



Principet e Avancuara të Gjuheve të Programimit

Lamir Shkurti, Can. PhD.
lamirshkurti@gmail.com

COLLECTION NË JAVA



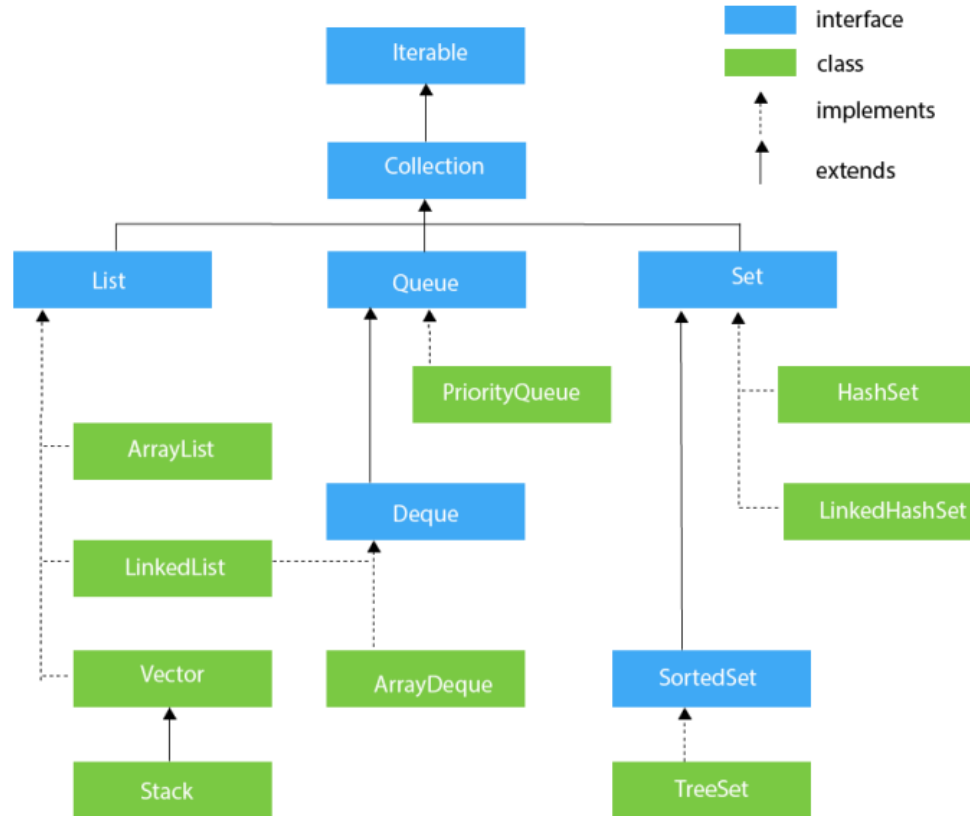
- Collection (Koleksioni) në Java është një framework (kornizë) që ofron një arkitekturë për të ruajtur dhe manipuluar një grup të objekteve.
- Collections në Java mund të arrijnë të gjitha operacionet që ju mund të kryeni me të dhëna (data) si: kërkimi, klasifikimi, futja e të dhënave, manipulimi dhe fshirja e tyre.
- Java Collection nënkupton një njësi të vetme të objekteve. Java Collection framework (kornizë) ofron shumë interface (Set, List, Queue, Deque) dhe klasa (ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet etj).

COLLECTION NË JAVA



- Çfarë është Collection në Java
 - Një Collection përfaqëson një njësi të vetme të objekteve, d.m.th., një grup.
- Çfarë është Collection framework
 - Collection framework përfaqëson një arkitekturë të unifikuar për ruajtjen dhe manipulimin e një grupi objektsh. Ajo ka:
 - Interfaca dhe implementimet e tyre, d.m.th., klasat
 - Algoritmet

HIERARKIA E COLLECTION FRAMEWORK



DISA METODA TË COLLECTION INTERFACE

Method	Description
<code>public boolean add(E e)</code>	It is used to insert an element in this collection.
<code>public boolean addAll(Collection<? extends E> c)</code>	It is used to insert the specified collection elements in the invoking collection.
<code>public boolean remove(Object element)</code>	It is used to delete an element from the collection.
<code>public boolean removeAll(Collection<?> c)</code>	It is used to delete all the elements of the specified collection from the invoking collection.
<code>public int size()</code>	It returns the total number of elements in the collection.
<code>public void clear()</code>	It removes the total number of elements from the collection.
<code>public boolean contains(Object element)</code>	It is used to search an element.
<code>public boolean containsAll(Collection<?> c)</code>	It is used to search the specified collection in the collection.
<code>public Iterator iterator()</code>	It returns an iterator.
<code>public Object[] toArray()</code>	It converts collection into array.
<code>public boolean isEmpty()</code>	It checks if collection is empty.
<code>public boolean equals(Object element)</code>	It matches two collections.
<code>public int hashCode()</code>	It returns the hash code number of the collection.

