Siniša Nikolić

Java Web Development kurs – Termin 06

Sadržaj

- Rad sa klasom *HashMap* u Javi,
- Rad sa izuzecima,
- Korišćenje *third party* biblioteka
- Kreiranje jar fajla
- Završavanje domaćeg sa prošlog časa, konsultacije,

Dodatni materijal:

- Korišćenje *third party* biblioteka koje omogućuju logovanje – *Apache Log4j*
- Korišćenje *third party* biblioteka koje omogućuju čitanje i pisanje u Excel datoteke – *Apache POI*

Asocijativne mape

- Memorijske strukture koje omogućuju brzu pretragu sadržaja po ključu
- Element se ubacuje u paru sa svojim ključem, koji mora da bude jedinstven

Rad sa klasom *HashMap* u Javi

- Predstavlja asocijativnu mapu
- U HashMap se stavljaju dva podatka:
  - ključ po kojem će se pretraživati
  - vrednost koja se skladišti u HashMap i koja se pretražuje po ključu
- Metodom put() se ključ i vrednost smeštaju u HashMap
- Metodom get() se na osnovu ključa dobavlja (samo čita) vrednost iz HashMap
- ako se ne nađe ključ, vratiće null

Rad sa klasom *HashMap* u Javi

```
HashMap<String, Student> hm = new HashMap<String, Student>();

hm.put("E10020", new Student("E10020", "Marko Markovic"));
hm.put("E10045", new Student("E10045", "Petar Petrovic"));
hm.put("E10093", new Student("E10093", "Jovan Jovanovic"));
String indeks = "E10045";
System.out.println("Student sa indeksom "+ indeks
    + " je " + hm.get(indeks).getIme());
indeks = "AAAAAA";
System.out.println("Student sa indeksom "+ indeks
    + " je " + hm.get(indeks).getIme()); // greska
```

Rad sa klasom *HashMap* u Javi

- Primeri dva različita programa koji upravlja radom studentske službe. Svi studenti se čuvaju u kolekciji tipa
  - lista (korišćenje kolekcije ArrayList)
  - mapa(korišćenje kolekcije HashMap)
  - pređenje metoda
    - pronadjiStudenta
    - ispisiSveStudente
    - sortirajStudentePoImenu

primer 01

Rad sa izuzecima

- Mehanizam prijavljivanja greške
- Greška se signalizira "bacanjem" izuzetka
- Metoda koja poziva potencijalno "grešnu" metodu "hvata" izuzetak
- Hijerarhija klasa započinje klasom *Throwable* – roditeljska klasa
  - *Error* – ozbiljne sistemske graške (npr. *VirtualMachineError* – is broken or has run out of resources necessary for it to continue operating, *CoderMalfunctionError*, *FactoryConfigurationError*,...)
  - *Exception* – bazna klasa za sve standardne izuzetke
    - *unchecked*: RuntimeException i njene naslednice – ne moraju da se obuhvate try/catch blokom
    - *checked*: Ostale klase koje nasleđuju Exception klasu i koje moraju da se obuhvate try/catch blokom

Rad sa izuzecima

- Checked (Exception i njene naslednice) – moraju da se uhvate EOFException, SQLException, FileNotFoundException, IOException, ...
- Unchecked (RuntimeException i njene naslednice) – ne moraju da se uhvate, jer mogu da se programski spreče NullPointerException, IndexOutOfBoundsException, ClassCastException, NumberFormatException,...

