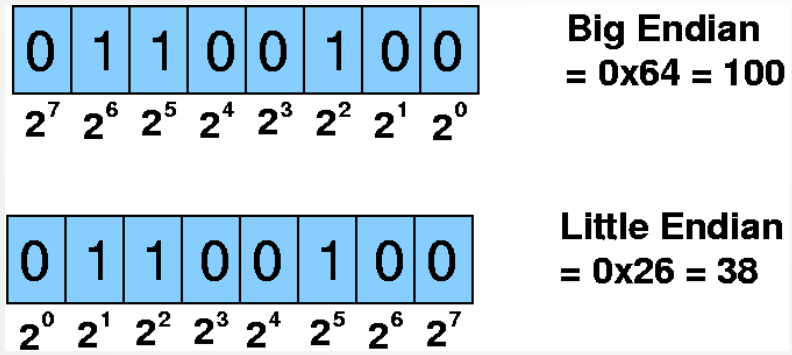
- Osmišljeni kao mehanizam koji omogućuje unificiran pristup podacima
- Bazirani na bajtovima
  - prenos jednog bajta
  - prenos niza bajtova
- 2 osnovne klase
  - ***Input Stream*** – *byte-ulaz*
  - ***Output Stream*** – *byte-izlaz*

Binarni tokovi (Input Stream/OutputStream)

- Apstraktna klasa ***Input Stream*** *čije naslednice* omogućuju čitanje bajtova sa izvora
  - System.in – primer ulaznog toka podatka sa tastature
- Apstraktna klasa ***Output Stream*** *čije naslednice* omogućuju pisanje bajtova na odredište
  - System.out – primer izlaznog toka podatka na ekran

Tekstualni tokovi (Reader/Writer)

- Bazirani na karakterima
  - prenos jednog karaktera
  - prenos niza karaktera
- Ispravljaju problem sa binarnim tokovima – slabu podršku *Unicode* rasporedu:
  - tokovi ne prenose dobro *Unicode* stringove – ćirilična slova su problem
  - poseban problem predstavljaju različite hardverske platforme (*little-endian*, *big-endian*)



Tekstualni tokovi (Reader/Writer)

- Tekstualni tokovi ne zamenjuju binarne tokove – oni ih dopunjuju
- 2 osnovne klase
  - ***Reader*** – *char-ulaz*
  - ***Writer*** – *char-izlaz*
- Čitači/pisači se koriste kada je potrebno preneti Unicode stringove ili karaktere – u ostalim situacijama koriste se tokovi
- Metode u čitačima:
  - read() – čita jedan karakter sa ulaza
  - read(char[] cbuf) – čita sa ulaza i smešta karaktere u niz
  - read(char[] cbuf,int offset,int length) – čita sa ulaza i smešta karaktere u određeni deo niza
  - skip(long n) – preskače zadati broj karaktera sa ulaza
  - close() – zatvara čitač

Tekstualni tokovi (Reader/Writer)

- Metode u pisačima:
  - write(char c) – piše jedan karakter na izlaz
  - write(char[] cbuf) – piše niz karakter na izlaz
  - write(char[] cbuf, int off, int len) – piše deo niza karakter na izlaz
  - flush() – pražnjenje bafera od čitača i slanje podataka
  - close() – zatvara pisač

- Za datoteke (FileReader/FileWriter)
- Za druge nizove karaktera (CharArrayReader/CharArrayWriter)
- Za stringove (StringReader/StringWriter)

Tekstualni tokovi (Reader/Writer)

- Uopšten rad sa tokovima operacije čitanje/pisanje:
  - otvori ulazni/izlazni tok
  - dok ima još informacija (while...)
    čitaj podatke iz toka/piši podatke u tok
  - zatvori tok.
- U Javi sa fajlovima:
  - Kreiraj objekat toka i poveži ga npr. sa fajlom na disku
    - dodeli toku željenu dodatnu funkcionalnost (npr. baferisanje)
  - dok ima još informacija
    čitaj podatke iz toka/piši podatke u tok
  - zatvori tok.

