

## Lesson 05 Demo 05

### Implementing HashMap and Hashtable

**Objective:** To demonstrate how to use HashMap, LinkedHashMap, and Hashtable in Java for efficient data handling

**Tools Required:** Eclipse IDE

**Prerequisites:** None

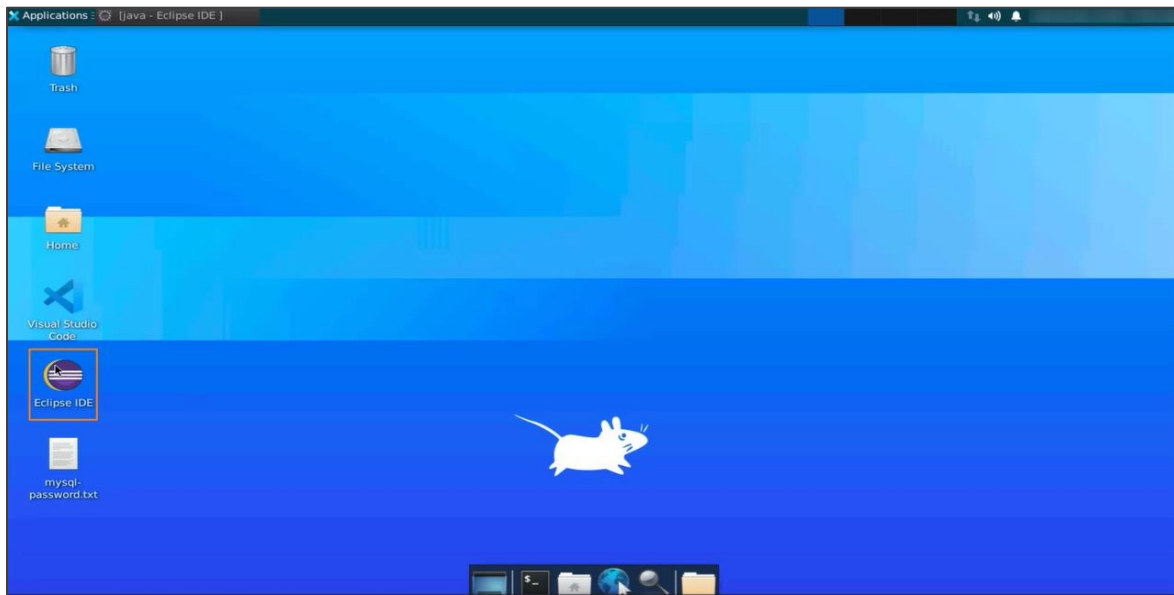
Steps to be followed:

1. Create a new project
2. Use a HashMap
3. Execute the code with example data
4. Sort the data based on keys
5. Iterate the data structure with example data
6. Use the remove() method and execute the code
7. Implement iteration to obtain all the keys from the map
8. Execute the entrySet() method and iterate through the code

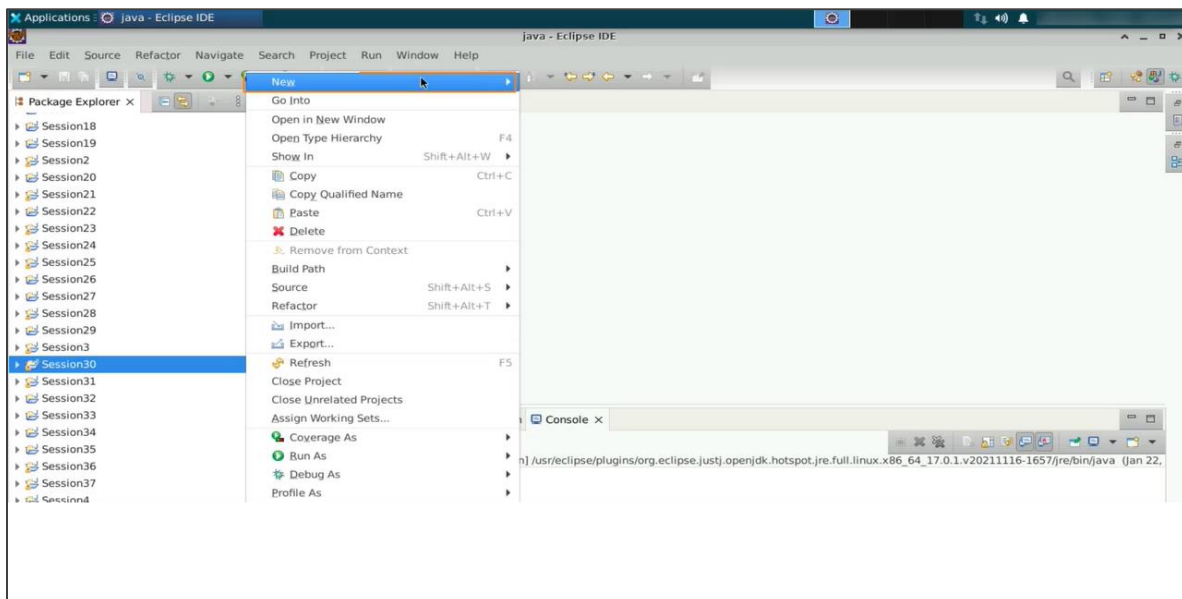
## Step 1: Create a new project

1.1 In the Collections framework, you will explore the usage of a data structure called Map, which stores data as key-value pairs. It is an essential data structure.

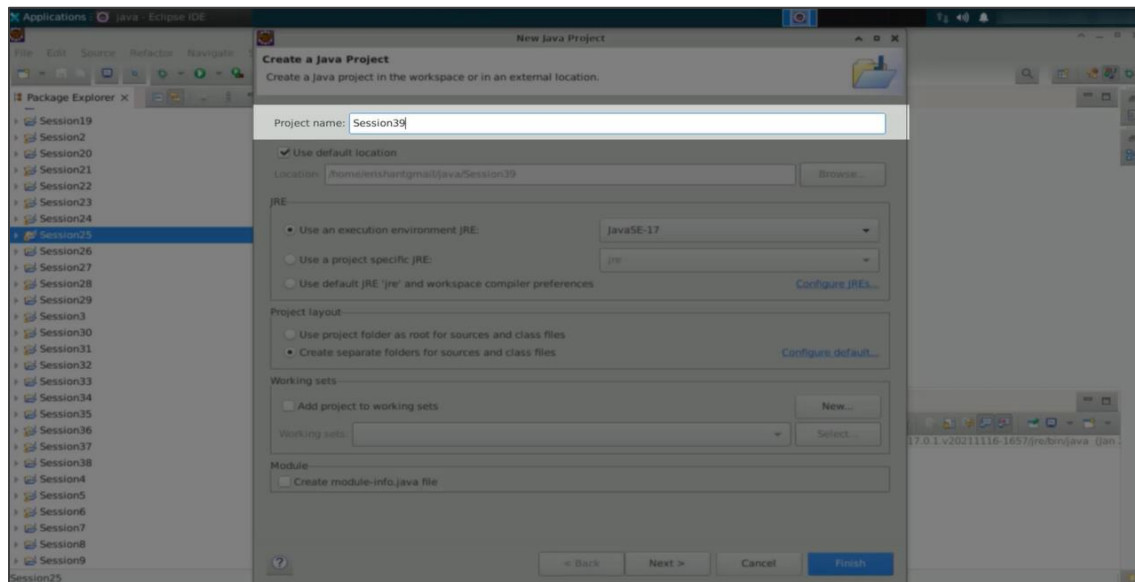
To create a HashMap, a LinkedHashMap, and a Hashtable, open the **Eclipse IDE**.



