Java tečaj, 6.dio

Datoteke. Tokovi.

AG 2018./2019.

Marko Čupić

Pregled

- Potpora za rad s datotekama te općenitije ulazno/izlaznim API-jem nalazi se u dva paketa
  - Paket java.io
  - Paket java.nio (od Jave 1.4)
- Od Jave 7 paket java.nio dobio je niz novih funkcionalnosti i proširenja

Datotečni sustav

- Temeljni razred: File (java.io)
  - Apstraktna reprezentacija bilo kojeg objekta datotečnog sustava (direktorija, datoteke)
  - Metode za dohvat informacija o tim objektima
  - Važno: stvaranje ovog objekta ne povlači i stvaranje datoteke/direktorija na disku!
    - Primjerak ovog razreda tretirati isključivo kao **stazu** u memoriji

Datotečni sustav: File

- Informacije o platformi
  - `File.separator: String`
  - `File.separatorChar: char`
  - `File.pathSeparator: String`
  - `File.pathSeparatorChar: char`
  - `File[] roots = File.listRoots();`

Datotečni sustav: File

```
public static void main(String[] args) {

 File direktorij = new File("d:/tmp/javaPrimjeri");
 File datoteka1 = new File(direktorij, "readme.txt");
 File datoteka2 = new File("d:/tmp/javaPrimjeri/readme.txt");

 File roditelj = datoteka1.getParentFile();
 boolean postoji = datoteka2.exists();
 boolean citljivo = datoteka2.canRead();
 boolean pisljivo = datoteka2.canWrite();
 boolean izvrsivo = datoteka2.canExecute();
 long velicina = datoteka2.length();
 boolean jeDatoteka = datoteka2.isFile();
 boolean jeDirektorij = datoteka2.isDirectory();
 boolean jeSkrivena = datoteka2.isHidden();
}
```

Datotečni sustav: File

- Listanje direktorija:

```
File direktorij = new File("d:/tmp/javaPrimjeri");
File[] djeca = direktorij.listFiles();

for(File file : djeca) {
  System.out.println(file.getName());
}
```

Datotečni sustav: File

- Još nekoliko drugih metoda
  - Statičke metode za stvaranje privremenih datoteka (ime datoteke nam nije bitno)
  - Atomičko stvaranje datoteke samo ako takva već ne postoji
  - Preimenovanje datoteke, brisanje
  - Stvaranje direktorija / poddirektorija
  - Apsolutne staze / kanonske staze
  - Informacije o particiji, ...

Primjeri uz File

- Sada ćemo riješiti nekoliko zadataka vezanih uz uporabu razreda `File`

(zadatci)

Datotečni sustav: Files i Path

- Od Jave 7 napravljeno je poprilično unaprijeđenje
  - Apstraktna staza do objekata datotečnog sustava predstavljena je razredom `Path` (paket `java.nio.file`)
  - Razred `Paths` nudi metode za stvaranje objekata tipa `Path`
  - Razred `Files` nudi statičke metode za dohvat informacija o `Path` objektima

http://docs.oracle.com/javase/tutorial/essential/io/path.html

Datotečni sustav: Files i Path

```
Path p =
    Paths.get("d:/tmp/javaPrimjeri/readme.txt");
System.out.println("Postoji: "+Files.exists(p));
System.out.println("Veličina: "+Files.size(p));

Path p2 = Paths.get("dir1/dir2");
Path p2a = p2.resolve("datoteka1.txt");
Path p2b = p2.resolve("../datoteka1.txt");

Path p3 = p2b.toAbsolutePath();
Path p3b = p3.normalize();
```

http://docs.oracle.com/javase/tutorial/essential/io/path.html

Datotečni sustav: Files i Path

- Razred `Files` nudi niz korisnih metoda:
  - Kopiranje datoteka
  - Stvaranje direktorija, datoteka i simboličkih linkova
  - Seljenje i brisanje
  - Obilazak strukture poddirektorija uporabom oblikovnog obrasca *Visitor*
  - ...

