4 Collectii

4.1 Collection

Operatii:

```
Collection collection = new ArrayList(); //crearea colectiei
collection.add(new Object()); //adaugarea unui obiect
collection.add(new Object());//adaugarea altui obiect

boolean wasElementRemoved = collection.remove(anObject); //eliminarea
                                    //unui element din colectie

Collection collection2 = new LinkedList();
collection2.add(new Object());
collection2.addAll(collection); //adaugarea unei colectii la alta colectie
collection2.removeAll(collection); //eliminarea elementelor unei colectii
                                    din alta colectie

//verificarea existentei unui element in lista
boolean containsElement = collection2.contains("an element");
boolean containsAll     = collection2.containsAll(elements); //verificare
                                    existentei mai multor elemente

Iterator iterator = collection.iterator();
while(iterator.hasNext()){          //parcurgerea colectiei cu iterator
  Object object = iterator.next();
  System.out.println(object);
}

for(Object object : collection) {   //parcurgerea colectiei cu for-each
    System.out.println(object);
}
```

Interfete care implementeaza interfata java Collection:

- List
- Set
- SortedSet
- NavigableSet
- Queue
- Deque

Mai multe la http://tutorials.jenkov.com/java-collections/collection.html

4.2 Liste

Implementari ale interfetei java.util.List:

- java.util.ArrayList
- java.util.LinkedList
- java.util.Vector
- java.util.Stack

```
List listA = new ArrayList();
List listB = new LinkedList();
List listC = new Vector();
List listD = new Stack();
```

Operatii cu liste:

```
List<String> list1 = new ArrayList<String>();
list1.add("elem a");  //adaugare element
list1.add(1, "elem B");  //adaugare element la o pozitie data
list1.add(2, "elem B");

List list2 = new ArrayList<String>();
list2.addAll(list1);

String elementA = list1.get(0);  //obtinerea unui element din lista de pe
                                 //pozitia 0
String elementB = list1.get(1);  //obtinerea unui element din lista de pe
                                 // pozitia 1

int index1 = list.indexOf(elementA);
int index2 = list.indexOf(elementB);

int lastIndex = list.lastIndexOf(elementB);//obtinerea ultimei aparitii unui
                                           //element


boolean containsElement = list.contains("elem a"); //verificarea existentei
                                                   // unui element

list.remove(elementA);                             //eliminarea unui element
                                                   // din lista
list.remove(1); //eliminarea unui element de pe o anumita pozitie
list.clear(); //eliminarea tuturor elementelor din lista.
int size = list.size(); //dimensiunea listei. Nr de elemente din lista.
list.retainAll(otherList); //retine in lista doar elementele prezente si in
                           //otherList

List sublist = list.subList(1, 3); //crearea unei subliste cu elemente din
                                   //prima lista.

Object[] objects = list.toArray(); //convertirea listei la un array
String[] objects1 = list1.toArray();

String[] values = new String[]{ "one", "two", "three" };
```

```
List list = Arrays.asList(values);  //covertirea unui array la List

Iterator<String> iterator = list.iterator();
while(iterator.hasNext()){       //parcurgere lista cu iterator
    String obj = iterator.next();
}

for(Object element : list) {     //parcurgere lista cu for-each
    System.out.println(element);
}

for(int i=0; i < list.size(); i++) {  //parcurgere lista cu for i to n
    Object element = list.get(i);
}

Stream<String> stream = stringList.stream();
stream.forEach( element -> { System.out.println(element); }); //parcurgere
                                    // lista folosind stream api (java 1.8+)

Collections.sort(list);  //sortarea unei liste

//sortare cu comparator specific
List<Car> list = new ArrayList<>();
list.add(new Car("Volvo V40" , "XYZ 201845", 5));
list.add(new Car("Citroen C1", "ABC 164521", 4));
list.add(new Car("Dodge Ram" , "KLM 845990", 2));
Comparator<Car> carBrandComparator = new Comparator<Car>() {
    @Override
    public int compare(Car car1, Car car2) {
        return car1.brand.compareTo(car2.brand);
    }
};
Collections.sort(list, carBrandComparator);
```

4.3 Set

Un Set este similar cu List cu exceptia faptului ca in Set acelasi element nu poate fi gasit de mai multe ori, pe cand intr-o lista acelasi element poate fi gasit pe mai multe pozitii.
De asemenea intr-un set ordinea interna nu este garantata. Intr-o lista ordinea interna exista si elementele pot fi iterate in acea ordine.
Implementari:

- java.util.EnumSet
- java.util.HashSet
- java.util.LinkedHashSet
- java.util.TreeSet

```
Set setA = new EnumSet();
```

```
Set setB = new HashSet();
Set setC = new LinkedHashSet();
Set setD = new TreeSet();
```

