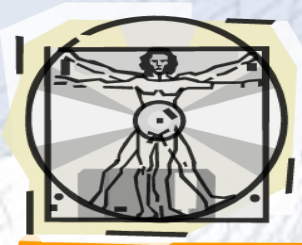
Programming Languages - II
Lists, Maps and Sets

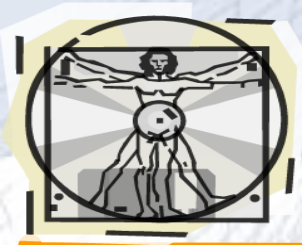
Özgür Koray ŞAHİNGÖZ
Prof.Dr.

Biruni University
Computer Engineering Department



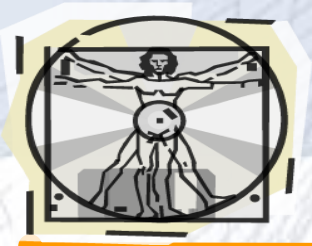
Change Array Size

```
public class Main {
    public static void main(String[] args) {
    int[] numberArray = { 12, 24, 63, 45 };
    System.out.println("Array before ReSize: ");
    for (int i = 0; i < numberArray.length; i++)
        System.out.println(numberArray[i]);

    numberArray = new int[6];
    numberArray[4]=71;
    numberArray[5]=98;

    System.out.println("Array after ReSize: ");

    for (int i = 0; i < numberArray.length; i++)
        System.out.println(numberArray[i]);
    } }
```



Change Array Size

```
public class Main {
    public static void main(String[] args) {
    int[] numberArray = { 12, 24, 63, 45 };
    System.out.println("Array before ReSize: ");
    for (int i = 0; i < numberArray.length; i++)
        System.out.println(numberArray[i]);

    int[] temp = new int[6];
    int length = numberArray.length;

    for (int j = 0; j < length; j++)
        temp[j] = numberArray[j];
    numberArray = temp;
    System.out.println("Array after ReSize: ");

    for (int i = 0; i < numberArray.length; i++)
        System.out.println(numberArray[i]);
    } }
```

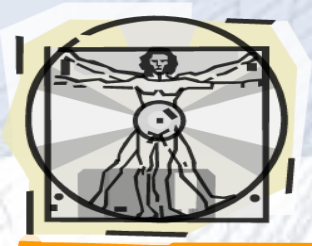


New Solutions

- This is not a dynamic structure

- To use a dynamic (easily resized arrays we need to get a help from LIBRARIES.

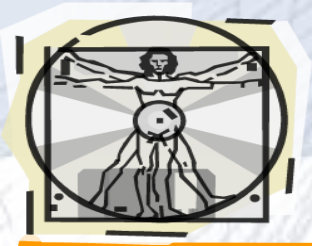
- Lists
- Maps
- Sets



LinkedList

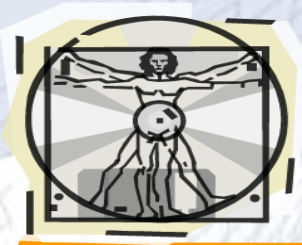
- In the previous chapter, you learned about the ArrayList class.
- The LinkedList class is almost identical to the ArrayList:

```
import java.util.LinkedList;
public class Main {
  public static void main(String[] args) {
    LinkedList<String> cars = new LinkedList<String>();
    cars.add("Volvo");
    cars.add("BMW");
    cars.add("Ford");
    cars.add("Mazda");
    System.out.println(cars);
  }
}
```



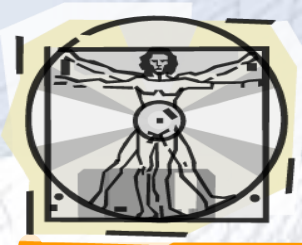
ArrayList vs. LinkedList

- The LinkedList class is a collection which can contain many objects of the same type, just like the ArrayList.
- The LinkedList class has all of the same methods as the ArrayList class because they both implement the List interface. This means that you can add items, change items, remove items and clear the list in the same way.
- However, while the ArrayList class and the LinkedList class can be used in the same way, they are built very differently.



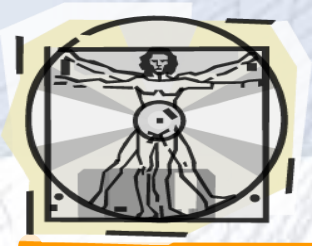
How the ArrayList/LinkedList works

- The ArrayList class has a regular array inside it. When an element is added, it is placed into the array. If the array is not big enough, a new, larger array is created to replace the old one and the old one is removed.
- The LinkedList stores its items in "containers." The list has a link to the first container and each container has a link to the next container in the list. To add an element to the list, the element is placed into a new container and that container is linked to one of the other containers in the list.