Tekstualni tokovi (Reader/Writer)

- Za čitanje i pisanje String objekata se Koriste klase *BufferedReader* i *PrintWriter* oko *FileReader*-a i *FileWriter*-a
  - BufferedReader ima metodu readLine
  - PrintWriter ima metodu println
- Primer čitanja reda:

```
BufferedReader in = new BufferedReader(new
   FileReader(new File("testReader.dat")));
String s2;

while((s2 = in.readLine()) != null) {
  System.out.println(s2);
}
in.close();
```

Tekstualni tokovi (Reader/Writer)

- Primer pisanja reda:

```
PrintWriter out = new PrintWriter(new FileWriter(new
   File("testWriter.dat")));
out.println("Tekst poruke koja se upisuje u fajl");
out.println("Red 2");
out.println("Red 3");
out.flush();
out.close();
```

Sprežne klase i Unos teksta sa tastature

- Klase koje se mogu koristiti sa povezivanje čitača/pisača sa tokovima, nazivaju se *sprežne klase*
- *InputStreamReader* i *OutputStreamWriter* služe za ručno sprezanje tokova i čitača/pisača
- Koristi se wrapper klasa i njena metoda parseXxx()
- Primer čitanje teksta sa tastature:

```
BufferedReader in = new BufferedReader(
new InputStreamReader(System.in);
String s2;
while((s2 = in.readLine()) != null) {
  System.out.println(s2);
  //parsiranje teksta u ceo broj
  int a = Integer.parseInt(s2)
}
```

Sprežne klase i Ispis teksta sa ekran

- Primer pisanja teksta sa ekran:

```
PrintWriter out = new PrintWriter(new
   OutputStreamWriter(System.out, "UTF8"));
out.println("Tekst poruke koja se ispisuje na ekran");
out.println("Tekst 2");
out.println("Tekst 3");
out.flush();
```

Primer02-02

Klasa Files

- Od Jave 1.7 moguće je odjednom učitati sve linije iz fajla oslanjajući se na klasu *Files* i njenu metodu *readAllLines.*
- Metoda je namenjena jednostavnim scenarijama, gde je zgodno čitati sve redove u jednoj operaciji.
- Metoda nije namenjena za velike fajlove koje sadrže desetine hiljada redova.
- Metoda *write* klase *Files* mogućuje upis liste Stringova u fajl. Pisanje je moguće raditi tako što će se proširiti sadržaj fajla ili izmeniti ceo fajl.

Klasa Files

```
//čitanje svih redova
List<String> lines =
    Files.readAllLines(Paths.get("rezultati.csv"),
    Charset.forName("UTF-8"));
for (String linijaIzFajla : lines) {
    System.out.println(linijaIzFajla);
}
//pisanje svih redova
Files.write(Paths.get("rezultati.csv"), lines,
    Charset.forName("UTF-8"), StandardOpenOption.WRITE);
```

Unos sa tastature – Klasa Scanner

- Korišćenja readera i sprežne klase nad tokom podataka sa tastature obezbeđuje samo pribavljanje String-ova.
- Alternativa je klasa Scanner koja ne učitava samo stringove, već i ostale primitivne tipove podataka (cele i relane brojeve, logička vrednost,..)
- Klasa Scanner služi za unos stringova i primitivnih tipova iz tekstualnih ulaza, 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:

