Vježbe 4 - iznimke, rad s datotekama, polja, spremnici

Matej Mihelčić

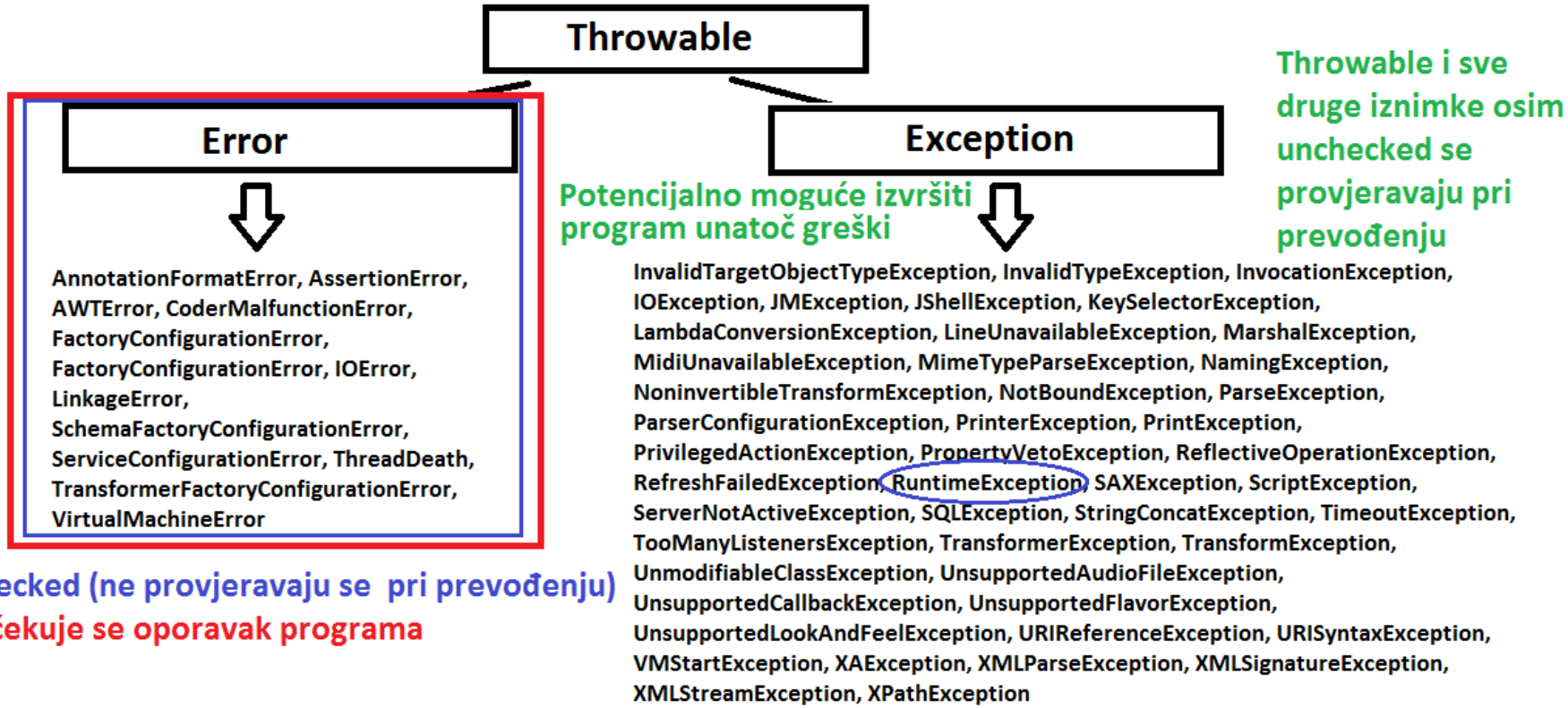
Prirodoslovno-matematički fakultet, Sveučilište u Zagrebu

matmih@math.hr

02. studenoga, 2022.

Iznimke

Iznimke se **izbacuju (dojavljaju)** kada program **prekrši neka semantička ograničenja** *Java* programskog jezika ili **određena ograničenja postavljena od strane programera**.