https://docs.oracle.com/javase/9/docs/api/java/nio/file/Files.html

Datotečni sustav: Files i Path

```
Path p = …;
System.out.println(Files.exists(p));
System.out.println(Files.isReadable(p));
System.out.println(Files.isWritable(p));
System.out.println(Files.isDirectory(p));

try (
  DirectoryStream<Path> stream =
     Files.newDirectoryStream(p)) {

        for (Path entry: stream) {
            …
        }
}
```

https://docs.oracle.com/javase/9/docs/api/java/nio/file/Files.html

Datotečni sustav: Files

- U razredu `Files` postoje i metode koje vraćaju prave tokove:

```
Stream<Path> find(Path start, int
maxDepth, BiPredicate<Path,
BasicFileAttributes> matcher,
FileVisitOption...options)

Stream<Path> list(Path dir)

Stream<Path> walk(Path start, … )
```

Datotečni sustav: File <=> Path

- Moguće je preslikavanje u oba smjera
  - Primjerci razreda `File` imaju metodu `toPath()`

```
File f = new File("...");
Path p = f.toPath();
```

  - Primjerci razreda `Path` imaju metodu `toFile()`

```
Path p = Paths.get("...");
File f = p.toFile();
```

http://docs.oracle.com/javase/7/docs/api/java/nio/file/Path.html

Datotečni sustav: Files

- Prikladno rješenje za opisani problem predstavlja oblikovni obrazac *Posjetitelj* (engl. *Visitor*)
- Posao koji je potrebno obaviti modeliran je zasebnim sučeljem `FileVisitor`
- Razred `Files` nudi porodicu statičkih metoda `walkFileTree(path, visitor)` koje obilaze čvorove podstabla zadane staze i za svaki čvor pozivaju metode posjetitelja

Datotečni sustav: FileVisitor

```
public interface FileVisitor<T> {
    FileVisitResult preVisitDirectory(
            T dir, BasicFileAttributes attrs
    ) throws IOException;
    FileVisitResult visitFile(
            T file, BasicFileAttributes attrs
    ) throws IOException;
    FileVisitResult visitFileFailed(
            T file, IOException exc
    ) throws IOException;
    FileVisitResult postVisitDirectory(
            T dir, IOException exc
    ) throws IOException;
}
```

Datotečni sustav: FileVisitor

```
public enum FileVisitResult {

    /* nastavi na elemente trenutnog direktorija */
    CONTINUE,
    /* prekini obilazak direktorija */
    TERMINATE,
    /* preskoči trenutni direktorij */
    SKIP_SUBTREE,
    /* preskoči braću trenutnog direktorija */
    SKIP_SIBLINGS;

}
```