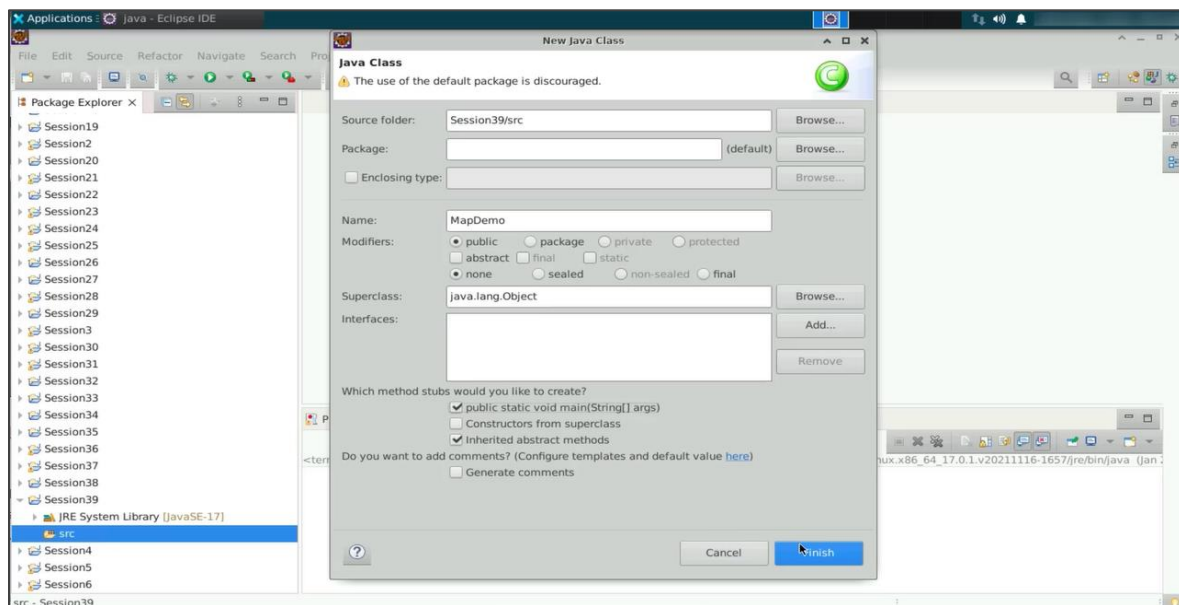
## 1.2 Create a new Java project



### 1.3 Name the new project **Session39**

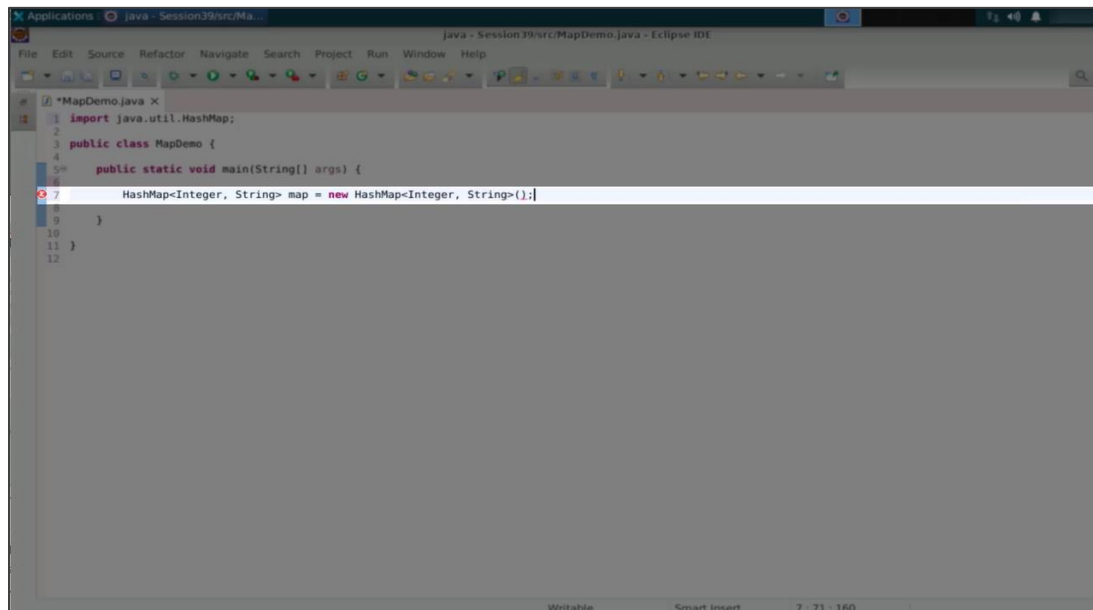


### 1.4 Right-click on the source folder and create a new class called **MapDemo** with the main method



## Step 2: Use a HashMap

### 2.1 Create a new HashMap called `map`

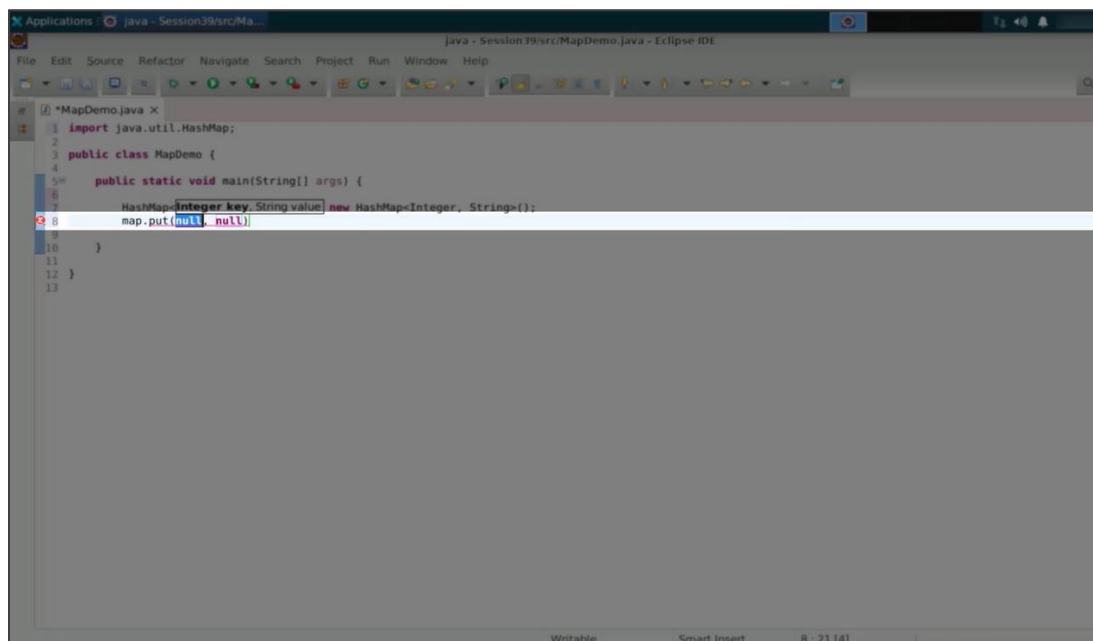


The screenshot shows the Eclipse IDE with a Java project named 'Session39/src/MapDemo.java'. The code in the editor is as follows:

```
1 import java.util.HashMap;
2
3 public class MapDemo {
4
5     public static void main(String[] args) {
6
7         HashMap<Integer, String> map = new HashMap<Integer, String>();
8     }
9 }
10
11
12
```

The status bar at the bottom indicates 'Writable', 'Smart Insert', and '7 : 71 : 160'.

### 2.2 In the map, instead of using the add method, we use the `put` method

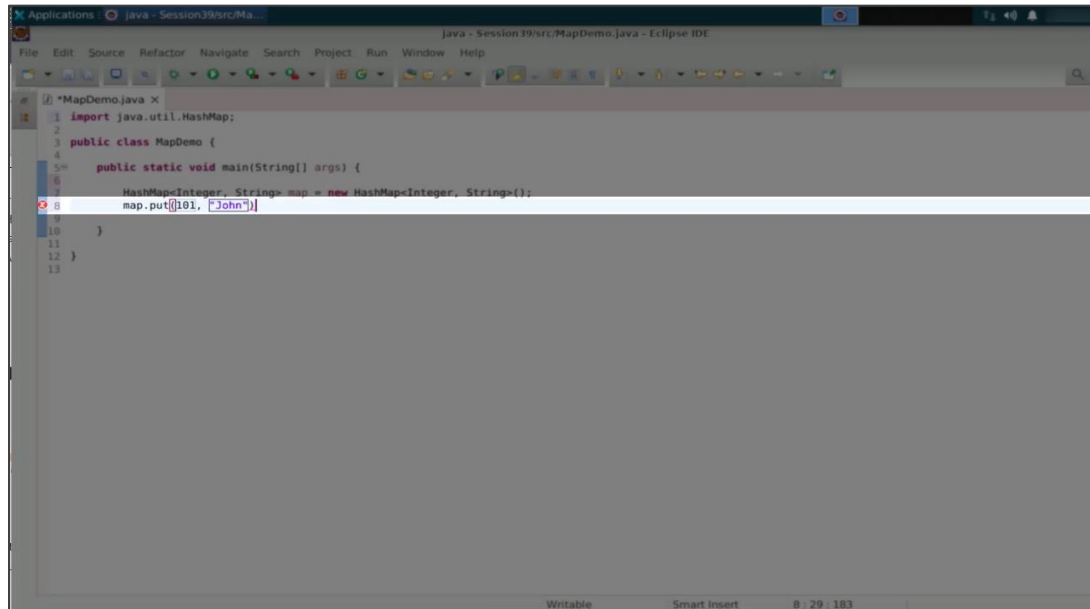


The screenshot shows the Eclipse IDE with the same Java project. The code in the editor is updated to use the `put` method:

```
1 import java.util.HashMap;
2
3 public class MapDemo {
4
5     public static void main(String[] args) {
6
7         HashMap<Integer key, String value> new HashMap<Integer, String>();
8         map.put(null, null);
9     }
10 }
11
12
13
```

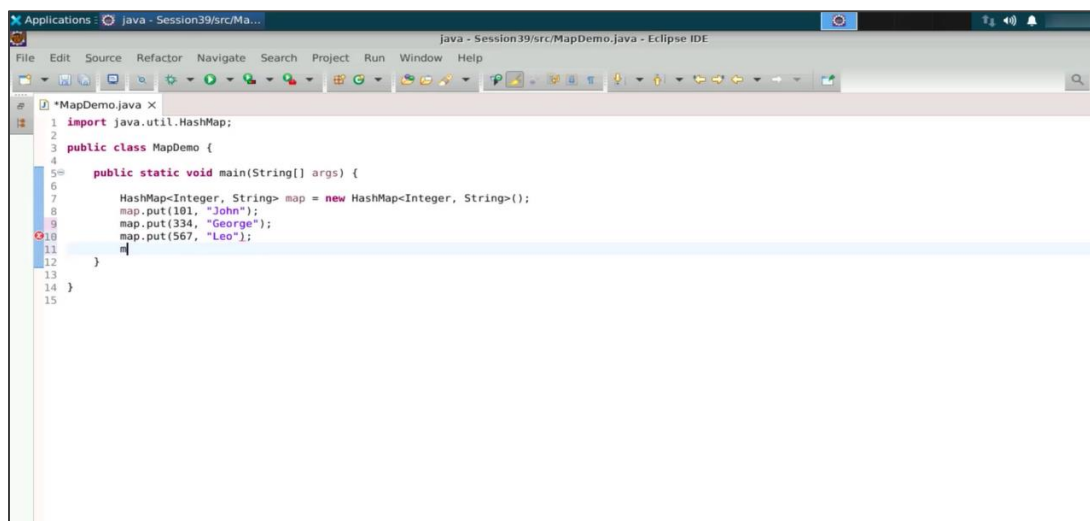
The status bar at the bottom indicates 'Writable', 'Smart Insert', and '8 : 21 [4]'.

2.3 To store data inside the HashMap, use the put method. For instance, assign the key **101** and the value **John**



```
1 import java.util.HashMap;
2
3 public class MapDemo {
4
5     public static void main(String[] args) {
6
7         HashMap<Integer, String> map = new HashMap<Integer, String>();
8         map.put(101, "John");
9     }
10 }
11
12
13
```

2.4 You can add more keys and records using the **map.put()** method

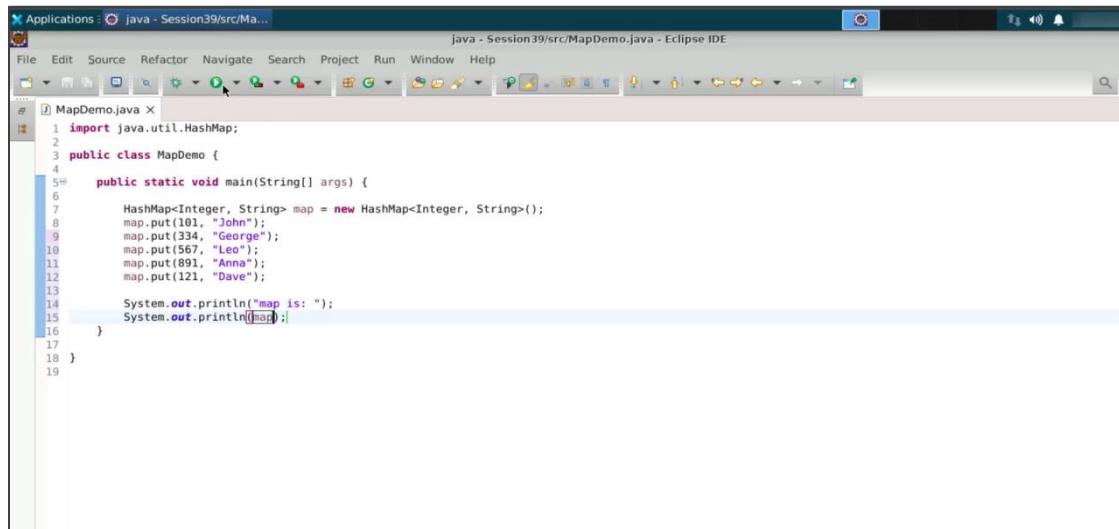


```
1 import java.util.HashMap;
2
3 public class MapDemo {
4
5     public static void main(String[] args) {
6
7         HashMap<Integer, String> map = new HashMap<Integer, String>();
8         map.put(101, "John");
9         map.put(334, "George");
10        map.put(567, "Leo");
11    }
12 }
13
14
15
```

It is important to ensure that the key is unique while the value can be duplicated. Like a HashSet, a HashMap allows you to work with uniqueness based on the keys but not the values.

