# 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

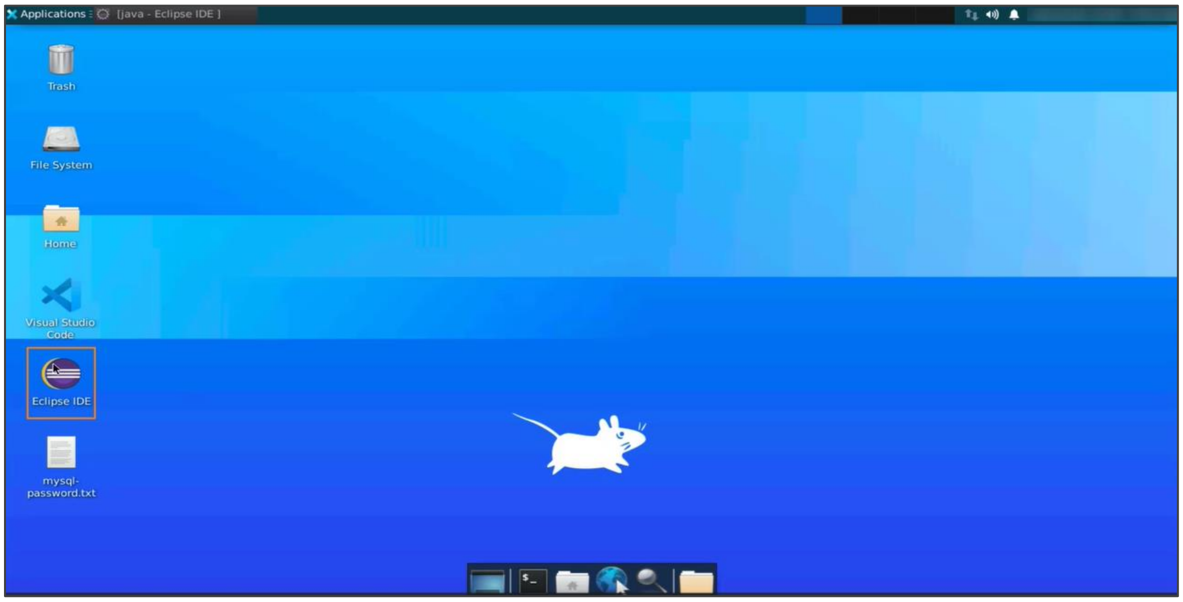
**Prerequisites:** None

Steps to be followed:

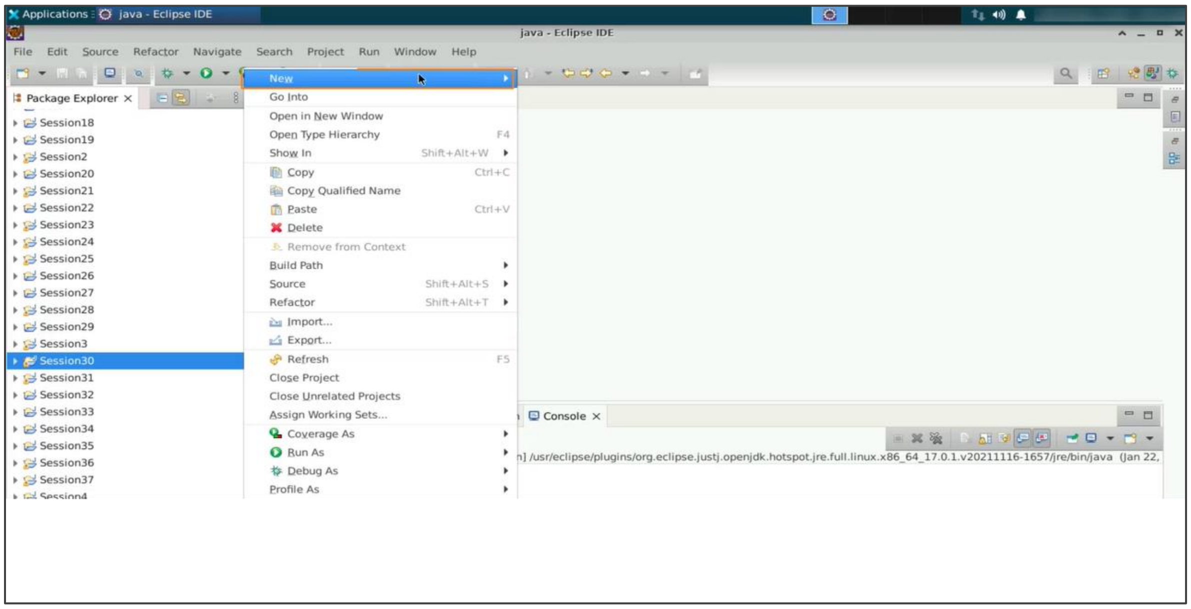
1. Create a new project
2. Use a HashMap
3. Execute the code with example data
4. Sort the data based on keys
5. Iterate the data structure with example data
6. Use the remove() method and execute the code
7. Implement iteration to obtain all the keys from the map
8. Execute the entrySet() method and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