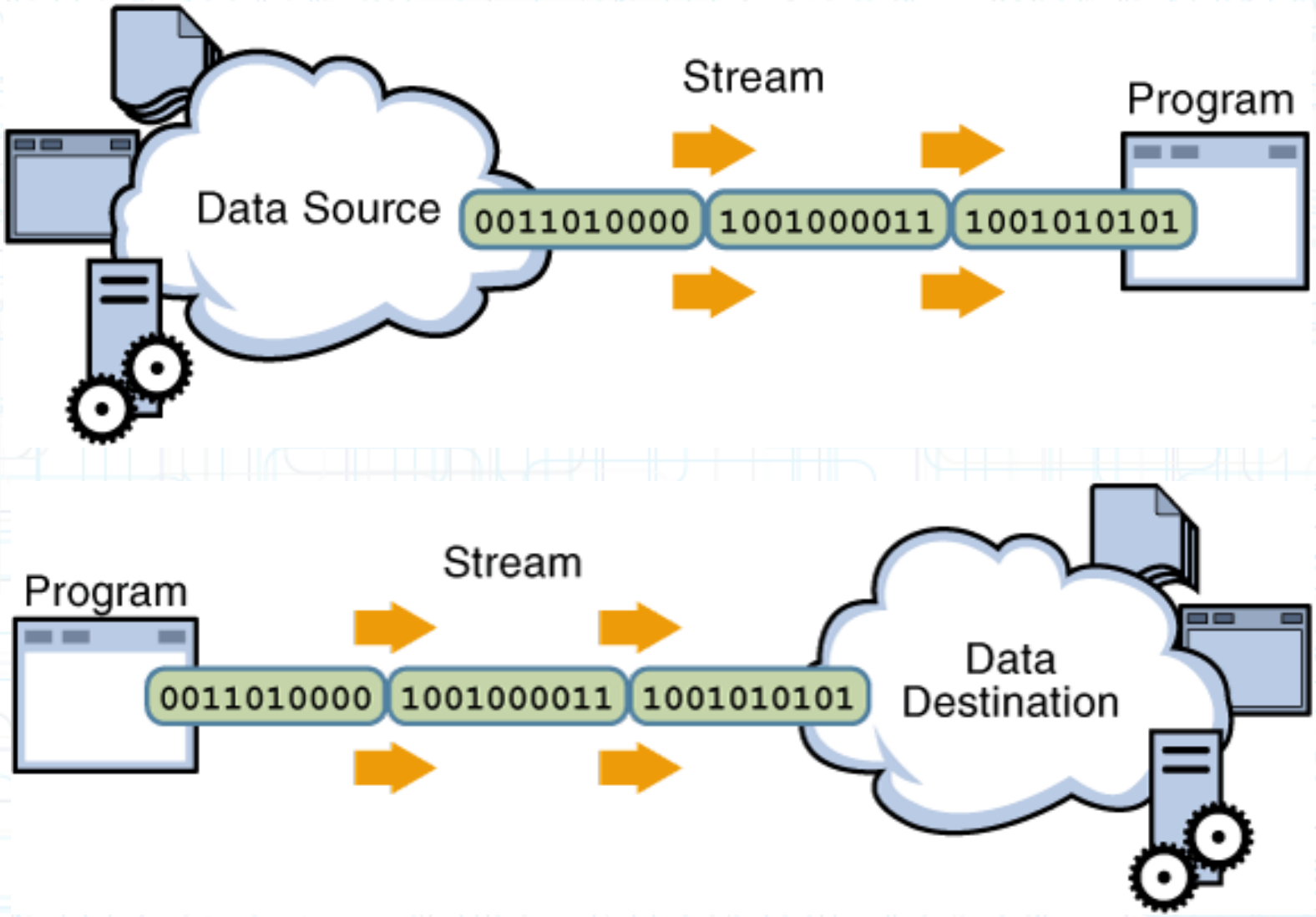
Datotečni sustav: Files

- Napišite program koji će oslanjajući se na `FileVisitor` ispisati podstablo zadanog direktorija.

I/O tokovi

- Umjesto modeliranja rada s datotekama, krećemo od općenitijeg modela: tok podataka
  - Tokovi su jednosmjerni
  - Ulazni tok: ponaša se kao izvor podataka; klijent može čitati podatak po podatak (ili više njih slijedno u spremnik)
  - Izlazni tok: ponaša se kao ponor podataka; klijent u njega može samo zapisivati

I/O tokovi


Stream
Program
Data Source
0011010000
1001000011
1001010101
Program
Stream
0011010000
1001000011
1001010101
Data
Destination

I/O tokovi

- Paket `java.io` podržava dvije vrste tokova podataka:
  - **Tokovi okteta**: podatci s kojima radimo su okteti (`byte`); prikladno za rad s binarnim podatcima
  - **Tokovi znakova**: podatci s kojima radimo su znakovi (`char`); prikladno za rad s tekstovnim podatcima
  - Prisjetimo se: u Javi okteti ≠ znakovi

Tokovi okteta

- Izvor i ponor okteta modelirani su apstraktnim razredima:
  - `InputStream`
  - `OutputStream`
- `InputStream` nudi metode za čitanje okteta odnosno polja okteta.
- `OutputStream` nudi metode za pisanje okteta odnosno polja okteta.