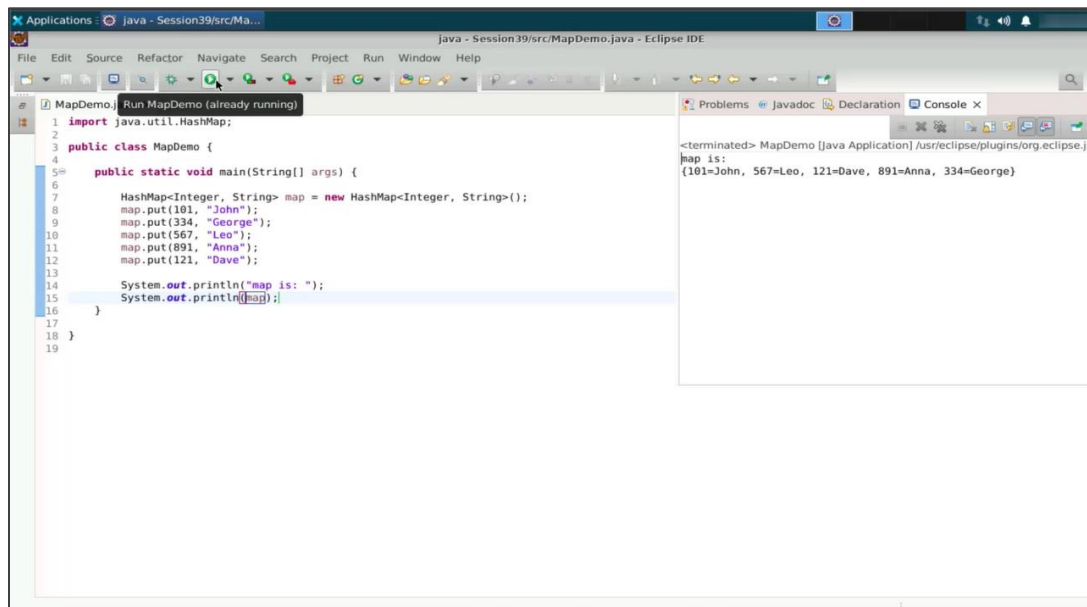
## Step 3: Execute the code with example data

3.1 To read the data from the map, print "map is: " followed by the map itself using the print statement



```
1 import java.util.HashMap;
2
3 public class MapDemo {
4
5     public static void main(String[] args) {
6
7         HashMap<Integer, String> map = new HashMap<Integer, String>();
8         map.put(101, "John");
9         map.put(334, "George");
10        map.put(567, "Leo");
11        map.put(891, "Anna");
12        map.put(121, "Dave");
13
14        System.out.println("map is: ");
15        System.out.println(map);
16    }
17 }
18
19 }
```

3.2 When you run the code, you will observe the data displayed, such as **John, Leo, Dave, Anna, and George**



```
1 import java.util.HashMap;
2
3 public class MapDemo {
4
5     public static void main(String[] args) {
6
7         HashMap<Integer, String> map = new HashMap<Integer, String>();
8         map.put(101, "John");
9         map.put(334, "George");
10        map.put(567, "Leo");
11        map.put(891, "Anna");
12        map.put(121, "Dave");
13
14        System.out.println("map is: ");
15        System.out.println(map);
16    }
17 }
18
19 }
```

Console Output:

```
<terminated> MapDemo [Java Application] /usr/eclipse/plugins/org.eclipse.jdt
map is:
{101=John, 567=Leo, 121=Dave, 891=Anna, 334=George}
```

The data is available in the form of key-value pairs.

3.3 You can add new data using the **map.put()** method. If you try to add a duplicate key, it will update the corresponding value.

```

1 import java.util.HashMap;
2
3 public class MapDemo {
4
5     public static void main(String[] args) {
6
7         HashMap<Integer, String> map = new HashMap<Integer, String>();
8         map.put(101, "John");
9         map.put(334, "George");
10        map.put(567, "Leo");
11        map.put(891, "Anna");
12        map.put(121, "Dave");
13
14        map.put(334, "Mike");
15
16        System.out.println("map is: ");
17        System.out.println(map);
18    }
19 }
20 }
21

```

```

<terminated> MapDemo [Java Application] /usr/eclipse/plugins/org.eclipse.j
map is:
{101=John, 567=Leo, 121=Dave, 891=Anna, 334=Mike}

```

3.4 You can add null as a key or a value in the HashMap

```

1 import java.util.HashMap;
2
3 public class MapDemo {
4
5     public static void main(String[] args) {
6
7         HashMap<Integer, String> map = new HashMap<Integer, String>();
8         map.put(101, "John");
9         map.put(334, "George");
10        map.put(567, "Leo");
11        map.put(891, "Anna");
12        map.put(121, "Dave");
13
14        // Insert and update the record in the Map
15        map.put(334, "Mike");
16
17        map.put(null, "Sia");
18
19        System.out.println("map is: ");
20        System.out.println(map);
21    }
22 }
23 }
24

```

```

java - Session39/src/Ma...
Java - Session39/src/MapDemo.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
java X
java.util.HashMap;
class MapDemo {
public static void main(String[] args) {
    HashMap<Integer, String> map = new HashMap<Integer, String>();
    map.put(101, "John");
    map.put(334, "George");
    map.put(567, "Leo");
    map.put(891, "Anna");
    map.put(121, "Dave");

    // insert and update the record in the Map
    map.put(334, "Mike");

    map.put(null, "Sia");
    map.put(777, null);

    System.out.println("map is: ");
    System.out.println(map);
}
}

```

```

<terminated> MapDemo [Java Application] /usr/eclipse/plugins/org.eclipse.justi.openjdk.hc
map is:
{null=Sia, 101=John, 567=Leo, 121=Dave, 777=null, 891=Anna, 334=Mike}

```

Duplicate values and multiple null values are allowed, but duplicate keys are not.

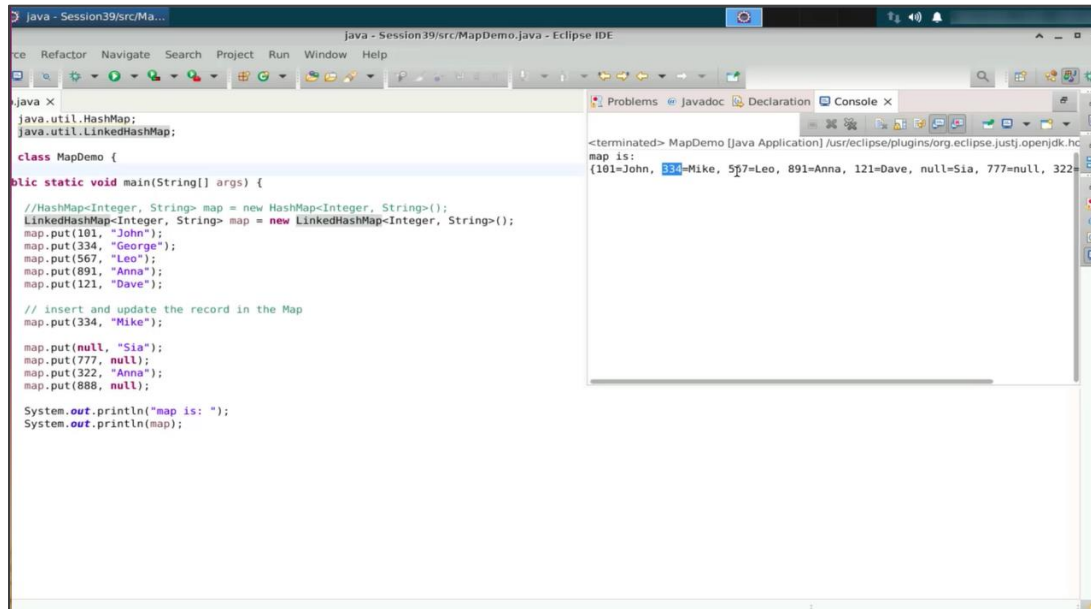
3.5 As you advance, you can use LinkedHashMap to maintain the order of insertion.

```

Applications : java - Session39/src/Ma...
Java - Session39/src/MapDemo.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
MapDemo.java X
1 import java.util.HashMap;
2 import java.util.LinkedHashMap;
3
4 public class MapDemo {
5
6     public static void main(String[] args) {
7
8         //HashMap<Integer, String> map = new HashMap<Integer, String>();
9         LinkedHashMap<Integer, String> map = new LinkedHashMap<Integer, String>();
10        map.put(101, "John");
11        map.put(334, "George");
12        map.put(567, "Leo");
13        map.put(891, "Anna");
14        map.put(121, "Dave");
15
16        // insert and update the record in the Map
17        map.put(334, "Mike");
18
19        map.put(null, "Sia");
20        map.put(777, null);
21        map.put(322, "Anna");
22        map.put(888, null);
23
24        System.out.println("map is: ");
25        System.out.println(map);
26    }
27 }
28 }
29

```

3.6 When you run the code, you will notice that the data is displayed in the same order in which you added it



The screenshot shows the Eclipse IDE interface. The left pane displays the source code for `MapDemo.java`. The code imports `java.util.HashMap` and `java.util.LinkedHashMap`, and defines a `MapDemo` class with a `main` method. The `main` method creates a `HashMap` and a `LinkedHashMap`, then populates them with key-value pairs. The `LinkedHashMap` is created with `new LinkedHashMap<Integer, String>()`. The key-value pairs added are: (101, "John"), (334, "George"), (567, "Leo"), (891, "Anna"), (121, "Dave"), (334, "Mike"), (null, "Sia"), (777, null), (322, "Anna"), and (888, null). The `main` method prints the map using `System.out.println("map is: ");` and `System.out.println(map);`. The right pane shows the console output, which displays the map in the order of insertion: `{101=John, 334=Mike, 567=Leo, 891=Anna, 121=Dave, null=Sia, 777=null, 322=Anna, 888=null}`.

```
java - Session39/src/MapDemo.java - Eclipse IDE
ce Refactor Navigate Search Project Run Window Help

java X
java.util.HashMap;
java.util.LinkedHashMap;

class MapDemo {

    public static void main(String[] args) {

        //HashMap<Integer, String> map = new HashMap<Integer, String>();
        LinkedHashMap<Integer, String> map = new LinkedHashMap<Integer, String>();
        map.put(101, "John");
        map.put(334, "George");
        map.put(567, "Leo");
        map.put(891, "Anna");
        map.put(121, "Dave");

        // insert and update the record in the Map
        map.put(334, "Mike");

        map.put(null, "Sia");
        map.put(777, null);
        map.put(322, "Anna");
        map.put(888, null);

        System.out.println("map is: ");
        System.out.println(map);
    }
}

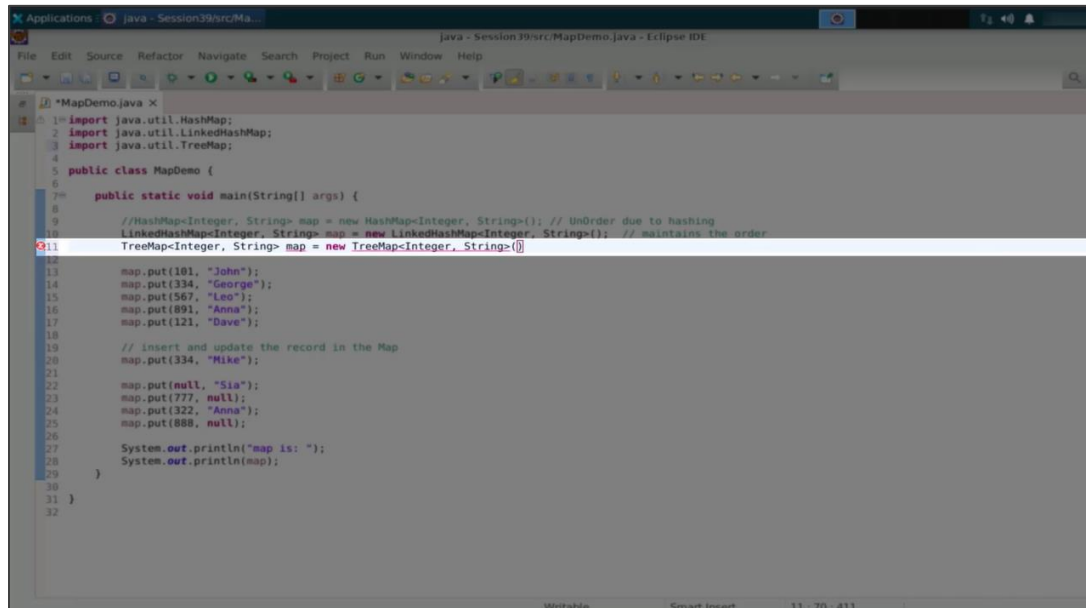
Problems Javadoc Declaration Console X

<terminated> MapDemo [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hc
map is:
{101=John, 334=Mike, 567=Leo, 891=Anna, 121=Dave, null=Sia, 777=null, 322=Anna, 888=null}
```

