1 Читање и писање текстуалних датотека у Јави 7. Пакет java.nio.

NIO је скраћеница од New I/O. Ради се о колекцији класа за улазно/излазне операције организованих у пакету **java.nio**, које су комплемент онима из стандардног пакета **java.io**. Уведене су са верзијом J2SE 1.4.

Java.io класе су оријентисане на ток, што подразумева да се бајтови читају директно из тока, тј. не врши се њихово кеширање. Померање напред и назад кроз прочитане податке није могуће, осим ако подаци нису додатно похрањени у баферу.
Java.nio класе су оријентисане на бафер, што подразумева да се подаци најпре смештају у бафер, а потом се из бафера читају. Могуће је померање напред и назад кроз прочитане податке.

У верзији Java SE 7 уводе се нове класе за опис фајл-система, под називом NIO2.
Тако, за потребе читања и писања фајлова уведен је нови пакет **java.nio.file**, који представља значајно унапређење и поправља уочене недостатке ранијих верзија.

Пакет *java.nio* и његови потпакети подржавају различита својства која се тичу улаза/излаза. Нека од њих су:

- бафери за податке примитивног типа
- енкодери и декодери за карактерске скупове
- препознавање образаца описаних регуларним изразима итд

Најзначајније класе из пакета **java.nio.file** су:

- `Paths` и `Path` – локације/имена фајлова, али не и њихов садржај
- `Files` – операције над садржајем фајла
- `StandardOpenOption` – тип енумерације, дефинише стандардне опције за отварање фајла: APPEND, CREATE, CREATE_NEW, READ, WRITE, ...

Од значаја су и:

- `StandardCharsets` и `Charset`, за кодирање текстуалних фајлова
- метод `File.toPath()`, који омогућује да старији кôд фино ради са `java.nio` API-јем.

Класа **java.nio.file.Path** служи за локализацију датотеке у фајл-систему. Обично представља путању до датотеке (која зависи од система). Путања се сматра празном уколико се састоји само од једног празног стринга. У том случају се приступа подразумеваном директоријуму.
Може се задати апсолутна путања до датотеке (почиње именом кореног директоријума) или релативна путања.

Класа **java.nio.file.Paths** садржи искључиво статичке методе за генерисање објекта класе **Path** конвертовањем стринга (ili URI-ja , java.net.URI) који представља путању.

Карактерски скуп у Јави представља пресликавање (мапирање) између секвенце бајтова и UNICODE скупа карактера (или његовог подскупа).
Пакет **java.nio.charset** описује класе за идентификацију карактерских скупова и обезбеђује алгоритме за енкодирање и декодирање.

Класа **java.nio.charset.Charset** садржи методе за прављење енкодера и декодера и за добијање имена карактерских скупова. Инстанце ове класе су непроменљиве.

Класа **java.nio.charset.StandardCharsets** садржи константне дефиниције стандардних карактерских скупова (који су гарантовано подржани на свакој имплементацији Јава платформе).
Дефиниције су типа: **public static final Charset**, а могуће вредности су `US_ASCII, ISO_8859_1, UTF_8, UTF_16BE, UTF_16LE, UTF_16`.

У зависности од величине фајла који се чита, односно у који се пише, препоручују се две, различите, технике. Напомена: примери за веће фајлове користе тзв. try-with-resources (уведен такође у Јави 7), који имплицитно затвара ресурс(е) онда када је то потребно, па клауза `finally`, која то експлицитно чини у традиционалном try-catch приступу, овде није потребна.

Мањи фајлови

а) читање:

```
final Charset KODIRANJE = StandardCharsets.UTF_8;
       // alternative:
       //     StandardCharsets.US_ASCII
       //     StandardCharsets.ISO_8859_1
       //     StandardCharsets.UTF_16;
       //     StandardCharsets.UTF_16BE - big endian
       //     StandardCharsets.UTF_16LE - little endian
String ulazniFajl = "ulaz.txt";
Path putanja = Paths.get(ulazniFajl);
try {
       // iscitaju se sve linije fajla i smeste u listu String-ova
       // (o listama ce biti reci kasnije)
       List<String> linije = Files.readAllLines(putanja, KODIRANJE);
       System.out.println(linije.toString());
              // obradjuje se (ovde samo ispisuje) dobijena lista linija
} catch(IOException e) {
       e.printStackTrace();
}
```

