CSE 215: Programming Language II Lab

Sec – 8, Faculty - MUO

Lab Officer: Tanzina Tazreen

Lab – 14

ArrayList, HashMap, HashSet

Java ArrayList

The ArrayList class is a resizable array, which can be found in the java.util package. The difference between an array and an ArrayList in Java, is that the size of an array cannot be modified (if you want to add or remove elements to/from an array, you have to create a new one). While elements can be added and removed from an ArrayList whenever you want.

- Elements can be inserted or accessed by their position in the list, using a zero-based index.
- A list may contain duplicate elements.

To use an ArrayList, (like class & object) you would have to import the ArrayList class then make an object of that class. For example:

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

ArrayList<String> cars = new ArrayList<String>(); // Create an ArrayList object
```

Add Items

to add elements to the ArrayList, use the add() method.

```
import java.util.ArrayList;

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

The ArrayList class has many other useful methods:

No	objective	Description	Example
1	Access an Item	use the get() method and refer to the index number	`cars.get(0);`
2	Change an Item	use the set() method and refer to the index number with the updated value as parameter.	`cars.set(0, "Opel");`
3	Remove an Item	use the remove() method and refer to the index number	`cars.remove(0);`
4	Remove all items	use the clear() method	`cars.clear();`
5	ArrayList Size	use the size method to find out how many elements an ArrayList has	`cars.size();`
6	Sort an ArrayList	Use sort() method from Collections class of java.util package for sorting lists alphabetically or numerically	`import java.util.Collections;` `Collections.sort(cars);`

Loop Through an ArrayList

To use other Data types, such as int, you must specify an equivalent wrapper class: Integer. For other primitive types, use: Boolean for boolean, Character for char, Double for double, etc.

```
public class Main {
  public static void main(String[] args) {
    ArrayList<Integer> myNumbers = new ArrayList<Integer>();
    myNumbers.add(10);
    myNumbers.add(15);
    myNumbers.add(20);
    myNumbers.add(25);
    for (int i = 0; i < myNumbers.size(); i++) {
      System.out.println(myNumbers.get(i));
    }
  }
}
```

Java HashMap

A HashMap store items in "**key**/**value**" pairs, and you can access the value via key. Java HashMap contains only unique keys. 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.

Creating a HashMap

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

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

Here, the keys (country name) are of String type and the values (capital name) are of also String type.

Add Items

to add items to a HashMap, 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);
  }
}
```

The HashMap class has many other useful methods:

No	objective	Description	Example

1	Access an Item	use the get() method and refer to its key	`capitalCities.get("England");`
2	Change an Item	use the replace() method and refer to the key with the updated value as parameter.	`capitalCities.replace("England", “Wells”);`
3	Remove an Item	use the remove() method and refer to the key	`capitalCities.remove("England");`
4	Remove all items	use the clear() method	`capitalCities.clear();`
5	HashMap Size	use the size method to find out how many items (pairs) the HashMap has	`capitalCities.size();`
6			`capitalCities.containsKey("England")`
7			`capitalCities.containsValue("London")`

Loop Through a HashMap

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

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

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

```
// Print keys and values
```

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

Java HashSet

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

Creating HashSet

Create a HashSet object called cars that will store Strings type items

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

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

Add Items

use the add() method to add items to a HashSet

```
// 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);
  }
}
```

The HashSet class has many other useful methods:

No	objective	Description	Example
1	Check If an Item Exists	use the contains() method to check an item exists or not. It returns an Boolean value	`cars.contains("Mazda");`

2	Remove an Item	use the remove() method to remove an item	`cars.remove("Volvo");`
3	Remove all items	use the clear() method	`cars.clear();`
4	HashSet Size	use the size method to find out how many items the HashSet has	`cars.size();`

Loop Through a HashSet

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

Tasks:

1. Create an arraylist and fill it with 10 items. Then

   Check if an ArrayList contains a given element
   Find the index of the second occurrence of an element in an the ArrayList

2. Create a HashMap and fill it up with key & value pair.
   Then print the values which have a duplicate copy.