Tokovi okteta

```
public abstract class InputStream implements Closeable {
    public abstract int read() throws IOException;
    public int read(byte b[]) throws IOException {…}
    public int read(byte b[], int off, int len)
                                        throws IOException {…}
    public long skip(long n) throws IOException {…}
    public int available() throws IOException {
        return 0;
    }
    public void close() throws IOException {}
    public synchronized void mark(int readlimit) {}
    public synchronized void reset() throws IOException {
        throw new IOException("mark/reset not supported");
    }
    public boolean markSupported() {
        return false;
    }
}
```

Tokovi okteta

```
public abstract class OutputStream
                      implements Closeable, Flushable {

    public abstract void write(int b) throws IOException;
    public void write(byte b[]) throws IOException {…}
    public void write(byte b[], int off, int len)
                                  throws IOException {…}
    public void flush() throws IOException {
    }
    public void close() throws IOException {
    }

}
```

Tokovi okteta

```
public interface Closeable extends AutoCloseable {
    public void close() throws IOException;
}


public interface Flushable {
    void flush() throws IOException;
}
```

Tokovi okteta

- Zašto apstraktno modeliranje? Izvor i ponor može biti bilo što:
  - Datoteka na disku računala
  - TCP priključna točka s kojom preko mreže razgovaramo s drugom aplikacijom
  - Potprogram koji na zahtjev generira tražene podatke (npr. `InputStream` koji vraća slučajne brojeve)
  - ...

Tokovi okteta

- Neke konkretne implementacije:
  - `FileInputStream`, `FileOutputStream` (čitanje i pisanje u datoteku)
  - `ByteArrayInputStream`, `ByteArrayOutputStream` (čitanje i pisanje iz/u zadani spremnik)

Tokovi okteta

```
private static void primjer2() {
  Path p = Paths.get("d:/tmp/javaPrimjeri/readme.bin");
  InputStream is = null;
  try {
    is = new Files.newInputStream(p);
    byte[] buff = new byte[1024];
    while(true) {
      int r = is.read(buff);
      if(r<1) break;
      // obradi samo buff[0] do buff[r-1]
    }
  } catch(IOException ex) {
    // Obradi pogrešku
  } finally {
    if(is!=null) {
      try { is.close(); } catch(IOException ignorable) {}
    }
  }
}
```

Tokovi okteta

```
private static void primjer4() {
  Path p = Paths.get("d:/tmp/javaPrimjeri/readme.bin");

  try (InputStream is = Files.newInputStream(p)) {
    byte[] buff = new byte[1024];
    while(true) {
      int r = is.read(buff);
      if(r<1) break;
      // obradi samo buff[0] do buff[r-1]
    }
  } catch(IOException ex) {
    // Obradi pogrešku
  }

}
```

Resursi se u ovom slučaju zatvaraju automatski, odmah po izlasku iz `try` bloka, **prije** ulaska u eventualne `catch` i `finally` dijelove.

Tokovi okteta

- Neovisno o konkretnom izvoru/ponoru okteta, iste možemo dekorirati:
  - Bufferirano čitanje/pisanje
  - Pushback izvor podataka
  - Čitanje serijaliziranih objekata
  - Kriptiranje/dekriptiranje u letu
  - Kompresija/dekompresija u letu
  - ...

Oblikovni obrazac Dekorator

- Npr.:

```
Path p = …;
FileOutputStream fos = Files.newInputStream(p);
OutputStream os = new BufferedOutputStream(fos);

os.write(2);
os.write(new byte[] {1,2,3});
os.write(2);
os.close();
```

Oblikovni obrazac Dekorator

- Npr.:

```
Path p = …;
FileOutputStream fos = Files.newInputStream(p);
OutputStream os = new ZipOutputStream(
            new BufferedOutputStream(
              fos
            ));
os.write(2);
os.write(new byte[] {1,2,3});
os.write(2);
os.close();
```

- Klijentski kod može po volji kombinirati koje će dekoratore koristiti

Dekoratori tokova okteta

- U `java.io` već imamo niz dekoratora:
  - `BufferedInputStream`
  - `DataInputStream`
  - `ObjectInputStream`
  - `PushbackInputStream`
  - `SequenceInputStream`
- Slično je i s `OutputStream` dekoratorima

Znakovni tokovi

- Već smo spomenuli da je u Javi okteti ≠ znakovi
- Da bismo znali kako znakove kodirati u oktete, trebamo znati koju ćemo kodnu stranicu koristiti
- Kodna stranica je u Javi modelirana razredom `Charset` (paket `java.nio.charset`)