For example, if you added the keys **101**, **334**, **567**, and **891**, the data will be shown in that exact order. This is because `LinkedHashMap` maintains the order of insertion.

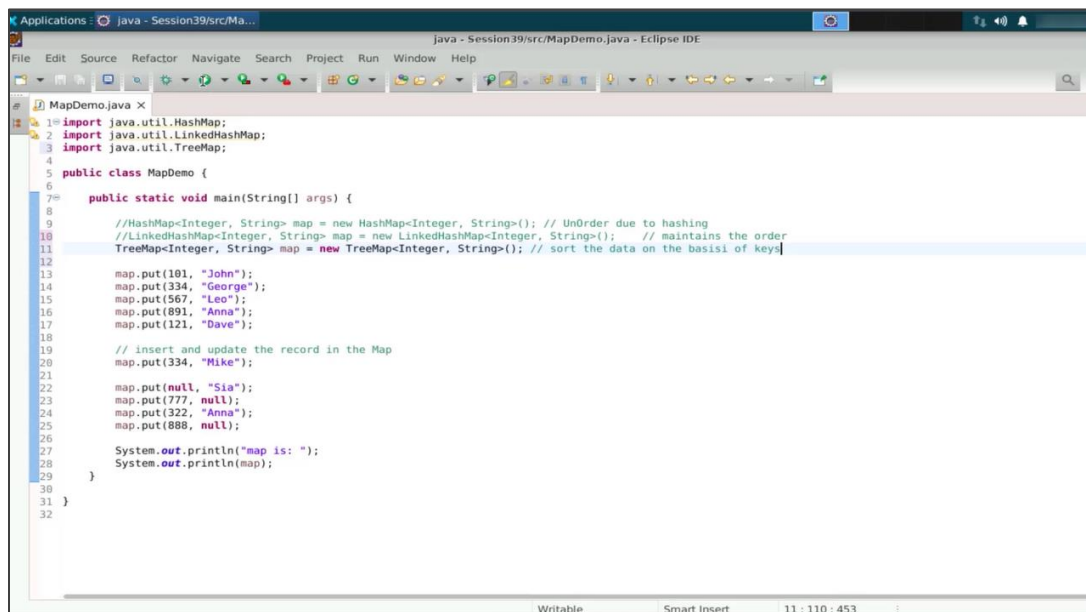
## Step 4: Sort the data based on keys

### 4.1 Create a new TreeMap



```
1 import java.util.HashMap;
2 import java.util.LinkedHashMap;
3 import java.util.TreeMap;
4
5 public class MapDemo {
6
7     public static void main(String[] args) {
8
9         //HashMap<Integer, String> map = new HashMap<Integer, String>(); // UnOrder due to hashing
10        //LinkedHashMap<Integer, String> map = new LinkedHashMap<Integer, String>(); // maintains the order
11        TreeMap<Integer, String> map = new TreeMap<Integer, String>();
12
13        map.put(101, "John");
14        map.put(334, "George");
15        map.put(567, "Leo");
16        map.put(891, "Anna");
17        map.put(121, "Dave");
18
19        // insert and update the record in the Map
20        map.put(334, "Mike");
21
22        map.put(null, "Sia");
23        map.put(777, null);
24        map.put(322, "Anna");
25        map.put(888, null);
26
27        System.out.println("map is: ");
28        System.out.println(map);
29    }
30 }
31
32
```

### 4.2 Add a comment indicating that this TreeMap will sort the data based on keys



```
1 import java.util.HashMap;
2 import java.util.LinkedHashMap;
3 import java.util.TreeMap;
4
5 public class MapDemo {
6
7     public static void main(String[] args) {
8
9         //HashMap<Integer, String> map = new HashMap<Integer, String>(); // UnOrder due to hashing
10        //LinkedHashMap<Integer, String> map = new LinkedHashMap<Integer, String>(); // maintains the order
11        TreeMap<Integer, String> map = new TreeMap<Integer, String>(); // sort the data on the basis of keys
12
13        map.put(101, "John");
14        map.put(334, "George");
15        map.put(567, "Leo");
16        map.put(891, "Anna");
17        map.put(121, "Dave");
18
19        // insert and update the record in the Map
20        map.put(334, "Mike");
21
22        map.put(null, "Sia");
23        map.put(777, null);
24        map.put(322, "Anna");
25        map.put(888, null);
26
27        System.out.println("map is: ");
28        System.out.println(map);
29    }
30 }
31
32
```

### 4.3 Resolve the null key issue

```

java - Session39/src/Ma...
java - Session39/src/MapDemo.java - Eclipse IDE
File Edit Refactor Navigate Search Project Run Window Help

java X
java.util.HashMap;
java.util.LinkedHashMap;
java.util.TreeMap;

class MapDemo {

public static void main(String[] args) {

//HashMap<Integer, String> map = new HashMap<Integer, String>(); // UnOrder due to h
//LinkedHashMap<Integer, String> map = new LinkedHashMap<Integer, String>(); // m
TreeMap<Integer, String> map = new TreeMap<Integer, String>(); // sort the data on ti

map.put(101, "John");
map.put(334, "George");
map.put(567, "Leo");
map.put(891, "Anna");
map.put(121, "Dave");

// insert and update the record in the Map
map.put(334, "Mike");

map.put(null, "Sia");
map.put(777, null);
map.put(322, "Anna");
map.put(888, null);

System.out.println("map is: ");
System.out.println(map);
}
}

```

Problems Javadoc Declaration Console X

```

<terminated> MapDemo [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full/bin/linux64/eclipse-justj.jar
Exception in thread "main" java.lang.NullPointerException
at java.base/java.util.Objects.requireNonNull(Objects.java:208)
at java.base/java.util.TreeMap.put(TreeMap.java:809)
at java.base/java.util.TreeMap.put(TreeMap.java:534)
at MapDemo.main(MapDemo.java:22)

```

### 4.4 Run the code and observe the sorted data based on keys

```

java - Session39/src/Ma...
java - Session39/src/MapDemo.java - Eclipse IDE
File Edit Refactor Navigate Search Project Run Window Help

java X
java.util.HashMap;
java.util.LinkedHashMap;
java.util.TreeMap;

class MapDemo {

public static void main(String[] args) {

//HashMap<Integer, String> map = new HashMap<Integer, String>(); // UnOrder due to h
//LinkedHashMap<Integer, String> map = new LinkedHashMap<Integer, String>(); // m
TreeMap<Integer, String> map = new TreeMap<Integer, String>(); // sort the data on ti

map.put(101, "John");
map.put(334, "George");
map.put(567, "Leo");
map.put(891, "Anna");
map.put(121, "Dave");

// insert and update the record in the Map
map.put(334, "Mike");

map.put(null, "Sia");
map.put(777, null);
map.put(322, "Anna");
map.put(888, null);

System.out.println("map is: ");
System.out.println(map);
}
}

```

Problems Javadoc Declaration Console X

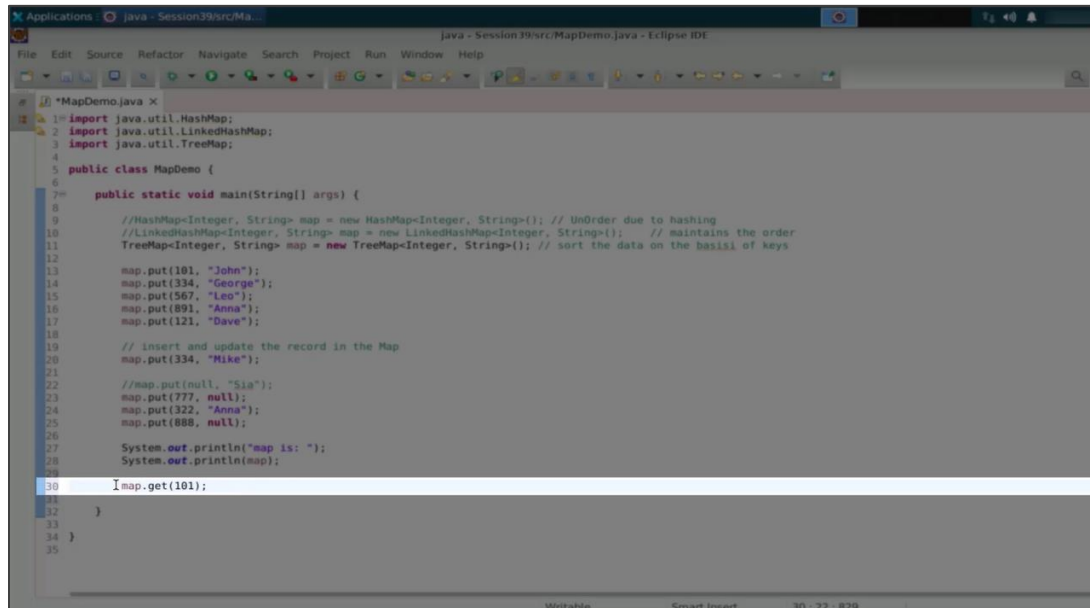
```

<terminated> MapDemo [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full/bin/linux64/eclipse-justj.jar
[101=John, 121=Dave, 322=Anna, 334=Mike, 567=Leo, 777=null, 888=null, 8=1]