ITERATOR INTERFACE



- Iterator interface siguron iterimin e elementeve vetëm në një drejtim (drejtimin përpara).
- Ekzistojnë vetëm tre metoda në këtë interface. Ato janë:

Method	Description
<code>public boolean hasNext()</code>	It returns true if the iterator has more elements otherwise it returns false.
<code>public Object next()</code>	It returns the element and moves the cursor pointer to the next element.
<code>public void remove()</code>	It removes the last elements returned by the iterator. It is less used.

ITERABLE INTERFACE



- Ky interface është interface root për të gjitha klasat e collection. Interfaci i collection bën extend interface-in.
- Ky interface përmban vetëm një metodë abstrakte
- `Iterator<T> iterator()` Kthen iteratorin mbi elementet e tipit T.

COLLECTION INTERFACE



- Ky interface zbatohet nga të gjitha klasat në collection framework.
- Deklaron metodat që do të ketë çdo collection.
- Me fjalë të tjera, mund të themi se interfaci i collection ndërton bazën nga e cila varet collection framework.

LIST INTERFACE



- List interface është interface fëmijë i Collection interface.
- List Interface implementohet nga klasat ArrayList, LinkedList, Vector dhe Stack.

```
List<data-type> list1 = new ArrayList();
```

```
List<data-type> list2 = new LinkedList();
```

```
List<data-type> list3 = new Vector();
```

```
List<data-type> list4 = new Stack();
```

ARRAY LIST INTERFACE



- Klasa ArrayList implementon List interface.
- Përdor një array dinamik për të ruajtur elementin kopjues të llojeve të ndryshme të të dhënave.
- Klasa ArrayList mban rendin e futjes së të dhënave (insertion order) dhe nuk është e sinkronizuar.
- Elementet e ruajtura në klasën ArrayList mund të aksesohen (Random Access Interface) rastësisht.

SHEMBULL 1



```
import java.util.ArrayList;
public class ArrayList0{
    public static void main(String args[]) {
        ArrayList<String> list = new ArrayList<String>();
        //Creating arraylist
        list.add("Mango");//Adding object in arraylist
        list.add("Apple");
        list.add("Banana");
        list.add("Grapes");
        //Printing the arraylist object
        System.out.println(list);
    }
}
```

SHEMBULL 2



```
import java.util.ArrayList;
import java.util.Iterator;
public class ArrayList1{
    public static void main(String args[]) {
        ArrayList<String> list = new ArrayList<String>(); //Creating arraylist
        list.add("Test 1"); //Adding object in arraylist
        list.add("Test 2");
        list.add("Test 1");
        list.add("Test 3");
        //Përshkrimi i listës duke përdorur Iterator
        Iterator itr = list.iterator();
        while (itr.hasNext()) {
            System.out.println(itr.next());
        }
    }
}
```

SHEMBULL 3



```
public class ArrayList3{
    public static void main(String args[]) {
        ArrayList<String> list = new ArrayList<String>(); //Creating
        arraylist
        list.add("Mango"); //Adding object in arraylist
        list.add("Apple");
        list.add("Banana");
        list.add("Grapes");
        //Traversing list through for-each loop
        for (String fruit:list)
            System.out.println(fruit);
    }
}
```

LINKDED LIST



- LinkedList implementon Collection interface.
- Mund të ruajë elementet e kopjuara (duplicate elements)
- Mban rendin e futjes së të dhënave (insertion order) dhe nuk është e sinkronizuar.
- Në LinkedList manipulimi është i shpejtë sepse nuk kërkohet zhvendosja e elementeve

SHEMBULL 1



```
import java.util.LinkedList;
public class LinkedList0{
    public static void main(String args[]) {
        LinkedList<String> list = new LinkedList<String>(); //Creating linkedlist
        list.add("Mango"); //Adding object in LinkedList
        list.add("Apple");
        list.add("Banana");
        list.add("Grapes");
        //Printing the LinkedList
        System.out.println(list);
    }
}
```

SHEMBULL 2



```
import java.util.Iterator;
import java.util.LinkedList;
public class LinkedList1{
    public static void main(String args[]) {
        LinkedList<String> list = new LinkedList<String>(); //Creating
        arraylist
        list.add("Test 1"); //Adding object in arraylist
        list.add("Test 2");
        list.add("Test 1");
        list.add("Test 3");
        //Përshkrimi i listës duke përdorur Iterator
        Iterator<String> itr = list.iterator();
        while (itr.hasNext()) {
            System.out.println(itr.next());
        }
    }
}
```


SHEMBULL 3



```
import java.util.LinkedList;
public class LinkedList3{
    public static void main(String args[]) {
        LinkedList<String> list = new LinkedList<String>();
        //Creating arraylist
        list.add("Mango"); //Adding object in arraylist
        list.add("Apple");
        list.add("Banana");
        list.add("Grapes");
        //Traversing list through for-each loop
        for (String fruit:list)
            System.out.println(fruit);
    }
}
```

VEKTORËT



- Vector përdorë një array dinamik për të ruajtur të dhënat.
- Është e ngjajshme me ArrayList, mirëpo është e sinkronizuar dhe përmban disa metoda të cilat nuk janë pjesë e Collection Framework