`Throwable` i sve njene podklase osim `Error` i `RuntimeException` koji je podklasa klase `Exception` se **provjeravaju pri prevođenju**.

Iznimke

Klase i sučelja pogrešaka i iznimaka se nalaze u gotovo svim potpaketima paketa `java`, iako se **većina nalazi** u paketu `java.lang`.

```
1  public class Iznimke {
2      static int vrati(int i)throws java.util.
       InputMismatchException{
3          if(i<0)
4              throw new java.util.InputMismatchException("
       argument < 0!"); //koristi klasa Scanner
5  //moze i java.lang.IllegalArgumentException (mozda i cesce
       koristeno)
6          return i;
7  }
8
9      public static void main(String args[]){
10         Iznimke.vrati(-4); //iznimka se ne provjerava
11     }
12
13 }
```

Primjer iznimke koja se ne provjerava

Iznimke

```
1  public class Iznimke {
2    static java.io.BufferedReader otvoriDatoteku(java.nio.file
       .Path p) throws java.io.IOException{
3          java.io.BufferedReader reader = java.nio.file.Files.
       newBufferedReader(p,java.nio.charset.StandardCharsets.
       UTF_8);
4                  return reader;
5          }
6
7     public static void main(String args[]){
8          java.io.File input = new java.io.File("");
9          java.nio.file.Path p = java.nio.file.Paths.get(input
       .getAbsolutePath());
10         java.io.BufferedReader citac = otvoriDatoteku(p); //
       pogreska pri prevodenju jer se iznimka provjerava
11    }
12 }
```

Primjer iznimke koja se provjerava

Iznimke

```
1  public class Iznimke {
2   static java.io.BufferedReader otvoriDatoteku(java.nio.file.
       Path p) throws java.io.IOException{
3         java.io.BufferedReader reader= java.nio.file.Files.
       newBufferedReader(p,java.nio.charset.StandardCharsets.
       UTF_8);
4                     return reader; }
5
6   public static void main(String args[]){
7    java.io.File input = new java.io.File("");
8    java.nio.file.Path p = java.nio.file.Paths.get(input.
       getAbsolutePath());
9    try{
10    java.io.BufferedReader citac = otvoriDatoteku(p);//OK,
       iznimka je obradena u catch dijelu, dodatkom finally
       mozda moze i nastaviti normalno izvrsavanje
11         }
12   catch(java.io.IOException e){ //moze i nadklasa Exception
13            e.printStackTrace(); }  }  }
```

Primjer iznimke koja se provjerava

Rad s datotekama

Klasa `File` iz paketa `java.io` **omogućava rad s datotekama**.

```
1  import java.io.File;  //ukljucujemo klasu File iz paketa
       java.io
2  import java.io.IOException;  // Isto za IOException
3
4  public class StvoriDatoteku {
5   public static void main(String[] args) {//stvaranje
6     try { //datoteke izbacuje iznimku koja se provjerava
7      //sada smijemo pozivati kod Windowsa
8        File datoteka = new File("putanja\\imedatoteke.txt");
9  File datoteka = new File("putanja/imedatoteke.txt");//Linux
10       if (datoteka.createNewFile()) {
11 System.out.println("Nova datoteka: " + datoteka.getName());
12       } else {
13         System.out.println("Datoteka vec postoji.");
14       }
15     } catch (IOException e) {
16       System.out.println("Doslo je do greske.");
17       e.printStackTrace(); } } }
```

Primjer rada s datotekama

Rad s datotekama

```
1  import java.io.FileWriter;   // Koristimo klasu FileWriter
       iz paketa java.io
2  import java.io.IOException;
3
4  public class PisiUdatoteku {
5    public static void main(String[] args) {
6      try {
7        FileWriter pisac = new FileWriter("
       putanjaDelimiterImedatoteke.txt");
8        pisac.write("Neki podaci!");//ne prebacuje u novi red,
        treba dodati '\n' za to
9        pisac.write("Drugi podatak\n"); //nakon upisa "Drugi
       podatak" prelazak u novi red
10       pisac.close(); //zatvaramo pisac
11       System.out.println("Pisanje uspjesno.");
12     } catch (IOException e) {
13       e.printStackTrace();//ispis opisa greske
14   } }
15 }
```