```

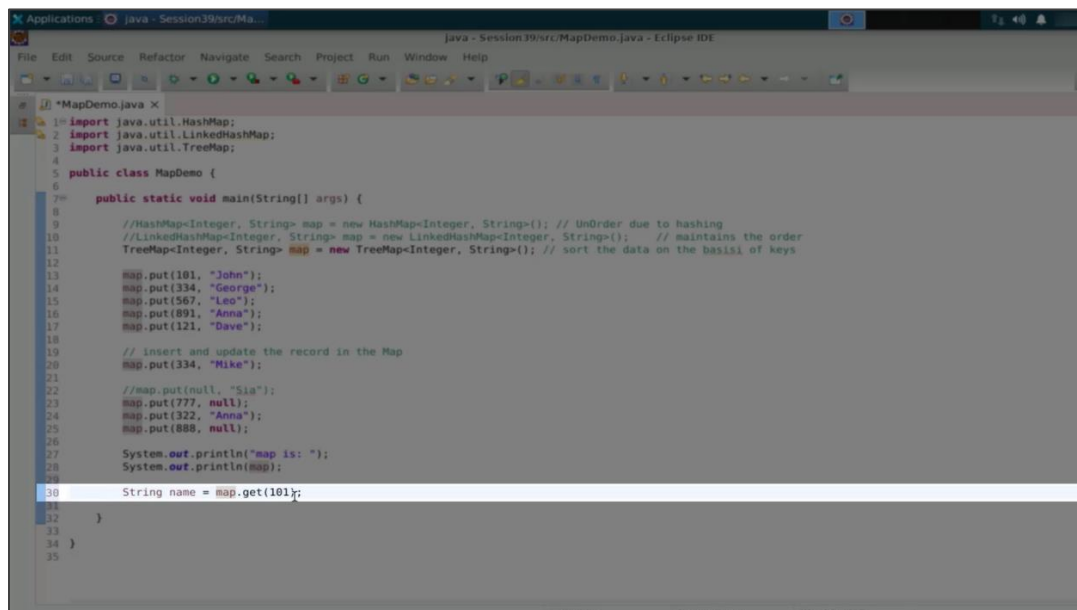
## Step 5: Iterate the data structure with example data

### 5.1 Execute built-in methods on the map such as `map.get(101)`



```
1 import java.util.HashMap;
2 import java.util.LinkedHashMap;
3 import java.util.TreeMap;
4
5 public class MapDemo {
6
7     public static void main(String[] args) {
8
9         //HashMap<Integer, String> map = new HashMap<Integer, String>(); // UnOrder due to hashing
10        //LinkedHashMap<Integer, String> map = new LinkedHashMap<Integer, String>(); // maintains the order
11        TreeMap<Integer, String> map = new TreeMap<Integer, String>(); // sort the data on the basis of keys
12
13        map.put(101, "John");
14        map.put(334, "George");
15        map.put(567, "Leo");
16        map.put(891, "Anna");
17        map.put(121, "Dave");
18
19        // Insert and update the record in the Map
20        map.put(334, "Mike");
21
22        //map.put(null, "Sia");
23        map.put(777, null);
24        map.put(322, "Anna");
25        map.put(888, null);
26
27        System.out.println("map is: ");
28        System.out.println(map);
29
30        map.get(101);
31    }
32 }
33
34 }
35 }
```

### 5.2 Assign the retrieved value to a variable named `name`: `(String name = map.get(101))`



```
1 import java.util.HashMap;
2 import java.util.LinkedHashMap;
3 import java.util.TreeMap;
4
5 public class MapDemo {
6
7     public static void main(String[] args) {
8
9         //HashMap<Integer, String> map = new HashMap<Integer, String>(); // UnOrder due to hashing
10        //LinkedHashMap<Integer, String> map = new LinkedHashMap<Integer, String>(); // maintains the order
11        TreeMap<Integer, String> map = new TreeMap<Integer, String>(); // sort the data on the basis of keys
12
13        map.put(101, "John");
14        map.put(334, "George");
15        map.put(567, "Leo");
16        map.put(891, "Anna");
17        map.put(121, "Dave");
18
19        // Insert and update the record in the Map
20        map.put(334, "Mike");
21
22        //map.put(null, "Sia");
23        map.put(777, null);
24        map.put(322, "Anna");
25        map.put(888, null);
26
27        System.out.println("map is: ");
28        System.out.println(map);
29
30        String name = map.get(101);
31    }
32 }
33
34 }
35 }
```

### 5.3 Print the value of name: ("name is: " + name)

```

1 import java.util.HashMap;
2 import java.util.LinkedHashMap;
3 import java.util.TreeMap;
4
5 public class MapDemo {
6
7     public static void main(String[] args) {
8
9         //HashMap<Integer, String> map = new HashMap<Integer, String>(); // UnOrder due to hashing
10        //LinkedHashMap<Integer, String> map = new LinkedHashMap<Integer, String>(); // maintains the order
11        TreeMap<Integer, String> map = new TreeMap<Integer, String>(); // sort the data on the basis of keys
12
13        map.put(101, "John");
14        map.put(334, "George");
15        map.put(567, "Leo");
16        map.put(891, "Anna");
17        map.put(121, "Dave");
18
19        // insert and update the record in the Map
20        map.put(334, "Mike");
21
22        //map.put(null, "Sia");
23        map.put(777, null);
24        map.put(322, "Anna");
25        map.put(888, null);
26
27        System.out.println("map is: ");
28        System.out.println(map);
29
30        String name = map.get(101);
31        System.out.println("name is: " + name);
32
33    }
34
35 }
36

```

### 5.4 Run the code and observe the printed name

```

<terminated> MapDemo [Java Application] /usr/eclipse/plugins/org.eclipse.justi.openjdk.hotspot.jre.full/bin/linux64/java
map is:
{101=John, 121=Dave, 322=Anna, 334=Mike, 567=Leo, 777=null, 888=null, 891=Anna}
name is: John

```

## 5.5 Repeat steps 9.1-9.4 with different keys to retrieve corresponding values

```

1 import java.util.HashMap;
2 import java.util.LinkedHashMap;
3 import java.util.TreeMap;
4
5 public class MapDemo {
6
7     public static void main(String[] args) {
8
9         //HashMap<Integer, String> map = new HashMap<Integer, String>(); // UnOrder due to hashing
10        //LinkedHashMap<Integer, String> map = new LinkedHashMap<Integer, String>(); // maintains the order
11        TreeMap<Integer, String> map = new TreeMap<Integer, String>(); // sort the data on the basis of keys
12
13        map.put(101, "John");
14        map.put(334, "George");
15        map.put(567, "Leo");
16        map.put(891, "Anna");
17        map.put(121, "Dave");
18
19        // insert and update the record in the Map
20        map.put(334, "Mike");
21
22        //map.put(null, "Sia");
23        map.put(777, null);
24        map.put(322, "Anna");
25        map.put(888, null);
26
27        System.out.println("map is: ");
28        System.out.println(map);
29
30        String name = map.get(121);
31        System.out.println("name is: " + name);
32
33    }
34 }
35
36

```

```

<terminated> MapDemo [Java Application] /usr/eclipse/plugins/org.eclipse.jdt
map is:
{101=John, 121=Dave, 322=Anna, 334=Mike, 567=Leo, 777=null, 888=null}
name is: Dave

```

## 5.6 Use `map.containsKey(567)` to check if the key 567 exists

```

1 import java.util.HashMap;
2 import java.util.LinkedHashMap;
3 import java.util.TreeMap;
4
5 public class MapDemo {
6
7     public static void main(String[] args) {
8
9         //HashMap<Integer, String> map = new HashMap<Integer, String>(); // UnOrder due to hashing
10        //LinkedHashMap<Integer, String> map = new LinkedHashMap<Integer, String>(); // maintains the order
11        TreeMap<Integer, String> map = new TreeMap<Integer, String>(); // sort the data on the basis of keys
12
13        map.put(101, "John");
14        map.put(334, "George");
15        map.put(567, "Leo");
16        map.put(891, "Anna");
17        map.put(121, "Dave");
18
19        // insert and update the record in the Map
20        map.put(334, "Mike");
21
22        //map.put(null, "Sia");
23        map.put(777, null);
24        map.put(322, "Anna");
25        map.put(888, null);
26
27        System.out.println("map is: ");
28        System.out.println(map);
29
30        String name = map.get(121);
31        System.out.println("name is: " + name);
32
33        if(map.containsKey(567))
34        {
35
36        }
37    }
38 }

```

Syntax error on token "}". Statement expected after this token

## 5.7 Print 567 is in the map if the key is present

```

1 import java.util.HashMap;
2 import java.util.LinkedHashMap;
3 import java.util.TreeMap;
4
5 public class MapDemo {
6
7     public static void main(String[] args) {
8
9         //HashMap<Integer, String> map = new HashMap<Integer, String>(); // UnOrder due to hashing
10        //LinkedHashMap<Integer, String> map = new LinkedHashMap<Integer, String>(); // maintains the order
11        TreeMap<Integer, String> map = new TreeMap<Integer, String>(); // sort the data on the basis of keys
12
13        map.put(181, "John");
14        map.put(334, "George");
15        map.put(567, "Leo");
16        map.put(891, "Anna");
17        map.put(121, "Dave");
18
19        // insert and update the record in the Map
20        map.put(334, "Mike");
21
22        //map.put(null, "Sia");
23        map.put(777, null);
24        map.put(322, "Anna");
25        map.put(888, null);
26
27        System.out.println("map is: ");
28        System.out.println(map);
29
30        String name = map.get(121);
31        System.out.println("name is: "+name);
32
33        if(map.containsKey(567)) {
34            System.out.println("567 is in the map");
35        }
36    }
37 }
38

```

## 5.8 Use map.containsValue("Dave") to check if the value Dave exists

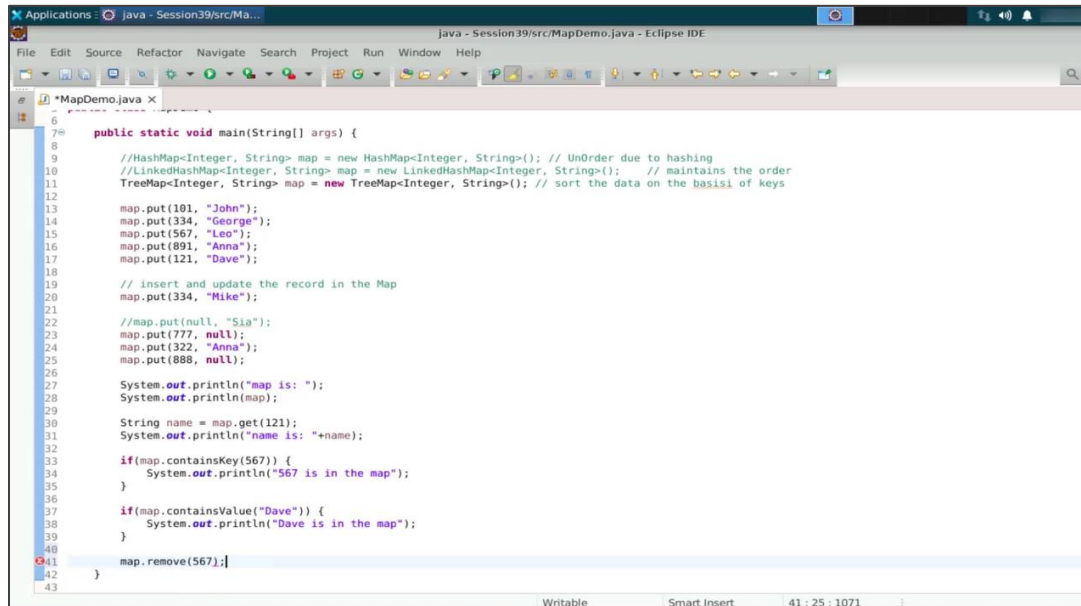
```

6
7     public static void main(String[] args) {
8
9         //HashMap<Integer, String> map = new HashMap<Integer, String>(); // UnOrder due to hashing
10        //LinkedHashMap<Integer, String> map = new LinkedHashMap<Integer, String>(); // maintains the order
11        TreeMap<Integer, String> map = new TreeMap<Integer, String>(); // sort the data on the basis of keys
12
13        map.put(181, "John");
14        map.put(334, "George");
15        map.put(567, "Leo");
16        map.put(891, "Anna");
17        map.put(121, "Dave");
18
19        // insert and update the record in the Map
20        map.put(334, "Mike");
21
22        //map.put(null, "Sia");
23        map.put(777, null);
24        map.put(322, "Anna");
25        map.put(888, null);
26
27        System.out.println("map is: ");
28        System.out.println(map);
29
30        String name = map.get(121);
31        System.out.println("name is: "+name);
32
33        if(map.containsKey(567)) {
34            System.out.println("567 is in the map");
35        }
36
37        if(map.containsValue("Dave")) {
38            System.out.println("Dave is in the map");
39        }
40    }
41 }
42
43