SHEMBULL 1



```
import java.util.Vector;
public class Vector0{
    public static void main(String args[]) {
        Vector<String> list = new Vector<String>(); //Creating a vector
        list.add("Mango");//Adding elements using add() method
        list.add("Apple");
        list.add("Banana");
        list.add("Grapes");
        System.out.println(list);
    }
}
```

SHEMBULL 2



```
import java.util.Iterator;
import java.util.Vector;
public class Vector1{
    public static void main(String args[]) {
        Vector <String> list = new Vector <String>();
        list.add("Test 1");
        list.add("Test 2");
        list.add("Test 1");
        list.add("Test 3");
        Iterator<String> itr = list.iterator();
        while (itr.hasNext()) {
            System.out.println(itr.next());
        }
    }
}
```

SHEMBULL 3



```
import java.util.Vector;
public class Vector3{
    public static void main(String args[]) {
        Vector<String> list = new Vector<String>();
        list.add("Mango");
        list.add("Apple");
        list.add("Banana");
        list.add("Grapes");
        for (String fruit:list)
            System.out.println(fruit);
    }
}
```

HASHTABLE



Pikat për të mbajtur mend

- Një Hashtable është array e një liste. Çdo listë është e njohur si bucket. Pozicioni i bucket identifikohet duke thirrur metodën `hashCode()`. Një Hashtable përmban vlera të bazuara në çelës.
- Klasa `Hashtable` përmban elemente unike.
- Klasa `Hashtable` nuk lejon vlera `null` (çelësin ose vlerën `null`).

METODAT E HASHTABLE



Metoda	Përshkrimi	Input	Output
Hashtable	Krijon një hashtable të re të zbrazët	Asnjë	Asnjë
put	Shton qiftin e dhënë <çelësi, vlera> në tabelë	Dy parametrat të tipit Object	Një referencë të tipit Object
get	Kthen vlerën e asocuar me çelësin e dhënë	Një parametër të tipit Object	Një referencë të tipit Object
remove	Fshinë çelësin dhe vlerën e asocuar me të nga hashtable	Një parametër të tipit Object	Një referencë të tipit Object
size	Kthen numrin e çelësve në hashtable	Asnjë	Një vlerë të tipit int

MAP INTERFACE



- Map është një strukturë e të dhënave e cila mbështet lidhjen key-value
- Ky interface nuk mbështet çelësat e kopjuar (duplicate keys)
- Një Map është e dobishme nëse ekzistojnë të dhëna dhe ne dëshirojmë të kryejmë operacione në bazë të çelësit.

SHEMBULL 1



```
import java.util.Hashtable;

public class Hashtable0{
    public static void main(String args[]) {
        Hashtable<Integer, String> hm = new Hashtable<Integer, String>();
        hm.put(1, "Test 1");
        hm.put(2, "Test 2");
        hm.put(3, "Test 3");
        hm.put(4, "Test 4");
        System.out.println(hm);
    }
}
```

SHEMBULL 2



```
import java.util.HashMap;
import java.util.Map;
public class HashTable1{
    public static void main(String[] args) {
        HashMap<String, Integer> hm = new HashMap<String, Integer>();
        // Adding mappings to HashMap
        hm.put("Filani Fisteku", 54);
        hm.put("Filan 2", 80);
        hm.put("Filan 3", 82);
        // Printing the HashMap
        System.out.println("Created hashmap is" + hm);
        // Loop through the hashmap
        System.out.println("HashMap for each: ");
        // Using for-each loop
        for (Map.Entry mapElement: hm.entrySet()) {
            System.out.println (mapElement.getKey() + " : " + mapElement.getValue());
        }
    }
}
```

SHEMBULL 2



```
public class Book {  
    int id;  
    String name, author, publisher;  
    int quantity;  
    public Book (int id, String name, String author, String  
publisher, int quantity) {  
        this.id = id;  
        this.name = name;  
        this.author = author;  
        this.publisher = publisher;  
        this.quantity = quantity;  
    }  
}
```

SHEMBULL 2



```
import java.util.Hashtable;
import java.util.Map;
public class HashtableExample {
    public static void main(String[] args) {
        //Creating map of Books
        Map<Integer, Book> map = new Hashtable<Integer, Book>();
        //Creating Books
        Book b1=new Book (101, "Let us C", "Yashwant Kanetkar", "BPB",8);
        Book b2=new Book (102, "Data Communications & Networking", "Forouzan", "Mc Graw Hill", 4);
        Book b3=new Book (103, "Operating System", "Galvin", "Wiley",6);
        //Adding Books to map
        map.put (1,b1);
        map.put (2,b2);
        map.put (3,b3);
        //Traversing map
        for (Map.Entry<Integer, Book> entry: map.entrySet()){
            int key = entry.getKey();
            Book b = entry.getValue();
            System.out.println (key+" Details: ");
            System.out.println (b.id+" "+b.name+" "+b.author+" "+b.publisher+" "+b.quantity);
        }
    }
}
```

Principet e Avancuara të Gjuhëve të Programimit