Rad sa izuzecima

```
try {
    // kod koji može da izazove
    // izuzetak
} catch (FileNotFoundException ex) {
    System.out.println("Datoteka ne postoji!");
} catch (ClassCastException ex) {
    System.out.println("Zabranjena konverzija");
} catch (IndexOutOfBoundsException ex) {
    System.out.println("Pristup van granica niza");
} catch (Exception ex) {
    System.out.println("Svi ostali izuzeci");
} finally {
    // kod koji se izvršava u svakom slučaju
}
```

Rad sa izuzecima

- Programsko izazivanje izuzetka

```
throw new Exception("Ovo je jedan izuzetak");
```

- Korisnički definisani izuzeci

```
class IzuzetakNeispravanSlog extends Exception {
    String slog;
    String token;

    public IzuzetakNeispravanSlog(String slog, String token){
        this.slog = slog;
        this.token = token;
    }
    public void ispisIzuzetak (){
        System.out.println("Neispravan slog " + slog + " kod
        tokena " + token);
    }
}
```

Rad sa izuzecima

- Ključna reč throws

```
void f(int i) throws MojException { ... }
```

- Propagacija izuzetaka
  - ne moramo da obuhvatimo try-catch blokom, već da deklarišemo da i pozivajuća metoda takođe baca izuzetak
  - tako možemo da prebacujemo odgovornost hvatanja izuzetka na gore