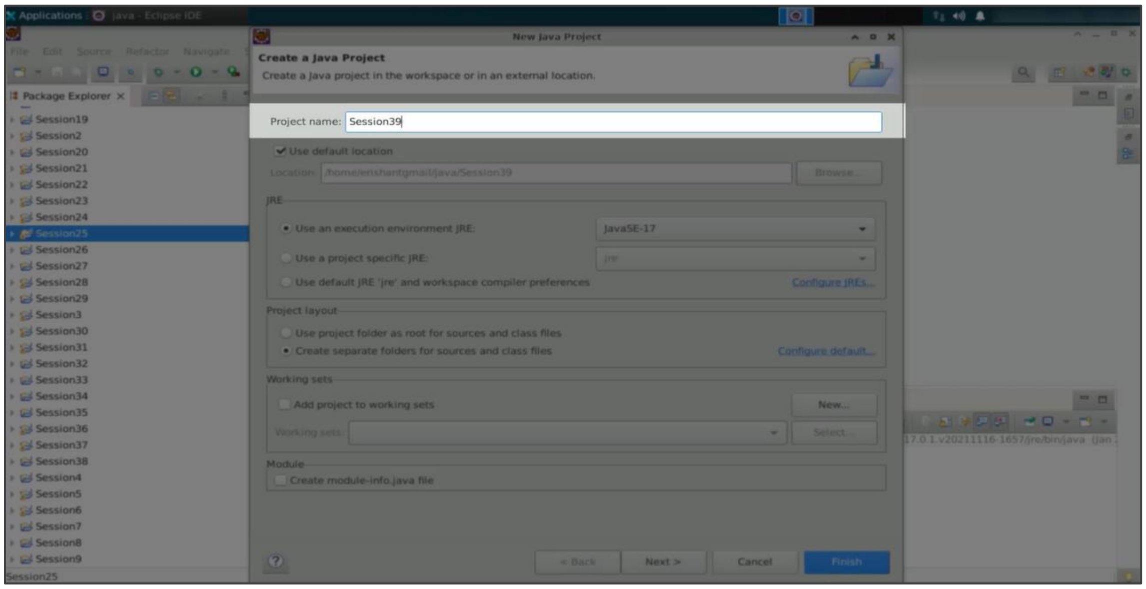


1.2 Create a new Java project



```
] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_17.0.1.v20211116-1657/jre/bin/java (Jan 22,
```