```

## Step 6: Use the remove() method and execute the code

### 6.1 Use map.remove(567) to remove a key-value pair

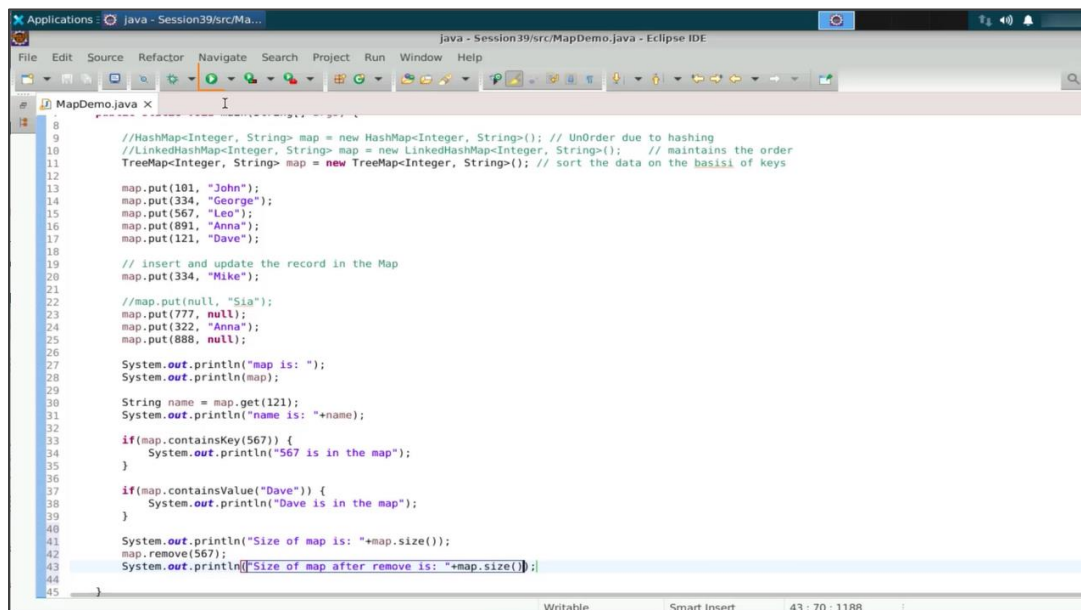


```

6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
public static void main(String[] args) {
    //HashMap<Integer, String> map = new HashMap<Integer, String>(); // UnOrder due to hashing
    //LinkedHashMap<Integer, String> map = new LinkedHashMap<Integer, String>(); // maintains the order
    TreeMap<Integer, String> map = new TreeMap<Integer, String>(); // sort the data on the basis of keys
    map.put(101, "John");
    map.put(334, "George");
    map.put(567, "Leo");
    map.put(891, "Anna");
    map.put(121, "Dave");
    // insert and update the record in the Map
    map.put(334, "Mike");
    //map.put(null, "Sia");
    map.put(777, null);
    map.put(322, "Anna");
    map.put(888, null);
    System.out.println("map is: ");
    System.out.println(map);
    String name = map.get(121);
    System.out.println("name is: "+name);
    if(map.containsKey(567)) {
        System.out.println("567 is in the map");
    }
    if(map.containsValue("Dave")) {
        System.out.println("Dave is in the map");
    }
    map.remove(567);
}

```

### 6.2 Print the size of the map before and after using remove():



```

8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
public static void main(String[] args) {
    //HashMap<Integer, String> map = new HashMap<Integer, String>(); // UnOrder due to hashing
    //LinkedHashMap<Integer, String> map = new LinkedHashMap<Integer, String>(); // maintains the order
    TreeMap<Integer, String> map = new TreeMap<Integer, String>(); // sort the data on the basis of keys
    map.put(101, "John");
    map.put(334, "George");
    map.put(567, "Leo");
    map.put(891, "Anna");
    map.put(121, "Dave");
    // insert and update the record in the Map
    map.put(334, "Mike");
    //map.put(null, "Sia");
    map.put(777, null);
    map.put(322, "Anna");
    map.put(888, null);
    System.out.println("map is: ");
    System.out.println(map);
    String name = map.get(121);
    System.out.println("name is: "+name);
    if(map.containsKey(567)) {
        System.out.println("567 is in the map");
    }
    if(map.containsValue("Dave")) {
        System.out.println("Dave is in the map");
    }
    System.out.println("Size of map is: "+map.size());
    map.remove(567);
    System.out.println("Size of map after remove is: "+map.size());
}

```

### 6.3 Run the code and observe the size change

```

8
9 //HashMap<Integer, String> map = new HashMap<Integer, String>(); // UnOrder due to h
10 //LinkedHashMap<Integer, String> map = new LinkedHashMap<Integer, String>(); // m
11 //TreeMap<Integer, String> map = new TreeMap<Integer, String>(); // sort the data on t
12
13 map.put(101, "John");
14 map.put(334, "George");
15 map.put(567, "Leo");
16 map.put(891, "Anna");
17 map.put(121, "Dave");
18
19 // insert and update the record in the Map
20 map.put(334, "Mike");
21
22 //map.put(null, "Sia");
23 map.put(777, null);
24 map.put(322, "Anna");
25 map.put(888, null);
26
27 System.out.println("map is: ");
28 System.out.println(map);
29
30 String name = map.get(121);
31 System.out.println("name is: "+name);
32
33 if(map.containsKey(567)) {
34     System.out.println("567 is in the map");
35 }
36
37 if(map.containsValue("Dave")) {
38     System.out.println("Dave is in the map");
39 }
40
41 System.out.println("Size of map is: "+map.size());
42 map.remove(567);
43 System.out.println("Size of map after remove is: "+map.size());
44
45

```

```

<terminated> MapDemo [Java Application] /usr/eclipse/plugins/org.eclipse.ja
map is:
{101=John, 121=Dave, 322=Anna, 334=Mike, 567=Leo, 777=null, 888=
name is: Dave
567 is in the map
Dave is in the map
Size of map is: 8
Size of map after remove is: 7

```

## Step 7: Implement iteration to obtain all the keys from the map

7.1 Obtain all the keys from the map using **map.keySet()** and assign the keys to a set:

**Set<Integer> keys = map.keySet();**

```

12
13
14 map.put(101, "John");
15 map.put(334, "George");
16 map.put(567, "Leo");
17 map.put(891, "Anna");
18 map.put(121, "Dave");
19
20 // insert and update the record in the Map
21 map.put(334, "Mike");
22
23 //map.put(null, "Sia");
24 map.put(777, null);
25 map.put(322, "Anna");
26 map.put(888, null);
27
28 System.out.println("map is: ");
29 System.out.println(map);
30
31 String name = map.get(121);
32 System.out.println("name is: "+name);
33
34 if(map.containsKey(567)) {
35     System.out.println("567 is in the map");
36 }
37
38 if(map.containsValue("Dave")) {
39     System.out.println("Dave is in the map");
40 }
41
42 System.out.println("Size of map is: "+map.size());
43 map.remove(567);
44 System.out.println("Size of map after remove is: "+map.size());
45
46 Set<Integer> keys = map.keySet();
47
48
49

```

## 7.2 Print the keys: ("Keys in the map are: " + keys);

```

12  TreeMap<Integer, String> map = new TreeMap<Integer, String>(); // sort the data on the basis of keys
13
14  map.put(101, "John");
15  map.put(334, "George");
16  map.put(567, "Leo");
17  map.put(891, "Anna");
18  map.put(121, "Dave");
19
20  // Insert and update the record in the Map
21  map.put(334, "Mike");
22
23  //map.put(null, "Sia");
24  map.put(777, null);
25  map.put(322, "Anna");
26  map.put(888, null);
27
28  System.out.println("map is: ");
29  System.out.println(map);
30
31  String name = map.get(121);
32  System.out.println("name is: "+name);
33
34  if(map.containsKey(567)) {
35      System.out.println("567 is in the map");
36  }
37
38  if(map.containsValue("Dave")) {
39      System.out.println("Dave is in the map");
40  }
41
42  System.out.println("Size of map is: "+map.size());
43  map.remove(567);
44  System.out.println("Size of map after remove is: "+map.size());
45
46  Set<Integer> keys = map.keySet();
47  System.out.println("Keys in map are: "+keys);
48
49  }

```

## 7.3 Run the code and observe the printed keys

```

12  TreeMap<Integer, String> map = new TreeMap<Integer, String>(); // sort the data on ti
13
14  map.put(101, "John");
15  map.put(334, "George");
16  map.put(567, "Leo");
17  map.put(891, "Anna");
18  map.put(121, "Dave");
19
20  // Insert and update the record in the Map
21  map.put(334, "Mike");
22
23  //map.put(null, "Sia");
24  map.put(777, null);
25  map.put(322, "Anna");
26  map.put(888, null);
27
28  System.out.println("map is: ");
29  System.out.println(map);
30
31  String name = map.get(121);
32  System.out.println("name is: "+name);
33
34  if(map.containsKey(567)) {
35      System.out.println("567 is in the map");
36  }
37
38  if(map.containsValue("Dave")) {
39      System.out.println("Dave is in the map");
40  }
41
42  System.out.println("Size of map is: "+map.size());
43  map.remove(567);
44  System.out.println("Size of map after remove is: "+map.size());
45
46  Set<Integer> keys = map.keySet();
47  System.out.println("Keys in map are: "+keys);
48
49  }

```

<terminated> MapDemo [Java Application] /usr/eclipse/plugins/org.eclipse.jdt.launcher/
 map is:
 {101=John, 121=Dave, 322=Anna, 334=Mike, 567=Leo, 777=null, 888=
 name is: Dave
 Dave is in the map
 567 is in the map
 Size of map is: 8
 Size of map after remove is: 7
 Keys in map are: [101, 121, 322, 334, 777, 888, 891]