б) писање:

```
final Charset KODIRANJE = StandardCharsets.UTF_8;
String izlazniFajl = "izlaz.txt";
Path putanja = Paths.get(izlazniFajl);

List<String> linije; // lista linija koje ce biti upisane u fajl
linije = Arrays.asList("прва линија","ђурђевак","čČ ćĆ šŠ žŽ");
              // popunjavanje liste linijama, ovde konkretnim String-ovima
try {
       Files.write(putanja, linije, KODIRANJE);
       // Files.write(putanja, linije, KODIRANJE, StandardOpenOption.APPEND);
       // iste linije DOPISUJE NA KRAJ postojeceg fajla

       // Files.write(putanja,  linije,  KODIRANJE, StandardOpenOption.APPEND,
                                                    StandardOpenOption.CREATE);
       // a da bi se fajl kreirao ako ne postoji, neophodno je dodati i tu opciju:
       // StandardOpenOption.APPEND, StandardOpenOption.CREATE
} catch(IOException e){
       e.printStackTrace();
}
```

Већи фајлови

а) читање:

```
final Charset KODIRANJE = StandardCharsets.UTF_8;
String ulazniFajl = "podaci.txt";
Path putanja = Paths.get(ulazniFajl);

// ukoliko zelite da se koristi podrazumevano kodiranje platforme,
// STO NIJE DOBRA IDEJA, I NE PREPORUCUJE SE,
// izostavite drugi argument konstruktora - KODIRANJE.name()
// ili neka on bude: Charset.defaultCharset().name()
try(Scanner sc = new Scanner(putanja, KODIRANJE.name())){
	// citanje sadrzaja metodima klase Scanner, ovde: linija po linija
	while (sc.hasNextLine()) {
		String linija = sc.nextLine();
		System.out.println(linija); // ispis na standardni izlaz
	}
} catch(IOException e) {
	e.printStackTrace();
}
```

б) писање:

```
final Charset KODIRANJE = StandardCharsets.UTF_8;
String izlazniFajl = "izlaz.txt";
Path putanja = Paths.get(ulazniFajl);

List<String> linije; // lista linija koje ce biti upisane u fajl
// U OVOM PRISTUPU NIJE NUZNO DA SADRZAJ BUDE SMESTEN U LISTU String-ova
linije = Arrays.asList("прва линија","ђурђевак","čČ ćĆ šŠ žŽ");
		// popunjavanje liste linijama, ovde konkretnim String-ovima

/* Metod Files.newBufferedWriter() otvara ili kreira datoteku za pisanje, i vraca
 * BufferedWriter objekat koji ce se koristiti za pisanje teksta na efikasan nacin.
 */
try (BufferedWriter out = Files.newBufferedWriter(putanja, KODIRANJE)){
	// Files.newBufferedWriter(putanja, KODIRANJE, StandardOpenOption.APPEND);
	// otvaranje fajla za nadovezivanje na kraj. Fajl mora da postoji.

	// Files.newBufferedWriter(putanja, KODIRANJE, StandardOpenOption.APPEND,
	//                                              StandardOpenOption.CREATE);
	// ako fajl postoji, vrsi se dopisivanje na kraj, inace se kreira novi fajl.
	for(String linija : linije) {
		out.write(linija); // pisanje String-a linija
		out.newLine(); // pisanje oznake kraja reda
	}
} catch(IOException e){
	e.printStackTrace();
}
```

Напомена:

уколико би се од горњег `BufferedWriter`-а креирао `PrintWriter`, за њега би било могуће позивање метода `println()` и `printf()`:

```
final Charset KODIRANJE = StandardCharsets.UTF_8;
String izlazniFajl = "izlaz.txt";
Path putanja = Paths.get(ulazniFajl);
try (PrintWriter out = new PrintWriter(Files.newBufferedWriter(putanja, KODIRANJE))){
      out.println("Prva linija fajla");
      double realan = 2e-3;
      out.printf("Broj %e na 10 decimala, desno poravnat u polju sirine 15: %15.10f%n",
                  realan, realan);
} catch(IOException e){
      e.printStackTrace();
}
```