1.3 Name the new project **Session39**



New Java Project

Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: Session39

[✓] Use default location

Location: /home/ershantgmail/java/Session39

JRE

(●) Use an execution environment JRE: JavaSE-17

( ) Use a project specific JRE: jre

( ) Use default JRE 'jre' and workspace compiler preferences

Project layout

( ) Use project folder as root for sources and class files

(●) Create separate folders for sources and class files

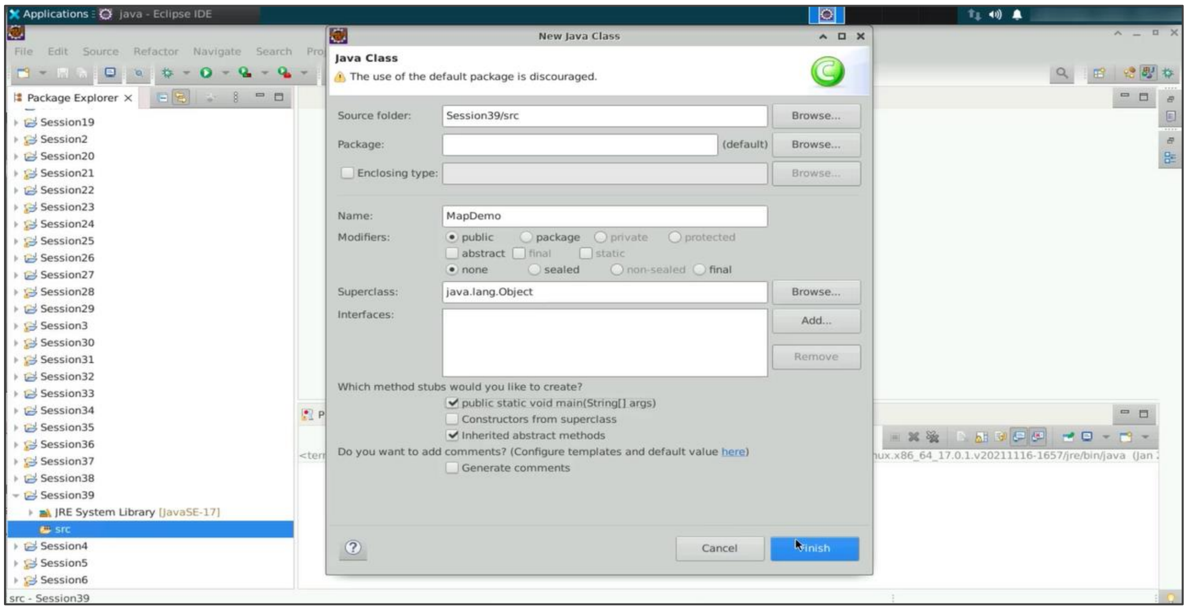
Working sets

[ ] Add project to working sets

Module

[ ] Create module-info.java file

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



New Java Class

Java Class

The use of the default package is discouraged.

Source folder: Session39/src

Package: (default)

[ ] Enclosing type:

Name: MapDemo

Modifiers: (●) public ( ) package ( ) private ( ) protected

[ ] abstract [ ] final [ ] static

(●) none ( ) sealed ( ) non-sealed ( ) final

Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to create?

[✓] public static void main(String[] args)

[ ] Constructors from superclass

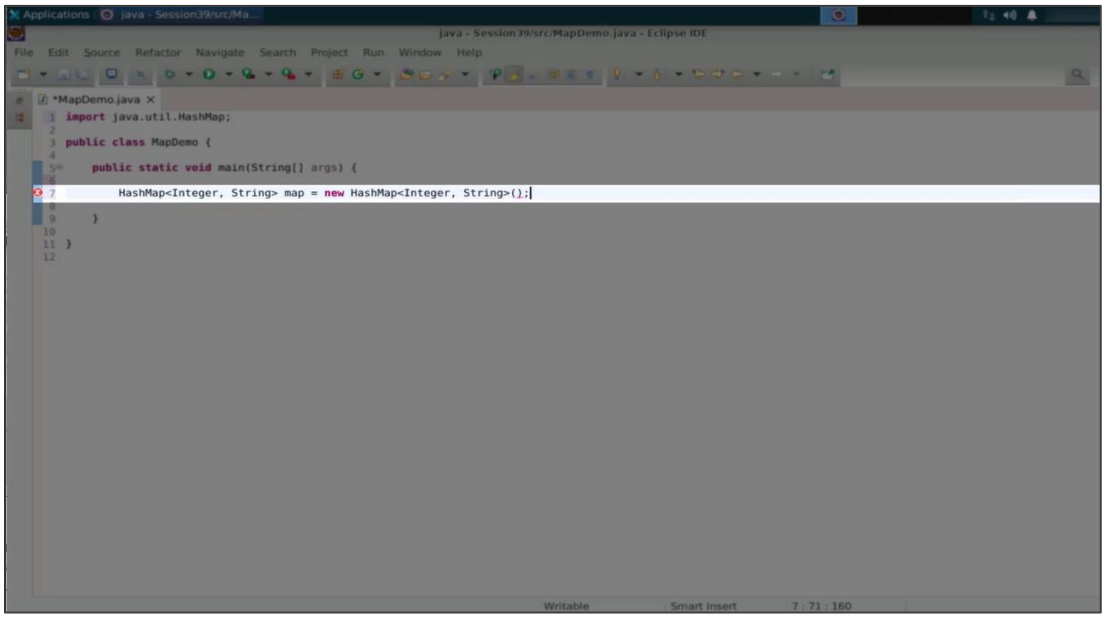
[✓] Inherited abstract methods

Do you want to add comments? (Configure templates and default value here)