primer 02

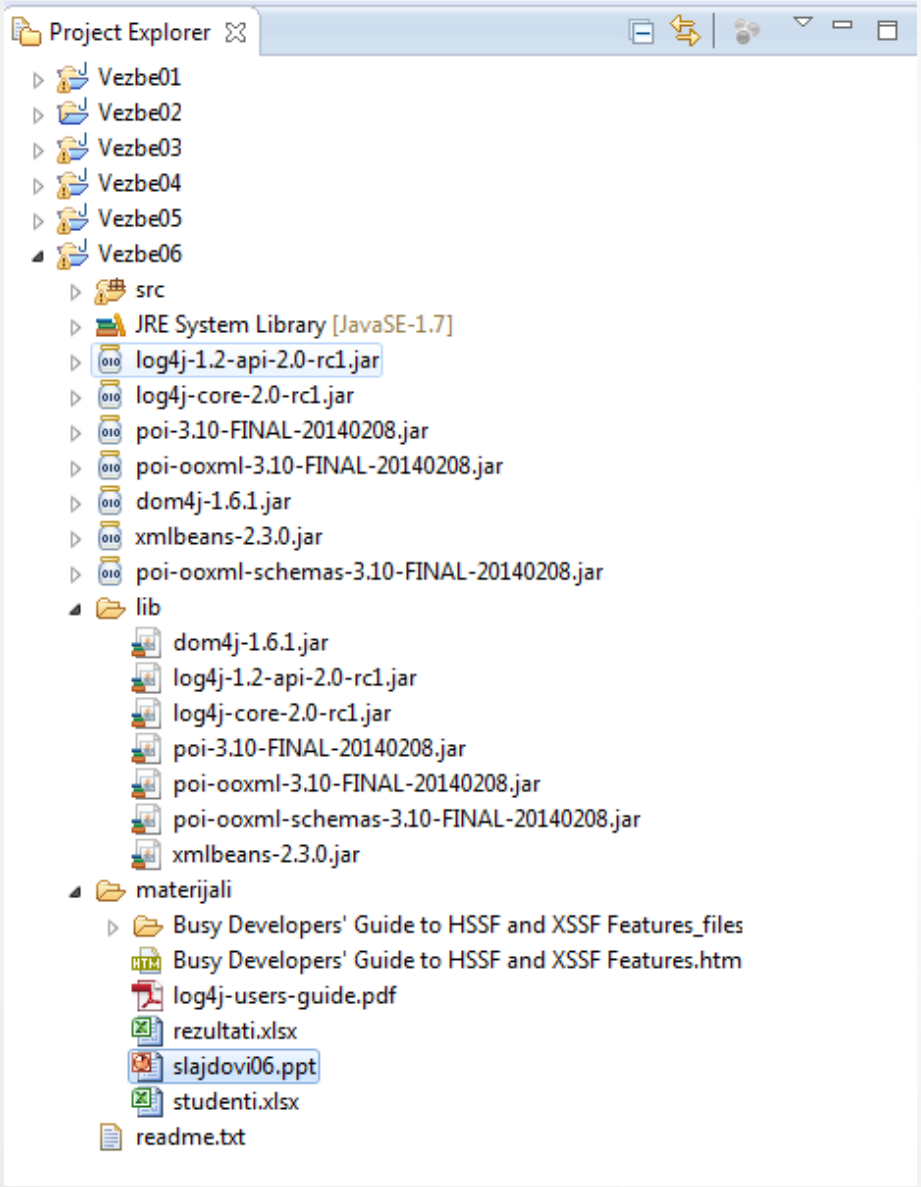
Korišćenje *third party* biblioteka

- Koriste se da prošire mogućnosti postojećih programa
- Ne izmišljamo točak! Jer je neko je već uradio nešto slično ili isto na zadatu temu
- Pretraga po Google/forumi/literatura
- Kada nabavimo biblioteku, šta dalje?

