CORE JAVA - Day 59

Agenda

- **putIfAbsent(key,value)**
- **putAll()**
- **replace(key, value)**
- **replace(key,oldvalue,newvalue)**
- **clear()**

Example 1: If the key is not present inside the HashMap, only then put the entry inside the map.

MapIntro.java

```
import java.util.HashMap;
import java.util.Scanner;

public class MapIntro
{
    public static void main(String[] args)
    {
        HashMap<String, String> myDetails = new HashMap<String,String>();
        myDetails.put("Firstname", "Somanna");
        myDetails.put("Surname", "Somanna");
        myDetails.put("Phone number", "8965474562");
        myDetails.put("Password", "##90$$");
        myDetails.put("DOB", "20/02/1947");
        myDetails.put("Gender", "Male");

        System.out.println(myDetails);

        Scanner s = new Scanner(System.in);
        System.out.println("Enter the key:");
        String key = s.next();
        System.out.println("Enter the value");
        String value = s.next();

        if(myDetails.containsKey(key)==false) {
            myDetails.put(key, value);
        }
        System.out.println(myDetails);
    }
}
```

Output:

```
$java MapIntro.java
{Firstname=Somanna, DOB=20/02/1947, Phone number=8965474562, Gender=Male, Surname=Somanna, Password=##90$$}
Enter the key:
Country
Enter the value
India
{Firstname=Somanna, DOB=20/02/1947, Phone number=8965474562, Country=India, Gender=Male, Surname=Somanna, Password=##90$$}
```

In the above program **myDetails.containsKey(key) == false** will check whether the key is present inside the HashMap or not, if it is not present only then it will put key value pair inside the HashMap. You can observe from the output also, Country is not present inside the hashmap that is when country=India key-value pair is put into the hashmap. There is an efficient way to check and put the entry inside the HashMap i.e, using **putIfAbsent()** as shown below.

MapIntro.java

```
import java.util.HashMap;
import java.util.Scanner;

public class MapIntro
{
    public static void main(String[] args)
    {
        HashMap<String, String> myDetails = new HashMap<String,String>();
        myDetails.put("Firstname", "Somanna");
        myDetails.put("Surname", "Somanna");
        myDetails.put("Phone number", "8965474562");
        myDetails.put("Password", "##90$$");
        myDetails.put("DOB", "20/02/1947");
        myDetails.put("Gender", "Male");

        System.out.println(myDetails);

        Scanner s = new Scanner(System.in);
        System.out.println("Enter the key:");
        String key = s.next();
        System.out.println("Enter the value");
        String value = s.next();

        myDetails.putIfAbsent(key, value);
        System.out.println(myDetails);
    }
}
```

Output:

```
$java MapIntro.java
{Firstname=Somanna, DOB=20/02/1947, Phone number=8965474562, Gender=Male, Surname=Somanna, Password=##90$$}
Enter the key:
Country
Enter the value
India
{Firstname=Somanna, DOB=20/02/1947, Phone number=8965474562, Country=India, Gender=Male, Surname=Somanna, Password=##90$$}
```

If you observe the output, you are getting same output when putIfAbsent() is used. The **putIfAbsent(key K, value V)** method of HashMap is used to map the specified key with the specified value, only if no such key exists in the HashMap instance.

Example 2: Write a program where you have to put all the elements present in one hashmap to another hashmap.

MapIntro.java

```
import java.util.HashMap;
import java.util.Set;

public class MapIntro {
    public static void main(String[] args) {
        HashMap<String, String> myDetails = new HashMap<String,String>();
        myDetails.put("Firstname", "Somanna");
        myDetails.put("Surname", "Somanna");
        myDetails.put("Phone number", "8965474562");
        myDetails.put("Password", "##90$$");
        myDetails.put("DOB", "20/02/1947");
        myDetails.put("Gender", "Male");

        HashMap<String, String> myData = new HashMap<String, String>();
        myData.put("Email", "Somanna@gmail.com");
        myData.put("Country", "India");
        myData.put("Blood Group", "O+");

        System.out.println(myDetails);

        Set<String> keys = myData.keySet();

        for(String key : keys) {
            String value = myData.get(key);
            myDetails.put(key, value);
        }
        System.out.println(myDetails);
    }
}
```

Output:

```
$java MapIntro.java
{Firstname=Somanna, DOB=20/02/1947, Phone number=8965474562, Gender=Male, Surname=Somanna, Password=##9
0$$}
{Firstname=Somanna, Blood Group=O+, Email=Somanna@gmail.com, DOB=20/02/1947, Phone number=8965474562, C
ountry=India, Gender=Male, Surname=Somanna, Password=##90$$}
```

You can see from the above example, to put all the elements from one hashmap to another hashmap first we are using keyset() method using which we are fetching all the keys present inside myDetails HashMap and storing it inside the set. Now using for-each loop value of each key is fetched and put that key value pair inside the myDetails HashMap. There is an efficient way to do the same using putAll() as shown below.