Operatii pe Set:

```
Set setA = new HashSet();
setA.add("element 1");   //adaugare elemente in Set
setA.add("element 2");
set.remove("object-to-remove"); //eliminare element din Set
set.clear();                    //eliminare tuturor elementor din Set

Set set2 = new HashSet();
set2.addAll(set);               //adaugare elemente din alt Set
set.removeAll(set2);            //eliminare elemente din alt Set

int size = set.size();           //nr de elemente in Set
boolean isEmpty = set.isEmpty();     //verificare daca setul este gol
boolean isEmpty = (set.size() == 0); //verificare daca setul este gol
boolean containsA = set.contains("A");//verificare daca setul contine un
                                      // element

//Generics
Set<MyObject> set = new HashSet<MyObject>(); //folosire 'generics'
                                             //(java 1.7+)

// conversie Set in List
List list = new ArrayList();
list.addAll(set);

Iterator iterator = set.iterator();
while(iterator.hasNext(){  //parcurgere Set cu iterator
  String element = (String) iterator.next();
}

for(Object object : set) {  //parcurgere Set cu for-each
    String element = (String) object;
}

Stream stream = set.stream();//parcurgere cu stream si lambda (java 1.8+)
stream.forEach((element) -> { System.out.println(element); });
```

4.4 Map

java.util.Map face legatura dintre o cheie si o valoare. Aceasta structura permite stocarea perechilor cheie-valoare.

Implementari:

- java.util.HashMap
- java.util.Hashtable
- java.util.EnumMap
- java.util.IdentityHashMap
- java.util.LinkedHashMap
- java.util.Properties
- java.util.TreeMap
- java.util.WeakHashMap

Operatii:

```
Map mapA = new HashMap();
mapA.put("key1", "element 1");
mapA.put("key2", Integer.valueOf(123));
mapA.put(null, "elem"); //HashMap permite cheie nula.
mapA.put("key3", null); //valori nule sunt permise
map.replace("key1", "new value"); //daca se gaseste cheia,
                                  //atunci se suprascrie valoarea
mapA.put("key3", "newVal"); //dupa executia acestei metode,
                            //cheii key3 i se asociaza obiectul "newVal"
Object value1 = mapA.get("key1"); //citirea elementului asociat
                                  // cheii "key1"
Object value2 = mapA.get("key2");
mapA.remove("key1"); //eliminare element cheie-valoare
mapA.clear(); //eliminarea tuturor elementelor
Object value3 = map.getOrDefault("key4", "default value"); //daca nu
                        //gaseste cheia "key4", returneaza "default value"
boolean hasKey = mapA.containsKey("key1");//verifica existenta unei chei
boolean hasValue = mapA.containsValue("element 1");//verifica existenta
                                                   //unei valori
int entryCount = mapA.size();

Map mapB = new TreeMap();
mapB.putAll(mapA);   //inserarea tuturor elementelor altui obiect Map

Iterator iterator = mapA.keySet().iterator();
while(iterator.hasNext(){  //parcurgere Map folosind iterator
  Object key   = iterator.next();
  Object value = mapA.get(key);
}

for(Object key : mapA.keySet()) { //parcurgere Map folosind for-each
    Object value = mapA.get(key);
}
```

Laborator 4

```
Stream<String> stream = map.keySet().stream(); //parcurgere cu stream si
                                               //lambda functions(java 1.8+)
stream.forEach((value) -> {
    System.out.println(value);
});

Iterator iterator = map.values().iterator(); //parcurgere a valorilor
while(iterator.hasNext()) {
    Object nextValue  iterator.next();
}

for(Object value : mapA.values()){ //parcurgere valori cu for-each
    System.out.println(value);
}

Stream<String> stream = mapA.values().stream(); //parcurgere valori
                                        //cu stream si lambda functions
stream.forEach((value) -> {
    System.out.println(value);
});

Set entries = map.entrySet(); //parcurgerea intrarilor in Map, cu iterator
Iterator iterator = entries.iterator();
while(iterator.hasNext()) {
    Map.Entry entry = (Map.Entry) iterator.next();
    Object key = entry.getKey();
    Object value = entry.getValue();
}

for(Object entryObj : map.entrySet()){
    Map.Entry entry = (Map.Entry) entryObj;
    Object key = entry.getKey();
    Object value = entry.getValue();
}
```

Laborator 4

4.5 Probleme

4.5.1 Fie declaratiile:

```
List<Integer> x = new ArrayList();
List<Integer> y = new ArrayList();
List<Integer> xPlusY = new ArrayList(); //a
Set<Integer>  zSet = new TreeSet();     //b
List<Integer> xMinusY = new ArrayList();//c
int p = 4;
List<Integer> xPlusYLimitedByP = new ArrayList(); //d
```

Dati valori aleatorii elementele listelor x şi y in domeniul [0..10] şi apoi ordonati aceste doua liste crescător.
Numarul de elemente pt lista x este 5. Numarul de elemente pt lista y este 7.
Să se scrie cod pentru a obţine colectii cu elemente ordonate crescător, după efectuarea următoarelor operaţii:
a) xPlusY conţine toate elementele şirurilor x şi y;
b) zSet conţine numai valorile comune din ambele şiruri, luate o singură dată;
c) xMinusY conţine valorile din şirul x care nu se află în şirul y;
d) xPlusYLimitedByP este mulţimea elementelor din x şi y ce nu depăşesc valoarea p.