When To Use?

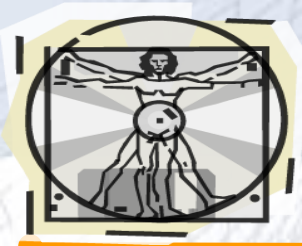
- Use an ArrayList for storing and accessing data,
- and LinkedList to manipulate data.



LinkedList Methods

- For many cases, the ArrayList is more efficient as it is common to need access to random items in the list, but the LinkedList provides several methods to do certain operations more efficiently:

- addFirst() Adds an item to the beginning of the list.
- addLast() Add an item to the end of the list
- removeFirst() Remove an item from the beginning of the list.
- removeLast() Remove an item from the end of the list
- getFirst() Get the item at the beginning of the list
- getLast() Get the item at the end of the list



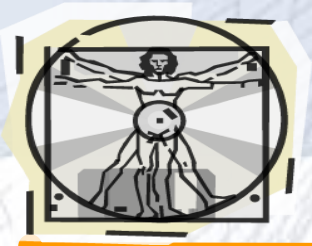
Java HashSet

A HashSet is a collection of items where every item is unique, and it is found in the `java.util` package:

Example

Create a `HashSet` object called **cars** that will store strings:

```
import java.util.HashSet;
// Import the HashSet class

HashSet<String> cars = new HashSet<String>();
```



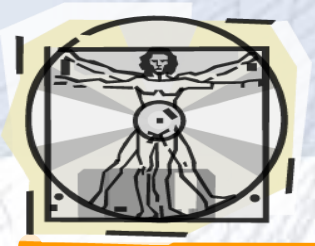
Add Items

The `HashSet` class has many useful methods. For example, to add items to it, use the `add()` method:

```
// Import the HashSet class
import java.util.HashSet;

public class Main {

public static void main(String[] args) {
    HashSet<String> cars = new HashSet<String>();
    cars.add("Volvo");
    cars.add("BMW");
    cars.add("Ford");
    cars.add("BMW");
    cars.add("Mazda");
    System.out.println(cars);
    }
}
```

```
[Volvo, Mazda, Ford, BMW]
```

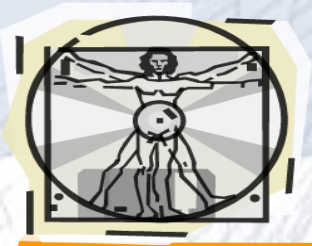
Note: In the example above, even though BMW is added twice it only appears once in the set because every item in a set has to be unique.



Check If an Item Exists

```
// Import the HashSet class
import java.util.HashSet;

public class Main {
  public static void main(String[] args) {
    HashSet<String> cars = new HashSet<String>();
    cars.add("Volvo");
    cars.add("BMW");
    cars.add("Ford");
    cars.add("BMW");
    cars.add("Mazda");
    System.out.println(cars.contains("Mazda"));
  }
}
```

```
true
```



Additional Methods

To remove an item, use the remove() method:

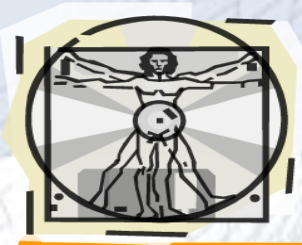
- cars.remove("Volvo");

To remove all items, use the clear() method:

- cars.clear();

To find out how many items there are, use the size method:

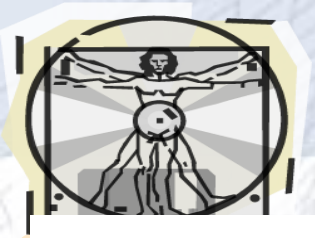
- cars.size();



Loop Through a HashSet

```
import java.util.HashSet;
public class Main {
  public static void main(String[] args) {
    HashSet<String> cars = new HashSet<String>();
    cars.add("Volvo");
    cars.add("BMW");
    cars.add("Ford");
    cars.add("BMW");
    cars.add("Mazda");
    for (String i : cars)
      System.out.println(i);
  }
}
```

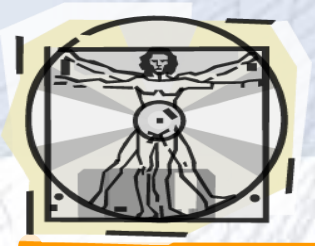
```
Volvo
Mazda
Ford
BMW
```



Other Types

```
import java.util.HashSet;

public class Main {

 public static void main(String[] args) {
   // Create a HashSet object called numbers
   HashSet<Integer> numbers = new HashSet<Integer>();
   // Add values to the set
   numbers.add(4);
   numbers.add(7);
   numbers.add(8); // Show which numbers between 1 and 10 are in the se
   for(int i = 1; i <= 10; i++) {
      if(numbers.contains(i))
         System.out.println(i + " was found in the set.");
      else
         System.out.println(i + " was not found in the set."); }
    }
}
```