Pisanje u datoteku

Rad s datotekama

```
1  import java.io.File;
2  import java.io.FileNotFoundException;
3  import java.util.Scanner;
4
5  public class CitajDatoteku {
6   public static void main(String[] args) {
7     try {
8  File datoteka = new File("putanjaDelimiterImeDatoteke.txt");
9        Scanner ucitavac = new Scanner(datoteka);
10       while (ucitavac.hasNextLine()) {
11         String linija = ucitavac.nextLine();
12         System.out.println(linija);
13       }
14       ucitavac.close();//zatvaramo ucitavac
15     } catch (FileNotFoundException e) {
16       System.out.println("Greska.");
17       e.printStackTrace();
18     } } }
```

Čitanje datoteke

Brisanje datoteka

```
1  import java.io.File;
2
3  public class BrisiDatoteku {
4    public static void main(String[] args) {
5  File datoteka = new File("putanjaDelimiterImedatoteke.txt");
6      if (datoteka.delete()) {
7  System.out.println("Izbrisana datoteka: " + datoteka.getName
       ());
8      } else {
9        System.out.println("Nije uspjelo.");
10     }  } }
```

Brisanje datoteke

Delimiter - znak za odvajanje elemenata putanje je '\\' kod operacijskog sustava **Windows** (zapravo se radi o posebnom znaku '\' koji se dobiva kao '\\') i '/' kod operacijskih sustava **Linux**. Cijela putanje je oblika: `Drive:\\mapa`$_1$`\\mapa`$_2$`\\...\\mapa`$_n$`\\imeDatoteke.txt` (Windows), `/mapa`$_1$`/mapa`$_2$`/.../mapa`$_n$`/imeDatoteke.txt` (Linux).

Polja

Polja su posebni **dinamički kreirani objekti** u *Javi* koji nasljeđuju klasu `Object`. Njihova uloga je **spremanje niza elemenata osnovnog ili referenciranog tipa**. **Višedimenzionalna polja** imaju **rekurzivnu definiciju** (kao i u $C/C++$-u, vidi predavanja). Niz znakova **nije identičan** `String`-u (`String` je **nepromijenjivi objekt** (eng. *immutable*), svaka izmjena uzrokuje stvaranje nove instance, dok se elementi polja mogu mijenjati). Niti polje znakova niti `String` ne sadrže '\0' (null ili terminalni znak). **Deklaracija polja ne stvara objekt**, već samo varijablu (referencu).

```
1 int[] jdpoljeIntova; //jednodimenzionalno polje intova
2 int jdpoljeIntova1[]; //jednodimenzionalno polje intova
3 short[][] ddpoljeShortova;//polje, polja shortova
4 short s, dds[][];   // varijabla i polje polja short-ova
5 Object[]  o1, o2;   // dva polja objekata
6 Collection<?>[] poljeSpremnika;  // polje spremnika
    nepoznatog tipa (s gornjom ogradom Object)
```

Deklaracija polja

Polja

```
Exception iznimke[]  = new Exception[3];
Object ddpoljeObjekata[][]  = new Exception[2][3];
int[] jdpoljeIntova = { 1, 1, 2, 6, 24, 120, 720, 5040 };
char jdpoljeZnakova[]        = { 'n', 'o', 't', ' ', 'a', ' '
    , 'S', 't', 'r', 'i', 'n', 'g' };
String[] jdpoljeStringova    = { "polje", "Stringova"};
```

Deklaracija i inicijalizacija polja

Inicijalizacija polja stvara objekt i u njega **potencijalno sprema zadane podatke**. **Ime tipa određenog objekta** dobivamo pozivom naredbe `getClass().getName()` na instancu objekta.

```
int[][] brojevi = { {1, 2, 3, 4}, {5, 6, 7} };
for (int i = 0; i < brojevi.length; ++i) {
      for(int j = 0; j < brojevi[i].length; ++j) {
        System.out.print(brojevi[i][j]+" ");
      }
      System.out.println();
    }
```