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

Unos sa tastature – 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 citamo primitivne tipove,
// ne uklanja se ENTER
// ne prazni ostatak testa sa ulaza
sc.nextLine();
System.out.println(s + ", " + i);
sc.close();
```

Primer03 Primer04

Podsetnik

- 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
- Postoje posebne klase za rad sa tastaturom i ekranom
- Nezavisno od ovog postoji i klasa java.io.File za koja nam omogućava osnovne operacije nad fajl sistemom (kreiranje fajla, provera da li postoji fajl, itd.)

Primer kompleksnog zadatak

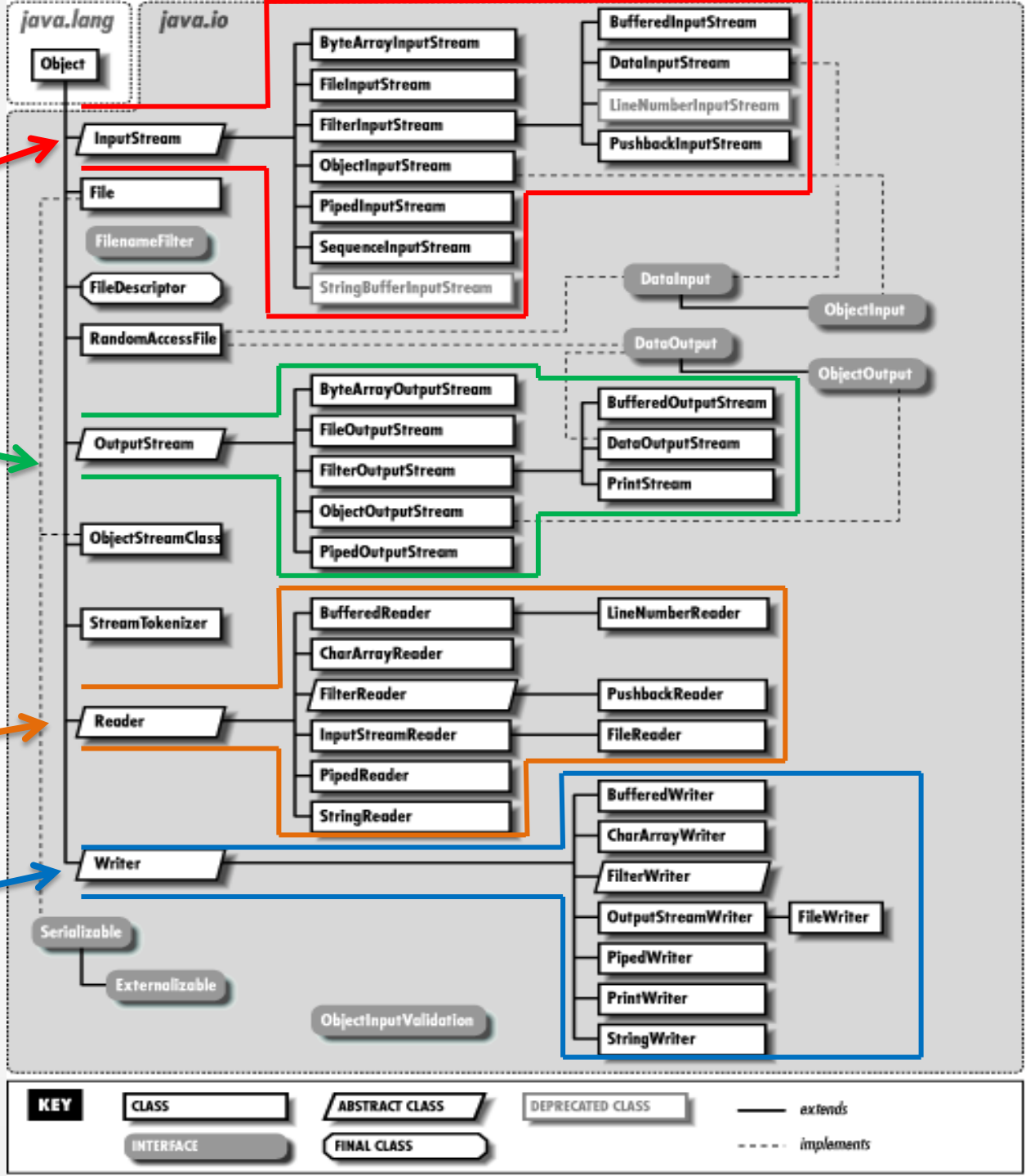
- Primer studentske službe koja vodi evidenciju o studentima. Podaci o studentima se perzistiraju u fajl sistemu. Aplikacija koristi UI (User Interface) klasu StudentUI u kojoj se podaci o studentima čuvaju u listi i koja sadrži metode za rad sa studentima. Korisnicima je omogućen: pregled, dodavanje, izmena, brisanje podataka o studentima.

Primer05 Zadatak01

Dodatni Materijal

IO paket

- Binarni tokovi koji prenose bajtove – 8 bita
- Tekstualni tokovi koji prenose karaktere