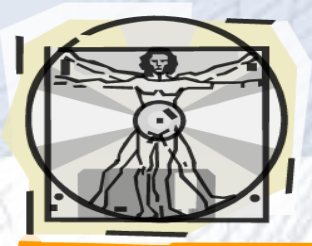
```
1 was not found in the set.
2 was not found in the set.
3 was not found in the set.
4 was found in the set.
5 was not found in the set.
6 was not found in the set.
7 was found in the set.
8 was found in the set.
9 was not found in the set.
10 was not found in the set.
```



Java HashMap

- In the ArrayList chapter, you learned that Arrays store items as an ordered collection, and you have to access them with an index number (int type). A HashMap however, store items in "key/value" pairs, and you can access them by an index of another type (e.g. a String).

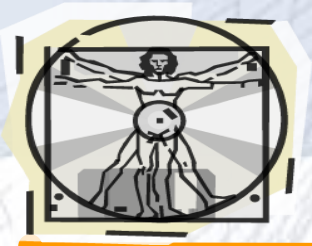
- One object is used as a key (index) to another object (value). It can store different types: String keys and Integer values, or the same type, like: String keys and String values:



Example

Create a `HashMap` object called **capitalCities** that will store `String` **keys** and `String` **values**:

```
import java.util.HashMap; // import the HashMap class

HashMap<String, String> capitalCities = new HashMap<String, String>();
```

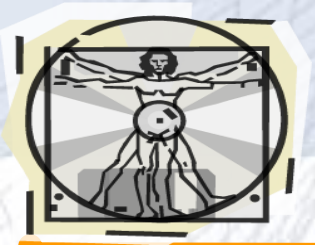


Add Items

- The HashMap class has many useful methods. For example, to add items to it, use the put() method:

```
// Import the HashMap class
import java.util.HashMap;
public class Main {
public static void main(String[] args) {
    // Create a HashMap object called capitalCities
    HashMap<String, String> capitalCities = new HashMap<String, String>();
    // Add keys and values (Country, City)
    capitalCities.put("England", "London");
    capitalCities.put("Germany", "Berlin");
    capitalCities.put("Norway", "Oslo");
    capitalCities.put("USA", "Washington DC");
    System.out.println(capitalCities);
} }
```

```
{USA=Washington DC, Norway=Oslo, England=London, Germany=Berlin}
```

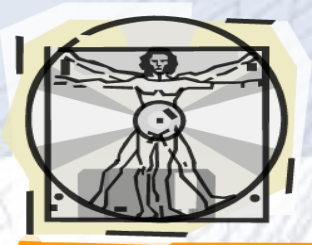


Access an Item

- To access a value in the HashMap, use the get() method and refer to its key:

```
import java.util.HashMap;

public class Main {
  public static void main(String[] args) {
    HashMap<String, String> capitalCities = new HashMap<String, String>();
    capitalCities.put("England", "London");
    capitalCities.put("Germany", "Berlin");
    capitalCities.put("Norway", "Oslo");
    capitalCities.put("USA", "Washington DC");
    System.out.println(capitalCities.get("England"));
  }
}
```

```
London
```

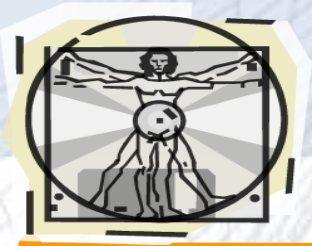


Remove an Item

- To remove an item, use the remove() method and refer to the key:

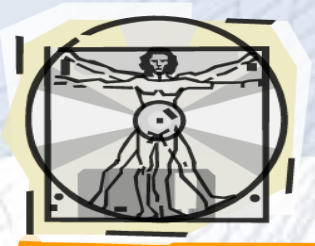
```
import java.util.HashMap;

public class Main {
  public static void main(String[] args) {
    HashMap<String, String> capitalCities = new HashMap<String, String>();
    capitalCities.put("England", "London");
    capitalCities.put("Germany", "Berlin");
    capitalCities.put("Norway", "Oslo");
    capitalCities.put("USA", "Washington DC");
    capitalCities.remove("England");
    System.out.println(capitalCities);
  }
}
```

```
{USA=Washington DC, Norway=Oslo, Germany=Berlin}
```



Addtional Methods

- To remove all items, use the clear() method:
  - capitalCities.clear();
- To find out how many items there are, use the size() method:
  - capitalCities.size();



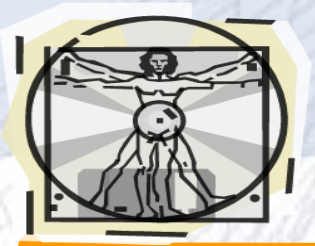
Loop Through a HashMap – keySet()

Loop through the items of a HashMap with a for-each loop.