## 7.4 Create an iterator for the keys: `Iterator<Integer> itr = keys.iterator();`

```

18  map.put(891, "Anna");
19  map.put(121, "Dave");
20
21  // Insert and update the record in the Map
22  map.put(334, "Mike");
23
24  //map.put(null, "Sia");
25  map.put(777, null);
26  map.put(322, "Anna");
27  map.put(888, null);
28
29  System.out.println("map is: ");
30  System.out.println(map);
31
32  String name = map.get(121);
33  System.out.println("name is: "+name);
34
35  if(map.containsKey(567)) {
36      System.out.println("567 is in the map");
37  }
38
39  if(map.containsValue("Dave")) {
40      System.out.println("Dave is in the map");
41  }
42
43  System.out.println("Size of map is: "+map.size());
44  map.remove(567);
45  System.out.println("Size of map after remove is: "+map.size());
46
47  Set<Integer> keys = map.keySet();
48  System.out.println("Keys in map are: "+keys);
49
50  Iterator<Integer> itr = keys.iterator();
51
52  }
53
54  }
55

```

## 7.5 Use a while loop with `itr.hasNext()` to iterate over the keys

```

18  map.put(891, "Anna");
19  map.put(121, "Dave");
20
21  // Insert and update the record in the Map
22  map.put(334, "Mike");
23
24  //map.put(null, "Sia");
25  map.put(777, null);
26  map.put(322, "Anna");
27  map.put(888, null);
28
29  System.out.println("map is: ");
30  System.out.println(map);
31
32  String name = map.get(121);
33  System.out.println("name is: "+name);
34
35  if(map.containsKey(567)) {
36      System.out.println("567 is in the map");
37  }
38
39  if(map.containsValue("Dave")) {
40      System.out.println("Dave is in the map");
41  }
42
43  System.out.println("Size of map is: "+map.size());
44  map.remove(567);
45  System.out.println("Size of map after remove is: "+map.size());
46
47  Set<Integer> keys = map.keySet();
48  System.out.println("Keys in map are: "+keys);
49
50  Iterator<Integer> itr = keys.iterator();
51  while(itr.hasNext()) {
52      |
53  }
54
55  }

```

## 7.6 Retrieve the corresponding value using map.get()

```

18  map.put(891, "Anna");
19  map.put(121, "Dave");
20
21  // insert and update the record in the Map
22  map.put(334, "Mike");
23
24  //map.put(null, "Sia");
25  map.put(777, null);
26  map.put(322, "Anna");
27  map.put(888, null);
28
29  System.out.println("map is: ");
30  System.out.println(map);
31
32  String name = map.get(121);
33  System.out.println("name is: "+name);
34
35  if(map.containsKey(567)) {
36      System.out.println("567 is in the map");
37  }
38
39  if(map.containsValue("Dave")) {
40      System.out.println("Dave is in the map");
41  }
42
43  System.out.println("Size of map is: "+map.size());
44  map.remove(567);
45  System.out.println("Size of map after remove is: "+map.size());
46
47  Set<Integer> keys = map.keySet();
48  System.out.println("Keys in map are: "+keys);
49
50  Iterator<Integer> itr = keys.iterator();
51  while(itr.hasNext()) {
52      Integer key = itr.next();
53      String value = map.get(key);
54  }
55

```

## 7.7 Print the key-value pairs

```

22  map.put(334, "Mike");
23
24  //map.put(null, "Sia");
25  map.put(777, null);
26  map.put(322, "Anna");
27  map.put(888, null);
28
29  System.out.println("map is: ");
30  System.out.println(map);
31
32  String name = map.get(121);
33  System.out.println("name is: "+name);
34
35  if(map.containsKey(567)) {
36      System.out.println("567 is in the map");
37  }
38
39  if(map.containsValue("Dave")) {
40      System.out.println("Dave is in the map");
41  }
42
43  System.out.println("Size of map is: "+map.size());
44  map.remove(567);
45  System.out.println("Size of map after remove is: "+map.size());
46
47  Set<Integer> keys = map.keySet();
48  System.out.println("Keys in map are: "+keys);
49
50  Iterator<Integer> itr = keys.iterator();
51  while(itr.hasNext()) {
52      Integer key = itr.next();
53      String value = map.get(key);
54      System.out.println(key+" \t "+value);
55  }
56
57  }
58
59

```

## 7.8 Run the code and observe the printed key-value pairs

The screenshot shows the Eclipse IDE with a Java project named 'Session39'. The file 'MapDemo.java' is open, showing code that creates a HashMap, adds several key-value pairs, and then prints them out. The console on the right shows the output of the program, including the size of the map before and after removing a key, and the keys in the map.

```

22 map.put(334, "Mike");
23
24 //map.put(null, "Sia");
25 map.put(777, null);
26 map.put(322, "Anna");
27 map.put(888, null);
28
29 System.out.println("map is: ");
30 System.out.println(map);
31
32 String name = map.get(121);
33 System.out.println("name is: "+name);
34
35 if(map.containsKey(567)) {
36     System.out.println("567 is in the map");
37 }
38
39 if(map.containsValue("Dave")) {
40     System.out.println("Dave is in the map");
41 }
42
43 System.out.println("Size of map is: "+map.size());
44 map.remove(567);
45 System.out.println("Size of map after remove is: "+map.size());
46
47 Set<Integer> keys = map.keySet();
48 System.out.println("Keys in map are: "+keys);
49
50 Iterator<Integer> itr = keys.iterator();
51 while(itr.hasNext()) {
52     Integer key = itr.next();
53     String value = map.get(key);
54     System.out.println(key+" \t "+value);
55 }
56
57
58
59

```

Console Output:

```

<terminated> MapDemo [Java Application] /usr/eclipse/plugins/org.eclipse.j
map is:
{101=John, 121=Dave, 322=Anna, 334=Mike, 567=Leo, 777=null, 888=
name is: Dave
567 is in the map
Dave is in the map
Size of map is: 8
Size of map after remove is: 7
Keys in map are: [101, 121, 322, 334, 777, 888, 891]
101 John
121 Dave
322 Anna
334 Mike
777 null
888 null
891 Anna

```

## Step 8: Execute the entrySet() method and iterate through the code

### 8.1 Obtain the entry set using map.entrySet()

The screenshot shows the same Eclipse IDE with 'MapDemo.java'. The code is identical to the previous one, but with an additional line at the bottom (line 59) that creates a Set of Map.Entry objects using the 'entrySet()' method. The IDE also shows a status bar at the bottom indicating that the local variable 'set' is not used.

```

59 Set<Entry<Integer, String>> set = map.entrySet();

```

Status bar: The value of the local variable set is not used

## 8.2 Assign the entry set to a variable:

**Set<Map.Entry<Integer, String>> entrySet = map.entrySet();**

```

27  map.put(322, "Anna");
28  map.put(888, null);
29
30  System.out.println("map is: ");
31  System.out.println(map);
32
33  String name = map.get(121);
34  System.out.println("name is: "+name);
35
36  if(map.containsKey(567)) {
37      System.out.println("567 is in the map");
38  }
39
40  if(map.containsValue("Dave")) {
41      System.out.println("Dave is in the map");
42  }
43
44  System.out.println("Size of map is: "+map.size());
45  map.remove(567);
46  System.out.println("Size of map after remove is: "+map.size());
47
48  Set<Integer> keys = map.keySet();
49  System.out.println("Keys in map are: "+keys);
50
51  Iterator<Integer> itr = keys.iterator();
52  while(itr.hasNext()) {
53      Integer key = itr.next();
54      String value = map.get(key);
55      System.out.println(key+" \t "+value);
56  }
57
58  Set<Entry<Integer, String>> set = map.entrySet();
59
60  }
61  }
62  }
63  }
64  }

```

## 8.3 Create an iterator for the entry set:

**Iterator<Map.Entry<Integer, String>> itr1 = entrySet.iterator();**

```

27  map.put(322, "Anna");
28  map.put(888, null);
29
30  System.out.println("map is: ");
31  System.out.println(map);
32
33  String name = map.get(121);
34  System.out.println("name is: "+name);
35
36  if(map.containsKey(567)) {
37      System.out.println("567 is in the map");
38  }
39
40  if(map.containsValue("Dave")) {
41      System.out.println("Dave is in the map");
42  }
43
44  System.out.println("Size of map is: "+map.size());
45  map.remove(567);
46  System.out.println("Size of map after remove is: "+map.size());
47
48  Set<Integer> keys = map.keySet();
49  System.out.println("Keys in map are: "+keys);
50
51  Iterator<Integer> itr = keys.iterator();
52  while(itr.hasNext()) {
53      Integer key = itr.next();
54      String value = map.get(key);
55      System.out.println(key+" \t "+value);
56  }
57
58  Set<Entry<Integer, String>> set = map.entrySet();
59  Iterator<Entry<Integer, String>> itr1 = set.iterator();
60
61  }
62  }
63  }
64  }

```

## 8.4 Use a while loop with itr1.hasNext()

```

31  System.out.println(map);
32
33  String name = map.get(121);
34  System.out.println("name is: "+name);
35
36  if(map.containsKey(567)) {
37      System.out.println("567 is in the map");
38  }
39
40  if(map.containsValue("Dave")) {
41      System.out.println("Dave is in the map");
42  }
43
44  System.out.println("Size of map is: "+map.size());
45  map.remove(567);
46  System.out.println("Size of map after remove is: "+map.size());
47
48  Set<Integer> keys = map.keySet();
49  System.out.println("Keys in map are: "+keys);
50
51  Iterator<Integer> itr = keys.iterator();
52  while(itr.hasNext()) {
53      Integer key = itr.next();
54      String value = map.get(key);
55      System.out.println(key+" \t "+value);
56  }
57
58  Set<Entry<Integer, String>> set = map.entrySet();
59  Iterator<Entry<Integer, String>> itr1 = set.iterator();
60
61  while(itr1.hasNext()) {
62      }
63  }
64
65  }
66
67  }
68

```

## 8.5 Retrieve each entry Map:

**Entry<Integer, String> entry = itr1.next();**

```

31  System.out.println(map);
32
33  String name = map.get(121);
34  System.out.println("name is: "+name);
35
36  if(map.containsKey(567)) {
37      System.out.println("567 is in the map");
38  }
39
40  if(map.containsValue("Dave")) {
41      System.out.println("Dave is in the map");
42  }
43
44  System.out.println("Size of map is: "+map.size());
45  map.remove(567);
46  System.out.println("Size of map after remove is: "+map.size());
47
48  Set<Integer> keys = map.keySet();
49  System.out.println("Keys in map are: "+keys);
50
51  Iterator<Integer> itr = keys.iterator();
52  while(itr.hasNext()) {
53      Integer key = itr.next();
54      String value = map.get(key);
55      System.out.println(key+" \t "+value);
56  }
57
58  Set<Entry<Integer, String>> set = map.entrySet();
59  Iterator<Entry<Integer, String>> itr1 = set.iterator();
60
61  while(itr1.hasNext()) {
62      Entry<Integer, String> entry = itr1.next();
63  }
64
65  }
66
67  }
68