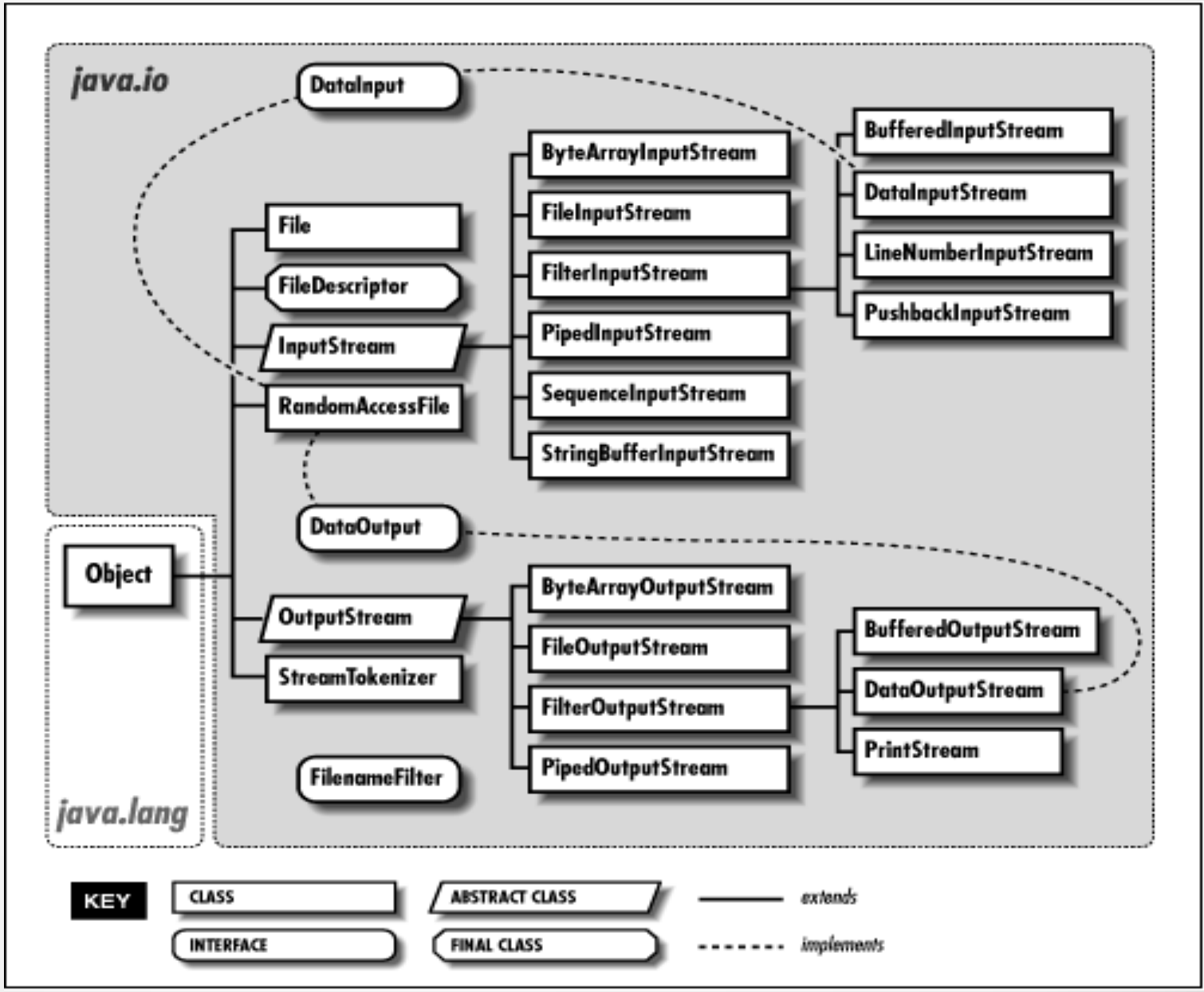


Binarni tokovi (Input Stream/OutputStream)-dodatno

- Omogućuju prenos podataka na različita izvorišta/odredišta:
  - datoteke (*FileInputStream*, *FileOutputStream*)
  - niza bajtova (*ByteArrayInputStream*, *ByteArrayOutputStream*)
  - objekata (*ObjectInputStream*, *ObjectOutputStream*)
  - itd
- Isti kod se koristi za čitanje/pisanje iz, na primer, datoteke ili mrežne konekcije

Binarni tokovi (Input Stream/OutputStream)

- Podela



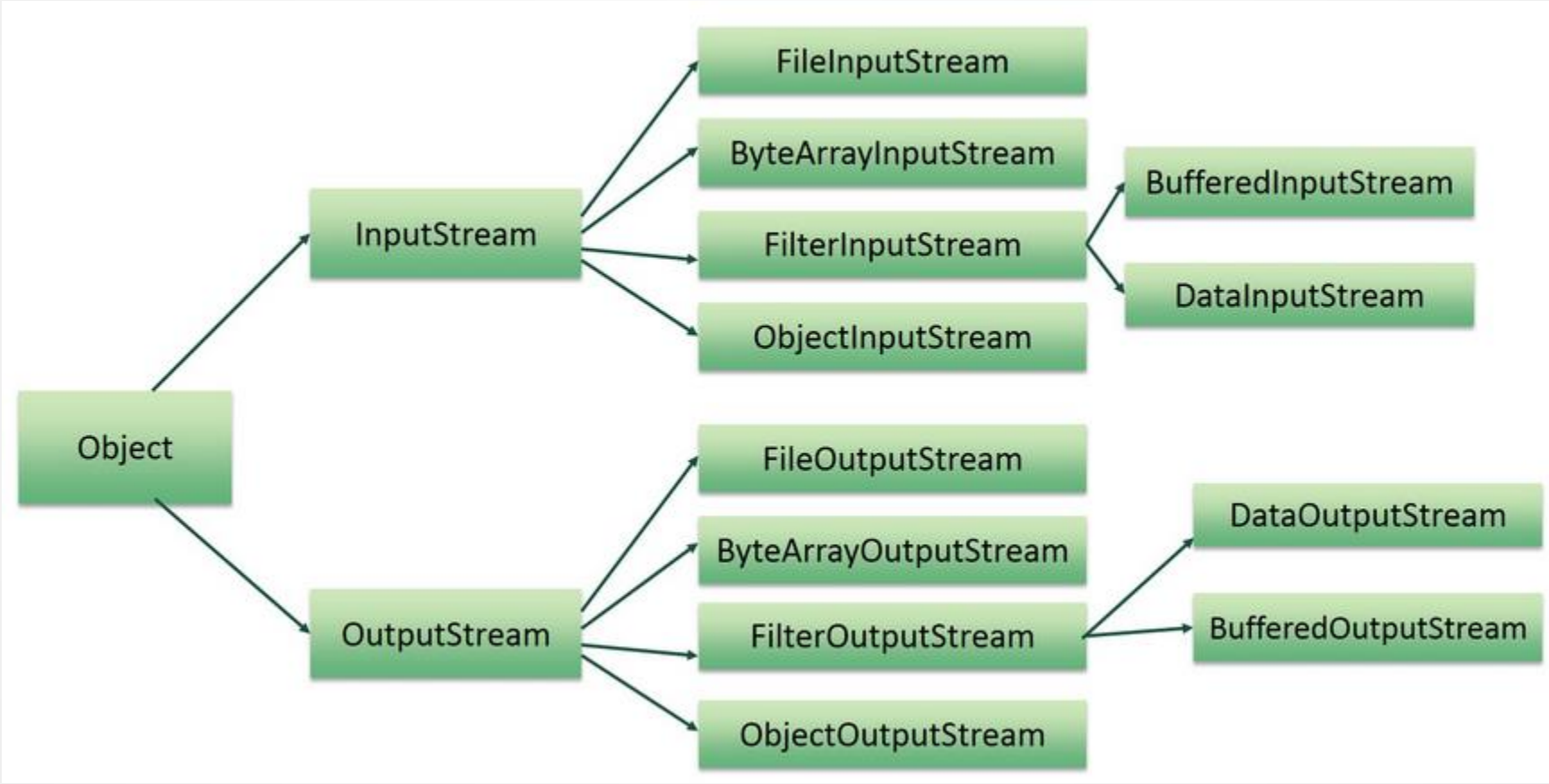
Binarni tokovi (Input Stream/OutputStream)

- Metode u tokovima za čitanje sa izvorišta:
  - 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
- Primer upotrebe – kopiranje sadržaja datoteke:

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

Tokovi (Streams) – dodatno

- Podela



Binarni tokovi (Input Stream/OutputStream)

- Koncept filtera – donose dodatnu funkcionalnost tokovima:
  - prenos primitivnih tipova na mašinski nezavisan način (*DataInputStream*, *DataOutputStream*)
  - baferizovan prenos podataka (*BufferedInputStream*, *BufferedOutputStream*)
- Primer kreiranja toka uz upotrebu filtera