- Note: Use the keySet() method if you only want the keys, and use the values() method if you only want the values:

```
import java.util.HashMap;
  public class Main {
    public static void main(String[] args) {
      HashMap<String, String> capitalCities = new HashMap<String, String>();
      capitalCities.put("England", "London");
      capitalCities.put("Germany", "Berlin");
      capitalCities.put("Norway", "Oslo");
      capitalCities.put("USA", "Washington DC");
      for (String i : capitalCities.keySet()) {
        System.out.println(i);     }
    }   }
```

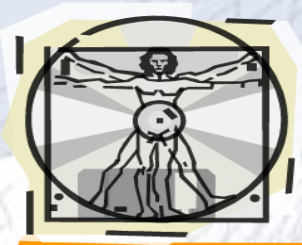
```
USA
Norway
England
Germany
```



values()

```java
import java.util.HashMap;

public class Main {
  public static void main(String[] args) {
    HashMap<String, String> capitalCities = new HashMap<String, String>();
    capitalCities.put("England", "London");
    capitalCities.put("Germany", "Berlin");
    capitalCities.put("Norway", "Oslo");
    capitalCities.put("USA", "Washington DC");

    for (String i : capitalCities.values()) {
      System.out.println(i);
    }
  }
}
```

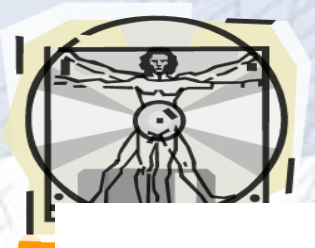
```
Washington DC
Oslo
London
Berlin
```



get(index)

```
import java.util.HashMap;

public class Main {
  public static void main(String[] args) {
    HashMap<String, String> capitalCities = new HashMap<String, String>();
    capitalCities.put("England", "London");
    capitalCities.put("Germany", "Berlin");
    capitalCities.put("Norway", "Oslo");
    capitalCities.put("USA", "Washington DC");

    for (String i : capitalCities.keySet()) {
      System.out.println("key: " + i + " value: " + capitalCities.get(i));
    }
  }
}
```

```
key: USA value: Washington DC
key: Norway value: Oslo
key: England value: London
key: Germany value: Berlin
```

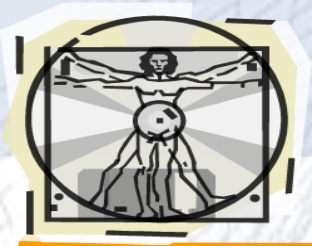


Example

Create a HashMap object called **people** that will store String **keys** and Integer **values**:

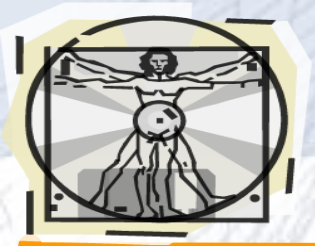
```
// Import the HashMap class
import java.util.HashMap;
public class Main {
public static void main(String[] args) {
    // Create a HashMap object called people
    HashMap<String, Integer> people = new HashMap<String, Integer>();
    // Add keys and values (Name, Age)
    people.put("John", 32);
    people.put("Steve", 30);
    people.put("Angie", 33);
    for (String i : people.keySet())
        System.out.println("key: " + i + " value: " + people.get(i));
    }
}
```

```
Name: Angie Age: 33
Name: Steve Age: 30
Name: John Age: 32
```



Example

- Write a Java program, which reads a String from the Keyboard and display the characters used in this String in the screen.
- (HashSet)
- Input
- "Biruni University of the biruni 123,!1,, ";
- Output
- [, !, B, b, e, f, h, i, ,, n, o, 1, r, 2, s, 3, t, u, U, v, y]



Example

- Write a Java program, which reads a String from the Keyboard and display the characters and their number of usage uin the screen
- (HashMap)

- Input
- "Biruni University of the biruni 123,!1,, ";
- Output
- { =6, !=1, B=1, b=1, e=2, f=1, h=1, i=6, ,=3, n=3, o=1, 1=2, r=3, 2=1, s=1, 3=1, t=2, u=2, U=1, v=1, y=1}