Iteriranje po elementima polja

Polja

Duljina polja `length` je **konstantan element član** instance polja.

```
int adva[][] = {{1,2,3,4},{1,2,3}};
        for(int i[]:adva){
            for(int j:i)
            System.out.print(j+" ");
        System.out.println();
        }

System.out.println(adva[7][3]); //Exception in thread "main"
     java.lang.ArrayIndexOutOfBoundsException: Index 7 out
    of bounds for length 2
```

Iteriranje po elementima polja

Za razliku od *C/C++*-a, *Java* **vrši provjeru pristupa elementima polja** (odnosno dolazi do iznimke ukoliko se pristupa indeksima koji nisu sadržani u polju).

Spremnici

Ne mapirajući spremnici u *Javi* su **klase iz paketa** `java.util` **koji nasljeđuju generičku klasu** `AbstractCollection<E>`, a ta klasa nasljeđuje klasu `Object`. Klase spremnika **implementiraju sučelje** `Collection<E>`. Predstavljaju **generičke**, **proširive spremnike** koji **mogu spremati elemente samo referenciranih tipova podataka**.

Paket `java.util` sadrži sljedeće spremnike:
- `ArrayList<E>` - **proširiva implementacija liste** koja koristi polje za spremanje elemenata.
- `Vector<E>` - **proširivo polje objekata**. Objektima se može pristupati koristeći `int` indeks.
- `Stack<E>` - **reprezentacija stoga objekata**. Proširuje klasu `Vector<E>`.
- `PriorityQueue<E>` - **baziran na hrpi**, **elementi su uređeni koristeći prirodni poredak** ili koristeći **posebno konstruirani komparator**.
- `ArrayDeque<E>` - **proširiva implementacija polja** koja **sadrži razne funkcije dvostranog reda**. **Vjerojatno brža od klase** `Stack<E>` (kada se koristi za reprezentaciju stoga) i **brža od** `LinkedList` (kada se koristi za reprezentaciju reda).

Spremnici

- `EnumSet<E>` - **specijalna implementacija skupa** za korištenje s tipom `enum`.
- `LinkedList<E>` - **implementacija dvostruko povezane liste** (**može se koristiti i kao red**).

Mapirajući spremnici
Mapirajući spremnici implementiraju klasu `AbstractMap<K,V>` ili `Dictionary<K,V>` i **implementiraju sučelje** `Map<K,V>` ili **interno koriste klasu** koja to čini.

- Hashtable<K,V> - **implementira hash tablicu** koja **mapira ključ na odgovarajuću vrijednost**, nasljeđuje klasu `Dictionary`. Svaki objekt **mora implementirati metodu** `hashCode` i `equals` da bi se mogao koristiti.
- HashMap<K,V> - **novija implementacija mapiranja ključa na vrijednost**. **Poboljšanje u odnosu na Hashtable**. Dozvoljava i korištenje `null`.

Spremnici

- TreeMap<K,V> - **implementacija mapiranja ključ-vrijednost bazirana na crveno-crnim stablima**. **Ključevi sortirani prema prirodnom uređaju elemenata** ili **posebno konstruiranom komparatoru**.
- `LinkedHashMap` - **implementacija mapiranja ključ-vrijednost** koja **dodatno koristi dvostruko povezanu listu** za **određivanje poretka iteriranja** po ključevima (**jednak poretku ubacivanja ključeva u mapu**).
- `HashSet<E>` - **implementacija skupa** koja interno koristi `HashMap`. **Nema garancija na poredak elemenata**.
- `LinkedHashSet<E>` - **implementacija skupa** koja **koristi** `LinkedHashMap` **koji koristi dvostruko povezanu listu**. Dvostruko povezana lista **osigurava** da se **iteriranje po elementima** odvija u **poretku u kojem su ubačeni u skup** (ponovno ubacivanje ne mijenja poredak).
- `TreeSet<E>` - **implementacija uređenog skupa** koji **interno koristi** `TreeMap`. Elementi su **uređeni prema prirodnom uređaju** ili **koristeći posebno konstruirani komparator**.