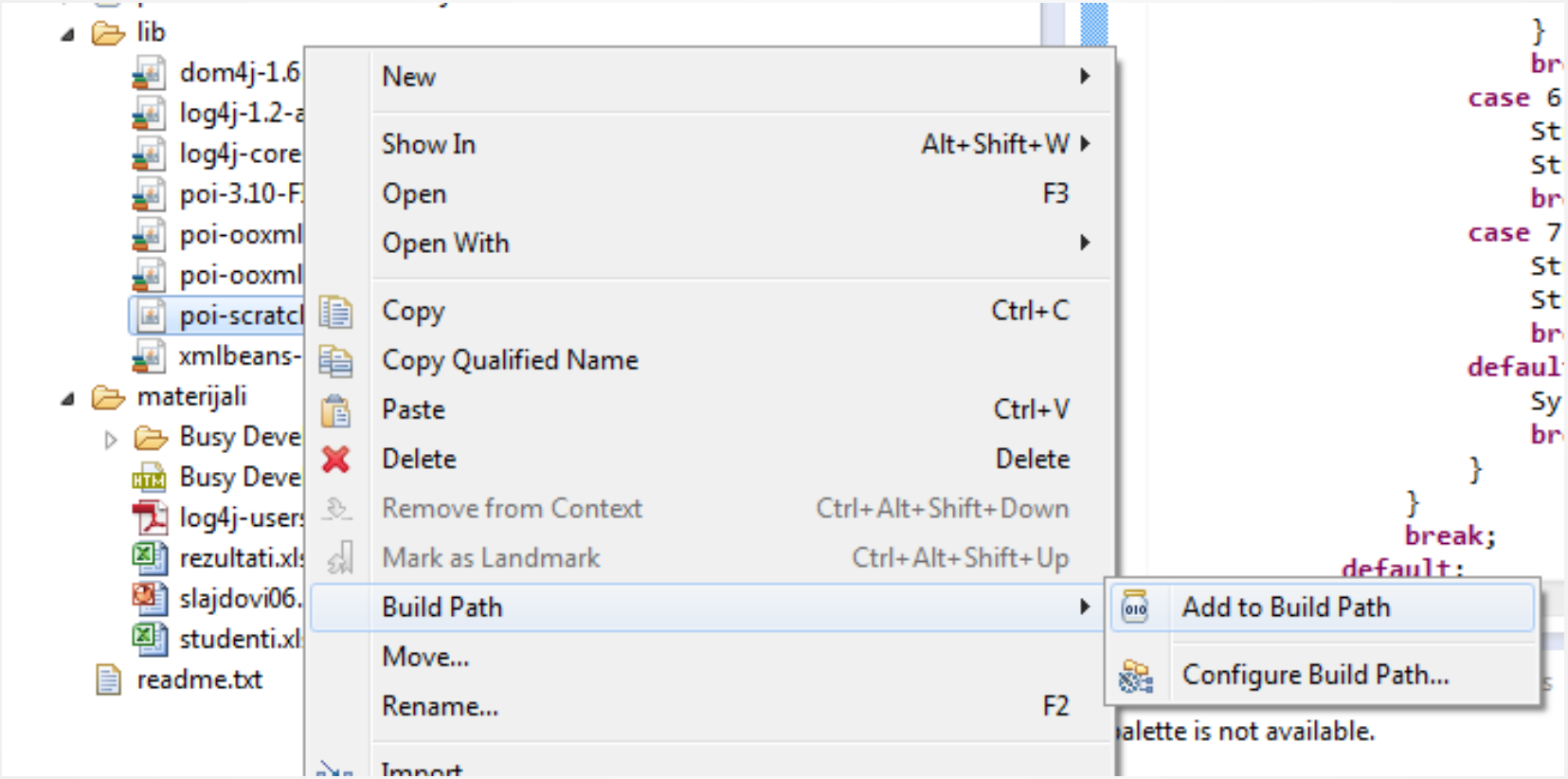
Korišćenje *third party* biblioteka

- Biblioteke su obično u obliku jar arhive
- Jar arhiva se mora dodati u classpath projekta
- Obično se napravi **lib** folder i u njega se iskopiraju jar arhive
- Ti jar-ovi se zatim dodaju u build path

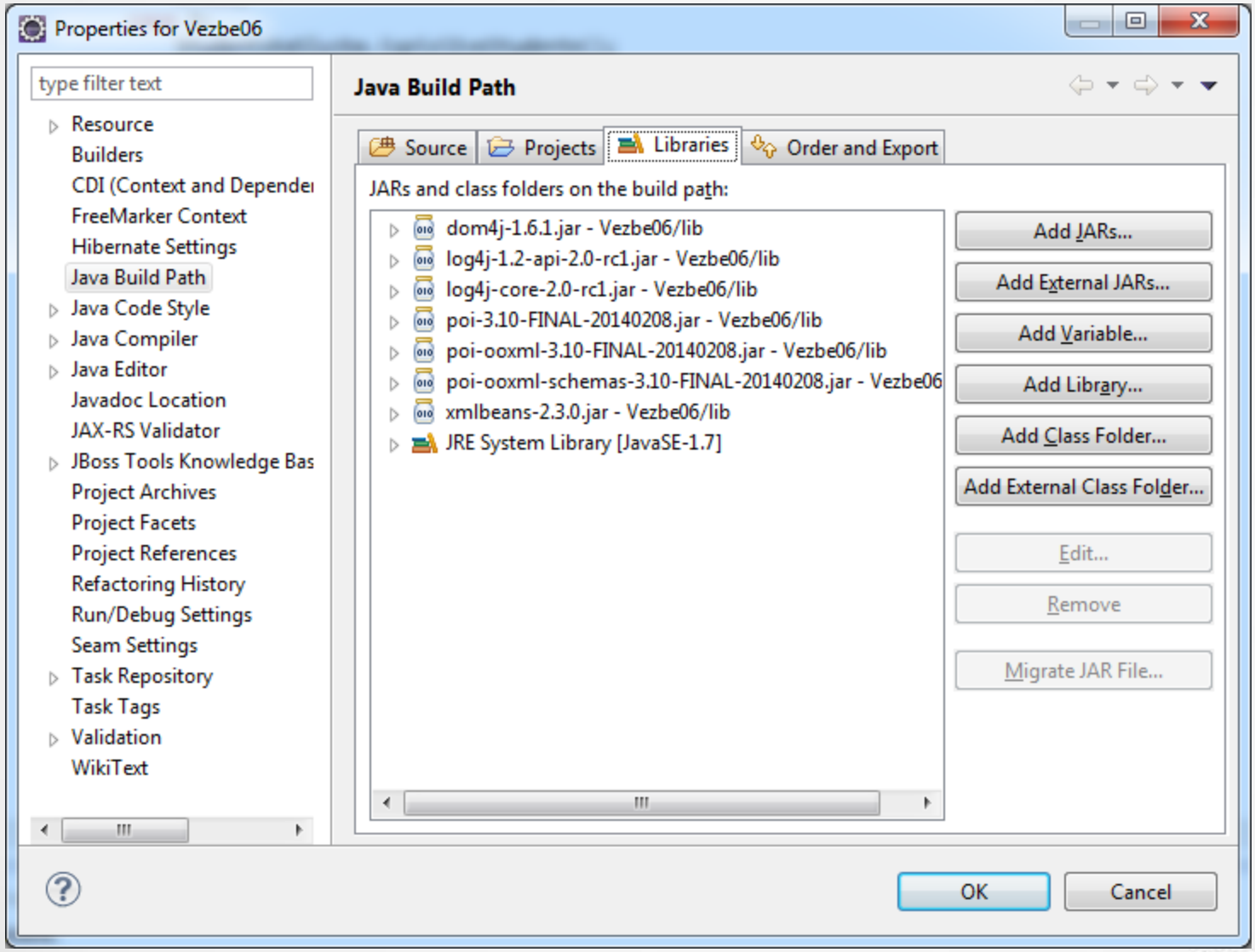
Korišćenje *third party* biblioteka – dodavanje u projekat



