

CORE JAVA - Day 56

Agenda

- Introduction to Map
- Introduction to HashMap



MAP: A map is an object that stores associations between keys and values, or key/value pairs. The Map interface maps unique keys to values. Both keys and values are objects. The Map is declared as shown here:

Interface **Map<k,v>**

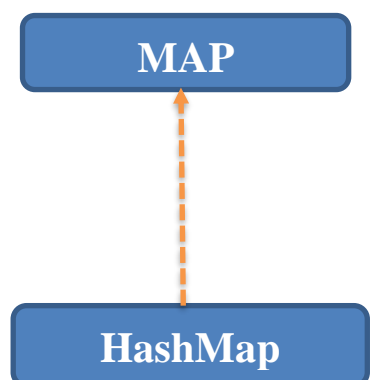
Here, K specifies the type of keys, and V specifies the type of values.

Since Map is an interface you need to instantiate a concrete implementation of the interface in order to use it. HashMap class will implement Map interface. HashMap declaration is:

HashMap<k,v>

It stores the data in (Key, Value) pairs. Inside the angular braces you have to mention the data type of key and value.

Now let's understand creation and few operations performed on HashMap with an example.



Example-1: Creating HashMap


```
import java.util.HashMap;

public class MapIntro {

    public static void main(String[] args) {
        //Create an empty HashMap
        HashMap<String, String> myDetails = new HashMap<String, String>();

        //Print the elements inside the hashMap
        System.out.println(myDetails);
    }
}
```

Output:

 Command Prompt

```
$java MapIntro.java
{}
```

Here an empty HashMap is created and no elements are added inside the HashMap so if you try to print the HashMap you are getting an empty HashMap in the output.

put(key, value) method in HashMap: It is used to insert a mapping into a map. This means we can insert a specific key and the value it is mapping to into a particular map. If an existing key is passed then the previous value gets replaced by the new value. If a new pair is passed, then the pair gets inserted as a whole. Let's now understand with examples.

Example-2: Adding the elements inside HashMap

```
import java.util.HashMap;

public class MapIntro {

    public static void main(String[] args) {
        //Create an empty HashMap
        HashMap<String, String> myDetails = new HashMap<String, String>();

        //Adding elements to the HashMap
        myDetails.put("FirstName", "Somanna");
        myDetails.put("Surname", "Somanna");
        myDetails.put("Phone number", "8965474562L");
        myDetails.put("Password", "##90$$");
        myDetails.put("DOB", "20/02/1947");
        myDetails.put("Gender", "Male");

        //Print the elements inside the hashMap
        System.out.println(myDetails);
    }
}
```

Output:

Command Prompt

```
$java MapIntro.java  
{FirstName=Somanna, DOB=20/02/1947, Phone number=8965474562L, Gender=Male, Surname=Somanna, Password=##90$$}
```

put(key, value) is used to **add** elements to the HashMap, it will accept two-parameter key and value. put(key, value) will put the elements in the HashMap only if the given key is not present inside the HashMap because duplicate keys are not allowed in the HashMap, duplicate values can be stored. **put(key,value)** will not only put the element inside the HashMap it will **return** something. If you observe the output you can clearly see that HashMap doesn't follow the insertion order. Now will see what put(key, value) will return with an example.

Example-3:

```
import java.util.HashMap;  
  
public class MapIntro {  
  
    public static void main(String[] args) {  
        //Create an empty HashMap  
        HashMap<String, String> myDetails = new HashMap<String, String>();  
  
        //Adding elements to the HashMap and print the value returned by HashMap  
        System.out.println(myDetails.put("FirstName", "Alex"));  
        System.out.println(myDetails.put("FirstName", "Somanna"));  
    }  
}
```

Output:

Command Prompt

```
$java MapIntro.java  
null  
Alex
```

In this example, the first time you are getting null as output because initially, HashMap was empty when **myDetails.put("FirstName", "Alex")** line executes it will check whether FirstName i.e given **key** is present inside the

HashMap or not as HashMap was empty it will return **null** and put FirstName and Alex inside the HashMap. When **myDetails.put("FirstName", "Somanna")** executes it will again go and check inside the HashMap whether FirstName i.e given **key** is present inside the HashMap or not, now FirstName is present inside the HashMap so it will return it's associated value i.e **Alex** and replace Somanna in place of Alex(Duplicate keys are not allowed).

Property of HashMap:

- Cannot store duplicate keys.
- Can store duplicate values.
- Doesn't maintain the insertion order.

get(key) method in HashMap: Returns the value to which the specified key is mapped, or null if this map contains no mapping for the key. Now let's understand get(key) with an example.

Example-4

```
import java.util.HashMap;

public class MapIntro {

    public static void main(String[] args) {
        //Create an empty HashMap
        HashMap<String, String> myDetails = new HashMap<String, String>();

        //Adding elements to the HashMap
        myDetails.put("FirstName", "Somanna");
        myDetails.put("Surname", "Somanna");
        myDetails.put("Phone number", "8965474562L");
        myDetails.put("Password", "##90$$");
        myDetails.put("DOB", "20/02/1947");
        myDetails.put("Gender", "Male");

        //Print the value returned by get()
        System.out.println(myDetails.get("FirstName"));

        System.out.println(myDetails.get("Lastname"));
    }
}
```

Output:

Command Prompt

```
$java MapIntro.java  
Somanna  
null
```

Here first, you are getting Somanna as the output as the key FirstName is present inside the HashMap its associated value is returned. Next, you are getting null as the output as the key LastName is not present inside the HashMap.