Spremnici

```java
import java.util.ArrayList;

ArrayList<Integer> lista0 = new ArrayList<>(); //deklaracija i inicijalizacija, koristimo operator dijamant <> koji sam zakljuci tip liste (desna strana iza new)
lista0.add(0); lista0.add(1); lista0.add(2); //dodavanje elemenata na kraj liste (pri svakom dodavanju se int konstante pakiraju u referencirani tip Integer)
lista0.add(0,15); //dodaje element na odredeni indeks (0 u ovom slucaju) u listu
System.out.println(lista0); //ArrayList ima nadjacanu funkciju toString(), ispis: [15, 0, 1, 2]
int b = lista0.get(2);//dohvacanje 3 elementa liste, otpakiravanje i spremanje u varijablu b
int indeks = lista0.indexOf(b); //vraca indeks prvog pojavljivanja b u listi
lista0.set(2,3); //vrijednost elementa na indeksu 2 promijeni u 3
```

Rad s klasom ArrayList.

Spremnici

```
1  lista0.remove(1); //brise element na indeksu 1.
2  Integer c = lista0.get(2);
3  lista0.remove(c); //brise prvo pojavljivanje objekta c
4  int velicina = lista0.size(); //sprema velicinu liste u
       varijablu velicina
5  List<Integer> podlista = lista0.sublist(0,2); //vraca dio
       liste izmedu indeksa 0 (ukljucivo) i 2 (iskljucivo)
6  Object p1[] = lista0.toArray(); //vraca polje objekata koje
       odgovara listi
7  Integer polje[] = {1,2,3,4,5,6,7};
8  List<Integer> lista = new ArrayList<Integer>();
9  Collections.addAll(lista, polje); //kopira elemente polja u
       listu (tipovi moraju odgovarati)
10 List<Integer> lista1 = Arrays.asList(polje); //vraca listu
       koja odgovara polju (tipovi moraju odgovarati)
11 int polje2[] = {0,2,4,6,8,10};
12 ArrayList<Integer> lista2 = new ArrayList<>();
13     for(int i:polje2)
14      lista2.add(i);//radi unatoc potrebe za pakiranjem
```

Rad s klasom ArrayList.

Spremnici

```
1  public class Par extends Object{
2      int x, y; //extends Object ne treba pisati
3  //zelimo koristiti svoju klasu kao tip unutar liste
4      Par(){ x = y = 0; }
5      Par(int xval, int yval){x = xval; y = yval;}
6
7      @Override public String toString(){
8          return "("+x+","+y+")";
9      }   }
10 //ukoliko zelimo omoguciti usporedbu elemenata novog tipa,
       trebamo napraviti komparator
11     public class KomparatorParova implements Comparator<Par>
        {
12     @Override public int compare(Par prvi, Par drugi){
13         if(prvi.x>drugi.x)
14             return 1;
15         else if(prvi.x == drugi.x){
16             if(prvi.y>drugi.y)
17                 return 1;
```

Rad s klasom ArrayList.

Spremnici

```
            else if(prvi.y == drugi.y)
                return 0;
            else return -1;
        }
        else return -1; } }
//sada mozemo napraviti
ArrayList<Par> lista3 = new ArrayList<Par>();
Par p1 = new Par(2,4); Par p2 = new Par(0,9); Par p3 = new
    Par(7,2); Par p4 = new Par(7,8); Par p5 = new Par(6,4);
    Par p6 = new Par(7,2);
lista3.add(p1); lista3.add(p2); lista3.add(p3); lista3.add(
    p4); lista3.add(p5); lista3.add(p6); lista3.add(p1);
    lista3.add(p1);
lista3.sort(new KomparatorParova());
System.out.println(lista3);// [(0,9), (2,4), (2,4), (2,4),
    (6,4), (7,2), (7,2), (7,8)]
lista3.get(2).x = 3;
System.out.println(lista3);// [(0,9), (3,4), (3,4), (3,4),
    (6,4), (7,2), (7,2), (7,8)]
```

Rad s klasom ArrayList.