```

## 8.6 Print the key-value pairs

```

11  System.out.println(map);
12
13  String name = map.get(121);
14  System.out.println("name is: "+name);
15
16  if(map.containsKey(567)) {
17      System.out.println("567 is in the map");
18  }
19
20  if(map.containsValue("Dave")) {
21      System.out.println("Dave is in the map");
22  }
23
24  System.out.println("Size of map is: "+map.size());
25  map.remove(567);
26  System.out.println("Size of map after remove is: "+map.size());
27
28  Set<Integer> keys = map.keySet();
29  System.out.println("Keys in map are: "+keys);
30
31  Iterator<Integer> itr = keys.iterator();
32  while(itr.hasNext()) {
33      Integer key = itr.next();
34      String value = map.get(key);
35      System.out.println(key+" \t "+value);
36  }
37
38  Set<Entry<Integer, String>> set = map.entrySet();
39  Iterator<Entry<Integer, String>> itr1 = set.iterator();
40
41  while(itr1.hasNext()) {
42      Entry<Integer, String> entry = itr1.next();
43      System.out.println(entry.getKey()+" \t "+entry.getValue());
44  }
45
46  }
47
48  }

```

## 8.7 Print Iterate in Keys

```

11  System.out.println(map);
12
13  String name = map.get(121);
14  System.out.println("name is: "+name);
15
16  if(map.containsKey(567)) {
17      System.out.println("567 is in the map");
18  }
19
20  if(map.containsValue("Dave")) {
21      System.out.println("Dave is in the map");
22  }
23
24  System.out.println("Size of map is: "+map.size());
25  map.remove(567);
26  System.out.println("Size of map after remove is: "+map.size());
27
28  System.out.println();
29  System.out.println("Iterate in Keys");
30  Set<Integer> keys = map.keySet();
31  System.out.println("Keys in map are: "+keys);
32
33  Iterator<Integer> itr = keys.iterator();
34  while(itr.hasNext()) {
35      Integer key = itr.next();
36      String value = map.get(key);
37      System.out.println(key+" \t "+value);
38  }
39
40  Set<Entry<Integer, String>> set = map.entrySet();
41  Iterator<Entry<Integer, String>> itr1 = set.iterator();
42
43  while(itr1.hasNext()) {
44      Entry<Integer, String> entry = itr1.next();
45      System.out.println(entry.getKey()+" \t "+entry.getValue());
46  }
47
48  }

```

## 8.8 Print Iterate using entry set

```

31  System.out.println(map);
32
33  String name = map.get(121);
34  System.out.println("name is: "+name);
35
36  if(map.containsKey(567)) {
37      System.out.println("567 is in the map");
38  }
39
40  if(map.containsValue("Dave")) {
41      System.out.println("Dave is in the map");
42  }
43
44  System.out.println("Size of map is: "+map.size());
45  map.remove(567);
46  System.out.println("Size of map after remove is: "+map.size());
47
48  System.out.println();
49  System.out.println("Iterate in Keys");
50  Set<Integer> keys = map.keySet();
51  System.out.println("Keys in map are: "+keys);
52
53  Iterator<Integer> itr = keys.iterator();
54  while(itr.hasNext()) {
55      Integer key = itr.next();
56      String value = map.get(key);
57      System.out.println(key+" \t "+value);
58  }
59
60  System.out.println();
61  System.out.println("Iterate using entry set");
62
63  Set<Entry<Integer, String>> set = map.entrySet();
64  Iterator<Entry<Integer, String>> itr1 = set.iterator();
65
66  while(itr1.hasNext()) {
67      Entry<Integer, String> entry = itr1.next();
68  }

```

## 8.9 Run the code and observe the printed key-value pairs

```

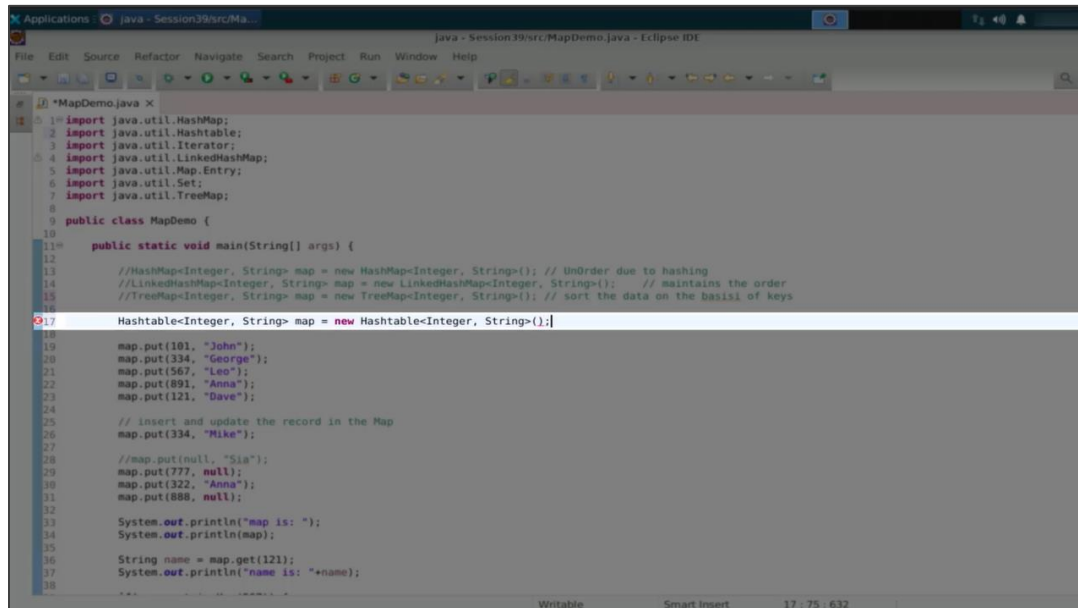
<terminated> MapDemo [Java Application] /usr/eclipse/plugins/org.eclipse.j
Keys in map are: [101, 121, 322, 334, 777, 888, 891]
101    John
121    Dave
322    Anna
334    Mike
777    null
888    null
891    Anna

Iterate using entry set
101    John
121    Dave
322    Anna
334    Mike
777    null
888    null
891    Anna

```

## Step 9: Work with a Hashtable

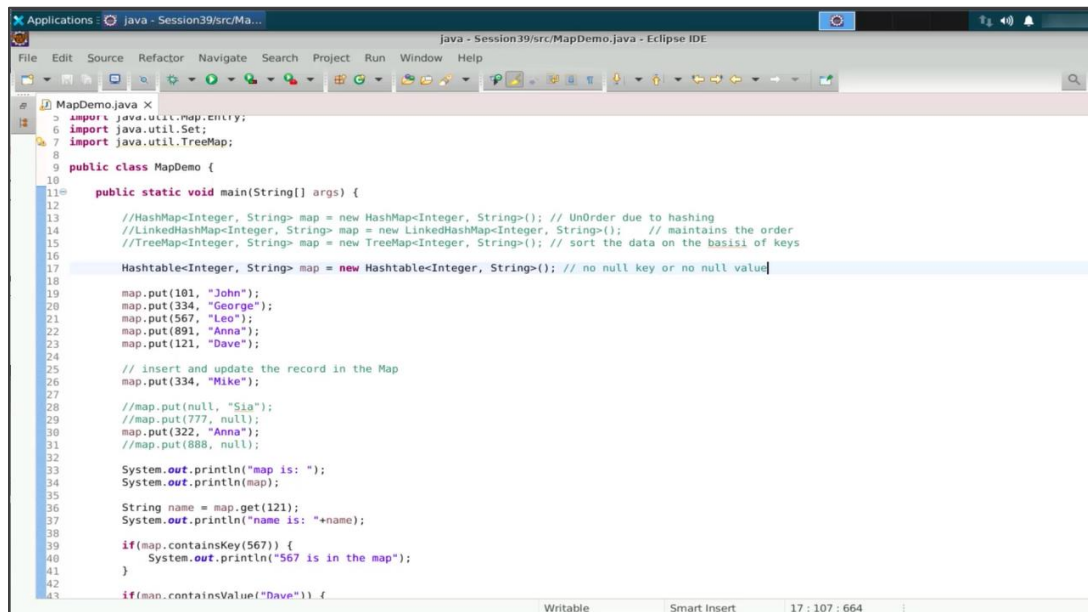
### 9.1 Create a Hashtable named `map`



```
1 import java.util.HashMap;
2 import java.util.Hashtable;
3 import java.util.Iterator;
4 import java.util.LinkedHashMap;
5 import java.util.Map.Entry;
6 import java.util.Set;
7 import java.util.TreeMap;
8
9 public class MapDemo {
10
11     public static void main(String[] args) {
12         //HashMap<Integer, String> map = new HashMap<Integer, String>(); // UnOrder due to hashing
13         //LinkedHashMap<Integer, String> map = new LinkedHashMap<Integer, String>(); // maintains the order
14         //TreeMap<Integer, String> map = new TreeMap<Integer, String>(); // sort the data on the basis of keys
15
16         Hashtable<Integer, String> map = new Hashtable<Integer, String>();
17
18         map.put(101, "John");
19         map.put(334, "George");
20         map.put(567, "Leo");
21         map.put(891, "Anna");
22         map.put(121, "Dave");
23
24         // insert and update the record in the Map
25         map.put(334, "Mike");
26
27         //map.put(null, "Sia");
28         map.put(777, null);
29         map.put(322, "Anna");
30         map.put(888, null);
31
32         System.out.println("map is: ");
33         System.out.println(map);
34
35         String name = map.get(121);
36         System.out.println("name is: "+name);
37
38     }
39 }
```

Note: Hashtable does not allow null keys or values.

## 9.2 Add a comment indicating the restriction on null keys or values



```
1  MapDemo.java X
2  java - Session39/src/MapDemo.java - Eclipse IDE
3  File Edit Source Refactor Navigate Search Project Run Window Help
4
5  1 import java.util.Map.Entry;
6  2 import java.util.Set;
7  3 import java.util.TreeMap;
8
9  4 public class MapDemo {
10
11  5     public static void main(String[] args) {
12
13  6         //HashMap<Integer, String> map = new HashMap<Integer, String>(); // UnOrder due to hashing
14  7         //LinkedHashMap<Integer, String> map = new LinkedHashMap<Integer, String>(); // maintains the order
15  8         //TreeMap<Integer, String> map = new TreeMap<Integer, String>(); // sort the data on the basis of keys
16
17  9         Hashtable<Integer, String> map = new Hashtable<Integer, String>(); // no null key or no null value
18
19 10         map.put(101, "John");
20 11         map.put(334, "George");
21 12         map.put(567, "Leo");
22 13         map.put(891, "Anna");
23 14         map.put(121, "Dave");
24
25 15         // insert and update the record in the Map
26 16         map.put(334, "Mike");
27
28 17         //map.put(null, "Sia");
29 18         //map.put(777, null);
30 19         map.put(322, "Anna");
31 20         //map.put(888, null);
32
33 21         System.out.println("map is: ");
34 22         System.out.println(map);
35
36 23         String name = map.get(121);
37 24         System.out.println("name is: "+name);
38
39 25         if(map.containsKey(567)) {
40 26             System.out.println("567 is in the map");
41 27         }
42
43 28         if(map.containsValue("Dave")) {
```

By following these steps, you have successfully implemented and demonstrated various functionalities of HashMap, LinkedHashMap, and Hashtable in Java.