```
DataOutputStream out =
      new DataOutputStream(
        new BufferedOutputStream(
                new FileOutputStream(new
   File("testStream.dat"))));
```

Tekstualni tokovi (Reader/Writer)-Dodatno

- FileWriter/FileReader
  - Ograničen skup metoda za rad (moguće je samo pisati/čitati karaktere/stringove)
  - Nije efikasna (za pisanje/čitanje svakog karaktera vrši se zaseban pristup disku – jako sporo)

Method Summary

abstract void	**close**() Close the stream, flushing it first.
abstract void	**flush**() Flush the stream.
void	**write**(char[] cbuf) Write an array of characters.
abstract void	**write**(char[] cbuf, int off, int len) Write a portion of an array of characters.
void	**write**(int c) Write a single character.
void	**write**(String str) Write a string.
void	**write**(String str, int off, int len) Write a portion of a string.

Method Summary

abstract void	**close**() Close the stream.
void	**mark**(int readAheadLimit) Mark the present position in the stream.
boolean	**markSupported**() Tell whether this stream supports the mark() operation.
int	**read**() Read a single character.
int	**read**(char[] cbuf) Read characters into an array.
abstract int	**read**(char[] cbuf, int off, int len) Read characters into a portion of an array.
boolean	**ready**() Tell whether this stream is ready to be read.
void	**reset**() Reset the stream.
long	**skip**(long n) Skip characters.

Tekstualni tokovi (Reader/Writer)

- Rešenje bi bilo umotati FileWriter u klasu BufferedWriter/PrintWriter, a FileReader u klasu BufferedReader
- BufferedWriter
  - Vrši baferisanje izlaza FileWriter-a, tj. više karaktera se čuva u memoriji pa se onda odjednom zapisuju u fajl – efikasno.
- BufferedReader
  - Vrši baferisanje ulaz FileReader-a, tj. više karaktera se čita odjednom iz fajla u memoriju pa se onda deo njih preuzima iz memorije- efikasno.
- PrintWriter
  - Sadrži u sebi BufferedWriter
  - Pruža korisne metode za pisanje formatiranih podataka, npr. println(), format(...), println(int x), println(long x)

Tekstualni tokovi (Reader/Writer)

- Detekcija kraja fajla EOF – EndOfFile
  - Obično unapred ne znamo koliko podataka ima u fajlu.
  - Metode za čitanje podataka vraćaju ‘nemoguću‘ (posebno odabranu) vrednost ako su naišle na kraj fajla
    - FileReader.read vraća -1
    - BufferedReader.readLine() vraća ‘null‘
  - Tipičan segment koda za detekciju EOF:

```
while ((c = myReader.read()) != -1){
 //...obrada c
}
```

Serijalizacija podataka

- 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 (i primitivni tipovi i drgi objekti) pa se javlja 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

Serijalizacija objekta

```
class RacunUBanci implements java.io.Serializable {
    int    sifraRacuna;
    double stanje;
    Klijent k;
    boolean skiniNovac(double zaPodizanje){
        if(stanje-zaPodizanje < 0){
            System.out.println("Nedovoljan saldo");
            return false;
        }
        stanje-=zaPodizanje;
        return true;
    }
    void uplatiNovac(double zaUplatu){
        stanje+=zaUplatu;
    }
}
class Klijent implements java.io.Serializable {
    String JMBG;
    String imeIPrezime;
    String adresa;
}
```

com.ftninformatika.vezbe04.primerDodatnoSerijalizacija

Serijalizacija podataka

- Kada se objekat može serijalizovati u niz bajtova moguće ga je upisati u fajl, a kasnije i očitati iz fajla korišćenjem klasa *ObjectInputStream* i *ObjectOutputStream*.
- 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

Konvencija davanja imena

- Nazivi klasa pišu se malim slovima, ali sa početnim velikim slovom (npr. Automobil, ArrayList).
- Ukoliko se naziv klase sastoji iz više reči, reči se spajaju i svaka od njih počinje velikim slovom (npr. HashMap).
- Nazivi metoda i atributa pišu se malim slovima (npr. size, width). Ako se sastoje od više reči, one se spajaju, pri čemu sve reči počevši od druge počinju velikim slovom (npr. setSize, handleMessage).
- Nazivi paketa pišu se isključivo malim slovima. Ukoliko se sastoje iz više reči, reči se spajaju (npr. mojpaket, velikipaket.malipaket).
- Detaljan opis konvencija nalazi se na adresi http://www.oracle.com/technetwork/java/codeconv-138413.html.