Znakovni tokovi

- Na svim Java platformama automatski su podržane i dostupne sljedeće kodne stranice:
  - `US-ASCII`, `ISO-8859-1`
  - `UTF-8`, `UTF-16BE`, `UTF-16LE`, `UTF-16`
- Razred `StandardCharset` omogućava dohvat svih tih kodnih stranica

```
Charset c = StandardCharsets.UTF_8;
```

Znakovni tokovi

- Alternativno, ako znamo ime kodne stranice, možemo koristiti i poziv:

```
Charset c2 = Charset.forName("ISO-8859-2");
```

- Tako možemo doći i do nestandardno podržanih kodnih stranica (ako su instalirane); inače iznimka

Znakovni tokovi

- Jednom kad imamo kodnu stranicu, konverzija okteta u znakove ide ovako:

```
Charset c = StandardCharsets.UTF_8;
Charset c2 = Charset.forName("ISO-8859-1");

byte[] okteti = new byte[] {97,99,104};
String tekst = new String(okteti, c);

byte[] drugoKodiranje = tekst.getBytes(c2);
```

Znakovni tokovi

- Znakovni tokovi unutar paketa `java.io` modelirani su sljedećim apstraktnim razredima:
  - Izvori znakova: razred `Reader`
  - Ponori znakova: razred `Writer`
- Sučelja su slična razredima `InputStream` i `OutputStream` samo što umjesto okteta i polja okteta primaju znakove i polja znakova

Znakovni tokovi

- Postoji nekoliko konkretnih implementacija
  - `FileReader` i `FileWriter`
    (koriste pretpostavljenu kodnu stranicu!)
  - `StringReader` i `StringWriter`
  - `CharArrayReader` i `CharArrayWriter`

Znakovni tokovi

- Postoji nekoliko dekoratora:
  - `BufferedReader`, `BufferedWriter`
  - `LineNumberReader`
  - `PushbackReader`
- `BufferedReader` praktičan, nudi:
  - `String readLine();`
  - `Stream<String> lines();`

Znakovni tokovi

- Konačno, postoji most između znakovnih tokova i tokova okteta
  - `InputStreamReader`:
    to je reader koji oktete čita iz toka okteta na koji je spojen, oktete dekodira uporabom zadane kodne stranice i time generira znakove
  - `OutputStreamWriter`:
    to je writer koji iz znakova generira oktete temeljem zadane kodne stranice

Znakovni tokovi

- Često korišteni idiom za rad s tekstovnim datoteka

```
BufferedReader br = new BufferedReader(
      new InputStreamReader(
        new BufferedInputStream(
          Files.newInputStream(p)),"UTF-8"));

String redak = br.readLine();

Writer bw = new BufferedWriter(
       new OutputStreamWriter(
         new BufferedOutputStream(
          Files.newOutputStream(p)),"UTF-8"));

bw.write(redak);
```

Znakovni tokovi

- Od Jave 8, prethodni primjer zamjenjujemo uporabom novog API-ja:

```
BufferedReader br = Files.newBufferedReader(path, charset);
String redak = br.readLine();

BufferedWriter bw = Files.newBufferedWriter(path, charset);
bw.write(redak);
```

Još korisne metode:

```
Stream<String> lines(Path path, Charset cs)
Stream<Path> list(Path dir)
List<String> readAllLines(Path path, Charset cs)
byte[] readAllBytes(Path path)
Path write(Path path, byte[] bytes, OpenOption... options)
Path write(Path path, Iterable<? extends CharSequence>
lines, Charset cs, OpenOption... options)
Stream<Path> find(…) / Stream<Path> walk(…)
```

Datoteke sa slučajnim pristupom

- Iako često korištena, apstrakcija tokova nije primjenjiva na sve zadatke za koje koristimo datoteke
- Za dobivanje datoteke sa slučajnim pristupom postoji razred `RandomAccessFile`, koji nudi metode tipa:
  - `getFilePointer()` i `seek(pozicija)`

Paket java.nio.channels

- Uveden od jave 1.4
- Predstavlja još jednu apstrakciju rada s tokovima uporabom kanala (Channel)
- Moćnije od podrške koju nudi java.io
  - “Pametniji” rad s memorijom i spremnicima
  - Mogućnost asinkronih U/I operacija
- Nažalost, grozno komplicirano.