Korišćenje *third party* biblioteka – dodavanje u projekat



Korišćenje *third party* biblioteka – dodavanje u projekat



zadatak 01

primer 03

Dodatni Materijal

Korišćenje *third party* biblioteka koje omogućuju logovanje – Apache Log4j

- Biblioteka za logovanje (zapisivanje dnevnika)
- Open source
- http://logging.apache.org/log4j/2.x/
- Ispis bitnih stvari u konzolu/fajl/nešto drugo
- Dodavanje u projekat
  - log4j api
  - log4j core

Korišćenje *third party* biblioteka koje omogućuju logovanje – Apache Log4j

- Osnovna upotreba:

```
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
public class HelloWorld {
  private static Logger logger =
      LogManager.getLogger("HelloWorld");
  public static void main(String[] args) {
    logger.error("Hello, World!");
  }
}
```

Korišćenje *third party* biblioteka koje omogućuju logovanje – Apache Log4j

- Moguće je postaviti različite nivoe za lovovanje:
  - off – nema logovanja (0)
  - fatal – greška, ali će aplikacija najverovatnije stati (100)
  - error – greška, ali aplikacija bi trebalo da nastavi (200)
  - warn – upozorenje (300)
  - info – informacija (400)
  - debug – debagiranje (500)
  - trace – debagiranje, finije od debug (600)
  - all – sve se loguje (Integer.MAX_VALUE)

Korišćenje *third party* biblioteka koje omogućuju logovanje – Apache Log4j

- Moguće je konfigururisati rad biblioteke na više načina:
  - Logger traži konfiguracioni fajl u "log4j.configurationFile" environment varijabli
  - zatim traži "log4j2-test.json" ili "log4j2-test.jsn" datoteku
  - zatim traži "log4j2-test.xml" datoteku
  - zatim traži "log4j2.json" ili "log4j2.jsn" datoteku
  - zatim traži "log4j2.xml" datoteku, i, na kraju
  - koristi klasu DefaultConfiguration

Korišćenje *third party* biblioteka koje omogućuju logovanje – Apache Log4j

- Primer log4j2.xml datoteke:

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration>
  <Appenders>
    <Console name="Console" target="SYSTEM_OUT">
      <PatternLayout pattern="%d{HH:mm:ss.SSS} [%t] %-
   5level %logger{36} - %msg%n" />
    </Console>
  </Appenders>
  <Loggers>
    <Root level="all">
      <AppenderRef ref="Console" />
    </Root>
  </Loggers>
</Configuration>
```

Korišćenje *third party* biblioteka koje omogućuju logovanje – Apache Log4j

- Doda se File appender:

```
<Appenders>
  <Console name="Console" target="SYSTEM_OUT">
    <PatternLayout pattern="%d{HH:mm:ss.SSS} [%t] %-
   5level %logger{36} - %msg%n" />
  </Console>

  <File name="MyFile" fileName="./logs/app.log">
    <PatternLayout>
      <Pattern>%d %p %c{1} [%t] %m%n</Pattern>
    </PatternLayout>
  </File>
</Appenders>
```

Korišćenje *third party* biblioteka koje omogućuju logovanje – Apache Log4j

- Primer:

```
<PatternLayout pattern="
%d{HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n
" />
```

- Neki najčešći parametri:
  - %d – datum i/ili vreme
  - %t – ime niti (thread) iz iz koje se loguje
  - %logger (ili %c) – klasa koja se loguje
  - %c{1} – samo ime klase, bez imena paketa
  - %msg (ili %m) – poruka
  - %n – novi red

primerDodatnoLogovanje

Korišćenje *third party* biblioteka koje omogućuju čitanje i pisanje u Excel datoteke – *Apache POI*

- Biblioteka za rad sa Microsoft Office dokumentima
- Open source
- http://poi.apache.org/
- Čitanje i pisanje MSOffice dokumenata
- Dodavanje u projekat:
  - poi-3.10-FINAL-20140208.jar
  - poi-ooxml-3.10-FINAL-20140208.jar
  - poi-ooxml-schemas-3.10-FINAL-20140208.jar
  - xmlbeans-2.3.0.jar
  - dom4j-1.6.1.jar
  - po potrebi dodati još...

Korišćenje *third party* biblioteka koje omogućuju čitanje i pisanje u Excel datoteke – *Apache POI*

- Osnovna klasa je Workbook
- Dobija se iz fabrike:

```
Workbook wb = WorkbookFactory.create(new
   FileInputStream("racuni.xlsx"));
Workbook wb = WorkbookFactory.create(new
   File("racuni.xlsx"));
```

- Kreiranje novog/overwrite postojećeg Excel fajla

```
Workbook wb = new XSSFWorkbook();
FileOutputStream fileOut = new
   FileOutputStream("racuni.xlsx");
wb.write(fileOut);
fileOut.close();
```

Korišćenje *third party* biblioteka koje omogućuju čitanje i pisanje u Excel datoteke – *Apache POI*

- Rad sa listovima
  - Kreiranje:

```
Sheet sheet = wb.createSheet("new sheet");
```

  - Pristup:

```
//na osnovu indeksa u nizu listova, ideksi počinju od 0
Sheet sheet = wb.getSheetAt(0);
//ili na osnovu imena
wb.getSheet("Sheet1")
```

Korišćenje *third party* biblioteka koje omogućuju čitanje i pisanje u Excel datoteke – *Apache POI*

- Rad sa redovima
  - Kreiranje:

```
Row row = sheet.createRow(0);
```

  - Pristup:

```
//na foreach petlje
 for (Row row : sheet) {
   // izbegnemo prvu vrstu (zaglavlje)
   if (row.getRowNum() == 0)
     continue;
   ...
 }
//ili na osnovu pozicije
 Row row = sheet.getRow(i);
```

Korišćenje *third party* biblioteka koje omogućuju čitanje i pisanje u Excel datoteke – *Apache POI*

- Rad sa ćelijama koje se nalaze u određenoj koloni
  - Kreiranje:

```
Cell cell = row.createCell(0);
```

  - Pristup:

```
//na foreach petlje
for (Cell cell : row) ...
//ili na osnovu pozicije
r = row.getCell(i);
d = row.getCell(i, Row.RETURN_BLANK_AS_NULL);
```

  - Čitanje:

```
d = ocena.getNumericCellValue();
s = grad.getStringCellValue();
```

  - Upis nove vrednosti:

```
cell.setCellValue(1);
```

Korišćenje *third party* biblioteka koje omogućuju čitanje i pisanje u Excel datoteke – *Apache POI*

- Brisanje reda

```
/** Remove a row by its index
* @param sheet a Excel sheet
* @param rowIndex a 0 based index of removing row */
public static void removeRow(HSSFSheet sheet, int
rowIndex) {
  int lastRowNum=sheet.getLastRowNum();
  if(rowIndex>=0 && rowIndex<lastRowNum){
    sheet.shiftRows(rowIndex+1,lastRowNum, -1);
  }
  if(rowIndex==lastRowNum){
    HSSFRow removingRow=sheet.getRow(rowIndex);
    if(removingRow!=null){
      sheet.removeRow(removingRow);
    }
  }
}
```

primerDodatnoEksel