JAVA HashMap and Hash Set

Dr. Krishnendu Guha

Assistant Professor/ Lecturer

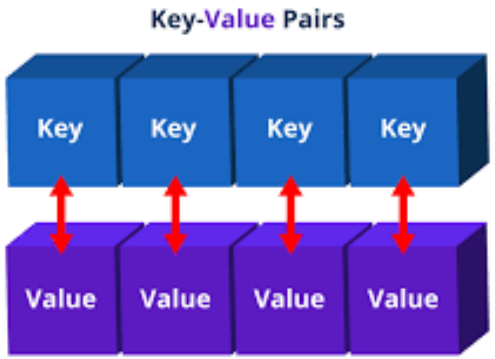
School of Computer Science and Information Technology

University College Cork

Java HashMap

In ArrayList, Arrays store items as an ordered collection, and we had to access them with an index number (int type).

A HashMap, store items in "**key**/**value**" pairs, and we 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>();
```

Adding Items to HashMap

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

Example

```
// 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:

Example

```
capitalCities.get("England");
```

```
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:

Example

```
capitalCities.remove("England");
```

```
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}
```

Remove all items

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

Example
capitalCities.clear();

```
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.clear();

    System.out.println(capitalCities);
  }
}
```

```
{}
```

HashMap Size

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

Example

```
capitalCities.size();
```

```
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.size());

  }
}
```

```
4
```

Loop Through a HashMap

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**:

Example

```
// Print keys for (String i : capitalCities.keySet()) { System.out.println(i); }
```

```
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
```

Example

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

```
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
```

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

```
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
```

Keys and values in a HashMap are actually objects.
Previously, we used objects of type "String".
Remember that a **String in Java is an object (not a primitive type).**
To use other types, such as int, we must specify an equivalent wrapper class: Integer.
For other primitive types, we use: Boolean for boolean, Character for char, Double for double, etc:

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
```

Java HashSet

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

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

```
import java.util.HashSet;
// Import the HashSet class
HashSet<String> cars = new HashSet<String>();
```

Adding items to the HashSet using 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: Though BMW is added twice it only appears once in the set because every item in a set has to be unique.

To check if an Item Exists

To check whether an item exists in a HashSet, we use the contains() method:

Example

```
cars.contains("Mazda");
```

```
// 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
```

To remove an item

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

Example

```
cars.remove("Volvo");
```

```
// 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");

    cars.remove("Volvo");

    System.out.println(cars);
  }
}
```

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

To remove all items

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

Example
cars.clear();

```
// 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");
    cars.clear();
    System.out.println(cars);  }}
```

```
[]
```

To find the HashSet size

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

Example
cars.size();

```
// 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.size());  }}
```

To loop through a HashSet

Loop through the items of an HashSet with a **for-each** loop:

Example

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

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

```
Volvo
Mazda
Ford
BMW
```

Items in an HashSet are actually objects.
Previously, we created items (objects) of type "String“, which in Java is an object (not a primitive type).
To use other types, such as int, we must specify an equivalent wrapper class: Integer.
For other primitive types, we use: Boolean for boolean, Character for char, Double for double, etc:

Example
A HashSet that stores Integer objects and shows which numbers between 1 and 10 are in the set:

```
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(9);
                // Show which numbers between 1 and 10 are in the set
                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 not found in the set.
9 was found in the set.
10 was not found in the set.
```

Thank You