Spremnici

```
1  //Mozemo definirati i odredene akcije na paru, npr.
       kvadrirati par
2  public class AkcijaNaParu implements Consumer<Par> {
3      @Override public void accept(Par p){
4              p.x = p.x*p.x;
5              p.y = p.y*p.y;
6      }
7  }
8  //sada mozemo napraviti
9  lista3.forEach(new AkcijaNaParu());
10 System.out.println(lista3); //[(0,81), (6561,65536),
       (6561,65536), (6561,65536), (36,16), (49,4), (49,4),
       (49,64)]
11 System.out.println(lista3.indexOf(new Par(49,4)));// vraca
       -1, razlog poziva equals() metodu na razini klase Object
        (u principu gleda jednakost referenci). -1 oznacava da
       tocno taj objekt nije u listi. Ukoliko zelimo
       provjeravati po vrijednosti, treba nadjacati metodu
       equals() u klasi Par.
```

Rad s klasom ArrayList.

Spremnici

```
1  @Override public boolean equals(Object p){
2              if(this == p)
3                  return true;// ukoliko je to referenca na
       isti objekt, pronasli smo ga
4              if (!(p instanceof Par)) //ukoliko objekt nije
       klase Par, sigurno nije ono sto trazimo
5                      return false;
6              Par tmp = (Par) p; //sada moramo ispitati
       vrijednost, da bi to mogli koristimo eksplicitnu
       konverziju da pretvorimo objekt p u objekt tipa Par (
       mozemo posto smo zakljucili da je objekt ispravnog tipa)
7              if((Integer.compare(x, tmp.x) == 0) && (Integer.
       compare(y, tmp.y) == 0)) //radimo provjeru jednakosti
8                  return true;
9              return false;
10     }
11 //sada kada pozovemo
12 System.out.println(lista3.indexOf(new Par(49,4)));//
       dobijemo 5 kao sto bi i zeljeli
```

Rad s klasom ArrayList.

Spremnici

```
1  HashMap<String,Integer> brojac = new HashMap<>();//mapa
   koja mapira objekte klase String u objekte klase Integer
2       BufferedReader citac = new BufferedReader ( new
   InputStreamReader ( System . in ) ) ;
3       int kraj=0, br=0;
4       String poc = "";
5
6       while(kraj!=1){
7           poc = citac.readLine();
8           if(poc.trim().equals("kraj")){//citamo rijeci
   dok se ne ucita "kraj"
9               kraj = 1;
10              break;
11          }
```

Rad s klasom HashMap.

Spremnici

```
1  if(brojac.containsKey(poc)){//provjeravamo postoji li u
   mapi kljuc jednak string objektu poc
2                br = brojac.get(poc); //ukoliko postoji,
   dohvati povezani Integer, spremi u varijablu tipa int (
   dogada se otpakiravanje)
3                br++;//inkrementiraj, jos jedno
   pojavljivanje Stringa poc
4                brojac.put(poc,br);//povezi poc s novim
   brojem pojavljivanja
5              }
6              else brojac.put(poc,1); //prvo pojavljivanje,
   povezi poc s brojem 1
7         }
8         System.out.println(brojac);//mapa ima nadjacanu
   metodu toString()
9         citac.close();//zatvorimo citac
10        brojac.clear();//oslobodimo svu memoriju koju
   zauzima mapa brojac, nakon poziva je brojac prazna mapa
```

Rad s klasom HashMap.

Spremnici

```
1  brojac.put("nesto", 1);
2  System.out.println(brojac); //{nesto=1}
3  brojac.compute("nesto",(key,val)->(val+12)); //mijenjamo
       vrijednost koju mapira kljuc "nesto"
4  System.out.println(brojac); //{nesto=13}
5
6  //moze i ovako
7  public class MijenjajMapu implements BiFunction<String,
       Integer,Integer> {
8          @Override public Integer apply(String t, Integer u)
       {
9              return 0;
10         }
11 }
12
13 brojac.compute("nesto", new MijenjajMapu());
14 System.out.println(brojac);//{nesto=0}
15 System.out.println(brojac.containsValue(0));// provjerava
       sadrzi li mapa vrijednost 0
```

Rad s klasom HashMap.