MapIntro.java

```
import java.util.HashMap;

public class MapIntro {
    public static void main(String[] args) {
        HashMap<String, String> myDetails = new HashMap<String,String>();
        myDetails.put("Firstname", "Somanna");
        myDetails.put("Surname", "Somanna");
        myDetails.put("Phone number", "8965474562");
        myDetails.put("Password", "##90$$");
        myDetails.put("DOB", "20/02/1947");
        myDetails.put("Gender", "Male");

        HashMap<String, String> myData = new HashMap<String, String>();
        myData.put("Email", "Somanna@gmail.com");
        myData.put("Country", "India");
        myData.put("Blood Group", "O+");

        System.out.println(myDetails);

        myDetails.putAll(myData);
        System.out.println(myDetails);
    }
}
```

Output:

```
$java MapIntro.java
{Firstname=Somanna, DOB=20/02/1947, Phone number=8965474562, Gender=Male, Surname=Somanna, Password=##9
0$$}
{Firstname=Somanna, Blood Group=O+, Email=Somanna@gmail.com, DOB=20/02/1947, Phone number=8965474562, C
ountry=India, Gender=Male, Surname=Somanna, Password=##90$$}
```

As you can see from the output, using putAll() you are getting same expected output. **putAll()** Copies all of the mappings from the specified map to this map. These mappings will replace any mappings that this map had for any of the keys currently in the specified map.

Example 3: Write a Program to replace the value of a specified key

MapIntro.java

```
import java.util.HashMap;

public class MapIntro {
    public static void main(String[] args) {
        HashMap<String, String> myDetails = new HashMap<String,String>();
        myDetails.put("Firstname", "Somanna");
        myDetails.put("Surname", "Somanna");
        myDetails.put("Phone number", "8965474562");
        myDetails.put("Password", "##90$$");
        myDetails.put("DOB", "20/02/1947");
        myDetails.put("Gender", "Male");

        System.out.println(myDetails);

        myDetails.replace("Password", "abcdef");

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

Output:

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

As you can clearly see from the output, initially the password was ##90$$ using replace() we have changed the password to abcdef. **replace(key, value)** will replaces the entry for the specified key only if it is currently mapped to some value.

One more way of replacing the value is if the specified key-value is present then replace the new value with old value, if the specified key-value pair is not present then HashMap remains as it is. Now we will understand this with an example.

MapIntro.java

```
import java.util.HashMap;

public class MapIntro {
    public static void main(String[] args) {
        HashMap<String, String> myDetails = new HashMap<String,String>();
        myDetails.put("Firstname", "Somanna");
        myDetails.put("Surname", "Somanna");
        myDetails.put("Phone number", "8965474562");
        myDetails.put("Password", "##90$$");
        myDetails.put("DOB", "20/02/1947");
        myDetails.put("Gender", "Male");

        System.out.println(myDetails);

        myDetails.replace("Gender", "Male", "Other");

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

Output:

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

You can see from the above example that, Male i.e old value is replaced with other i.e new value as Gender and Male as a key-value pair is present inside the HashMap. **replace(key, OldValue, NewValue)** will replaces the entry for the specified key only if currently mapped to the specified value.

Example 4: Remove all the elements present inside the HashMap

```
import java.util.HashMap;

public class MapIntro {
    public static void main(String[] args) {
        HashMap<String, String> myDetails = new HashMap<String,String>();
        myDetails.put("Firstname", "Somanna");
        myDetails.put("Surname", "Somanna");
        myDetails.put("Phone number", "8965474562");
        myDetails.put("Password", "##90$$");
        myDetails.put("DOB", "20/02/1947");
        myDetails.put("Gender", "Male");

        System.out.println(myDetails);

        myDetails.clear();

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

Output:

```
$java MapIntro.java
{Firstname=Somanna, DOB=20/02/1947, Phone number=8965474562, Gender=Male, Surname=Somanna, Password=##9
0$$}
{}
```

As you can see from the above example, using clear() you can remove all the entry present inside the HashMap. **clear()** will remove all of the mappings from this map. The map will be empty after this call returns.