4.5.2 Se consideră o lista ce contine date referitoare la studenţi, organizate astfel: nume-prenume, grupa şi cinci note. Scrieţi un program pentru a rezolva următoarele probleme:

a) Dati valori aleatorii pt notele studentilor in intervalul [4..10]
b) afişarea elementelor în ordinea:
   b1) alfabetică, pe grupe;
   b2) descrescătoare a mediilor celor cinci note - pentru integralişti;
   b3) crescătoare a numărului de restanţe - pentru restanţieri

4.5.3 initializati lista de studenti prin citirea unui fisier text. Pe fiecare linie se afla detaliile unui student separate de cate un spatiu: nume prenume grupa nota1 nota2 nota3 ...
Introduceti apoi aceste intr-un map (Map<Student, Integer>) in care sa numarati aparitiile fiecarui student.
Pentru lucrul cu obiecte ale claselor scrise de noi trebuie sa generati, in clasa Student, equals() si hashcode(). IntelliJ Idea poate face asta: ALT + Insert (sau meniu > Code > Generate sau click-dreapta > Generate).

Fisierul se numeste input.txt si are continutul:
Ion Popescu 223_1 7 9 6 4
Maria Popa 223_2 5 6 7 8
Ion Popescu 223_1 7 9 6 4
Diana Oprea 223_2 10 4 5 4
Elena Dragomir 223_3 9 8 7 8
Ion Popescu 223_1 7 9 6 4

Laborator 4

Hints

4.5.1:

```
java.util.Random rand = new java.util.Random();
rand.nextInt(31);                          //nr aleator in intervalul [0..30]
```

Pt usurare Implementati in clasa Student metoda:

```
public String toString() {
    return String.format("%s (%s)", nume, grupa);
}
```

4.5.2:

Lista studentilor ar putea fi:

```
List<Student> studenti = new ArrayList();
```

Iar clasa Sudent (declarata in fisierul Student.java !):

```
public class Student {

    private String nume;
    private String grupa;
    private List<Integer> note;

    public Student(String nume, String grupa){
        this.nume = nume;
        this.grupa = grupa;
        note = new ArrayList();
    }

    public String getNume() {
        ...
    }

    public String getGrupa() {
        ...
    }

    public void adaugaNota(int nota){
        ...
    }

    public boolean esteIntegralist(){
        ...
    }

    public float getMedie(){
        ...
    }
```

```
}
```

b) Construiti liste separate pt integralisti, restantieri. Folositi clase comparatoare dedicate pt fiecare punct.

ex de comparator:

```
public class ByGradesComparator implements Comparator<Student> {

    @Override
    public int compare(Student o1, Student o2) {
        if(o1.getMedie() == o2.getMedie()) {
            return 0;
        }
        return o1.getMedie() < o2.getMedie() ? 1 : -1; //descrescator
    }
}
```

Exemplu de utilizare

```
      Collections.sort(integralisti, new ByGradesComparator());
```

4.5.3

```
static Map<Student, Integer> readInputFile(String name) {
    Map<Student, Integer> students = new HashMap<>();
    BufferedReader buffReader = null;
    try {
        File file = new File(name);
        if (!file.exists()) {
            System.out.println("File not found : " + file.getAbsolutePath());
            return students;
        }
        buffReader = new BufferedReader(new FileReader(file));
        String line;
        while ((line = buffReader.readLine()) != null) {
            .....
        }
    } catch (IOException ioe) {
        ioe.printStackTrace();
        return null;
    } finally {
        if (buffReader != null) {
            try {
                buffReader.close();
            } catch (IOException ioe) {
            }
        }
    }
    return students;
}
```

4.6 Referinte

Lïst
http://tutorials.jenkov.com/java-collections/list.html

Set
http://tutorials.jenkov.com/java-collections/set.html

Map
http://tutorials.jenkov.com/java-collections/map.html

Java Collections
http://tutorials.jenkov.com/java-collections/index.html

Comparator Interface in Java with Examples
https://www.geeksforgeeks.org/comparator-interface-java/

Hashing
https://www.geeksforgeeks.org/hashing-in-java/