Spremnici

```
1 Set<Entry<String,Integer>> skupParova = brojac.entrySet();//
    vraca skup parova (kljuc, vrijednost)
2 int k = brojac.getOrDefault("b",0);//dohvati vrijednosti
    mapiranu na kljuc ili vrati "default" zadan kao drugi
    parametar
3 brojac.remove("nesto"); //brise mapiranje sa zadanim kljucem
    , moze i (kljuc, vrijednost) parom iz mape. Vraca
    vrijednost koja je prije bila asocirana s kljucem ili
    null ukoliko kljuc ne postoji
4 System.out.println(brojac); //{}
5 System.out.println(brojac.size()); //vraca broj mapiranja u
    mapi
6 Collection<Integer> vls = brojac.values();//vraca Collection
     pogled na mapu koja sadrzi vrijednosti. Efektivno
    omogucava da se vrijednostima pristupa kao da su u listi
     (mozemo iterirati koristeci iterator ili for(Tip c:
    kolekcija))
```

Rad s klasom HashMap.

Spremnici

```
1  int vrijednost = brojac.merge("nesto", 100, (staraVrijednost
       , novaVrijednost) -> staraVrijednost + novaVrijednost);
       //ubacuje par ("nesto", 100) u mapu ukoliko kljuc "nesto
       " ne postoji u mapi, inace izvrsava funkciju
       remappingFunction koja definira ponasanje. Vraca novu
       vrijednost povezanu s kljucem ili null.
2
3  //iteriranje po mapi
4  Iterator<String> it = brojac.keySet().iterator();//dohvatimo
        iterator skupa kljuceva u mapi
5     while(it.hasNext()){ //iteriramo po elementima
6        String kljuc = it.next(); //dohvatimo sljedeci kljuc
7        Integer vrijednost = brojac.get(kljuc);
8        System.out.println("( "+kljuc+" , "+vrijednost+")");//
       mozemo koristiti iterator za iteriranje po svim
       spremnicima koji implementiraju sucelje Iterable (gotovo
        svi java core spremnici)
9                  }
10    }
```

Rad s klasom HashMap.

Zadatak

Zadatak - zadaća

Proučite ostale spremnike i njihove funkcionalnosti.

Zadatak 1

U datotekama `matrica1.txt` i `matrica2.txt` su zapisane dvije matrice A dimenzija $m \times n$ i B dimenzija $n \times k$. U datotekama nisu posebno navedene dimenzije matrica već su samo elementi zapisani po redcima (elementi u svakom retku su odvojeni običnim razmakom). Napišite program koji provjerava jesu li datoteke ispravno zapisane (sadrže li zaista matricu), učitava navedene matrice, sprema ih u dvodimenzionalna polja koja sadrže elemente osnovnog tipa `double`, izračunajte $C = A \cdot B$ te spremite matricu C po redcima u datoteku `rezultat.txt`.

Zadatak

Zadatak 2

U tekstualnoj datoteci `brojevi.txt` su zapisane znamenke $0-9$. Svaki redak sadrži niz znamenki nepoznate duljine, znamenke su odvojene razmakom. Nije poznato koliko redaka je zapisano u datoteku. Napišite program koji računa brojeve pojavljivanja znamenaka u svakom retku. Program treba u datoteku `izlaz.txt` u svaki redak zapisati parove brojeva (odvojene razmakom): $(Zn, Count)$, gdje Zn predstavlja znamenku $0-9$ a $Count$ broj pojavljivanja znamenke u retku. U svakom retku izlazne datoteke mora biti zapisan broj pojavljivanja svih znamenaka (uključujući i one koje imaju broj pojavljivanja jednak 0).

Zadatak 3

Implementirajte Dijsktrin algoritam za traženje najkraćeg puta u težinskom grafu od zadanog čvora s do svih drugih čvorova u grafu. Graf reprezentirajte preko liste susjedstva.