[ ] Generate comments

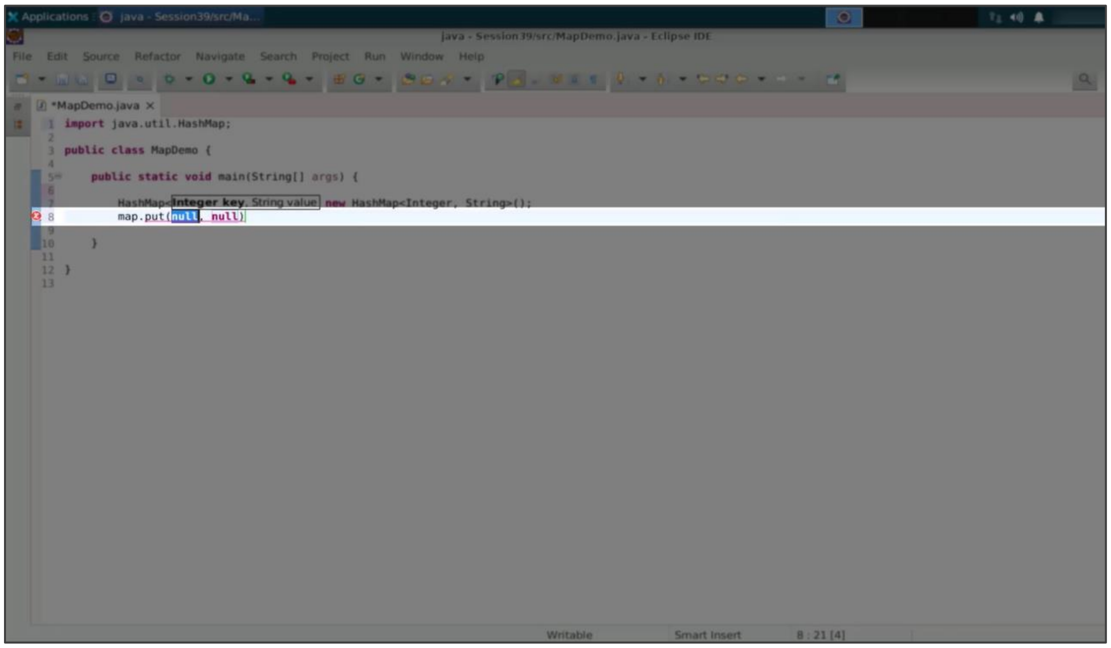
## Step 2: Use a HashMap

2.1 Create a new HashMap called **map**


Applications
java - Session39/src/Ma...
java - Session39/src/MapDemo.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
*MapDemo.java
Writable
Smart Insert
7 : 71 : 160

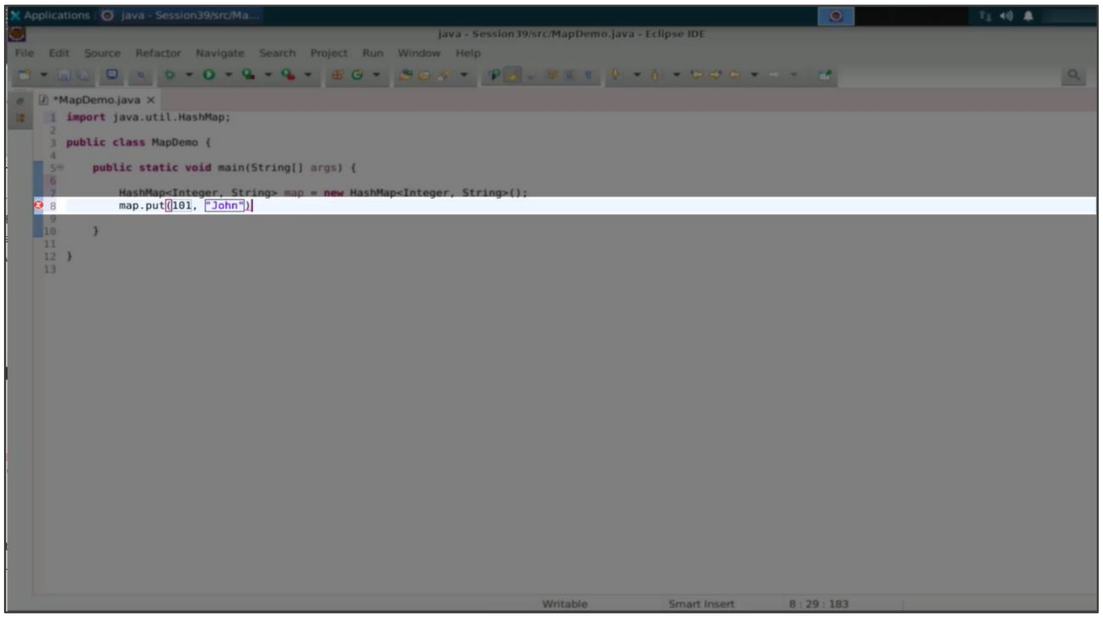
```java
import java.util.HashMap;

public class MapDemo {

    public static void main(String[] args) {

        HashMap<Integer, String> map = new HashMap<Integer, String>();

    }

}
```

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


Applications
java - Session39/src/Ma...
java - Session39/src/MapDemo.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
*MapDemo.java
Integer key, String value
Writable
Smart Insert
8 : 21 [4]

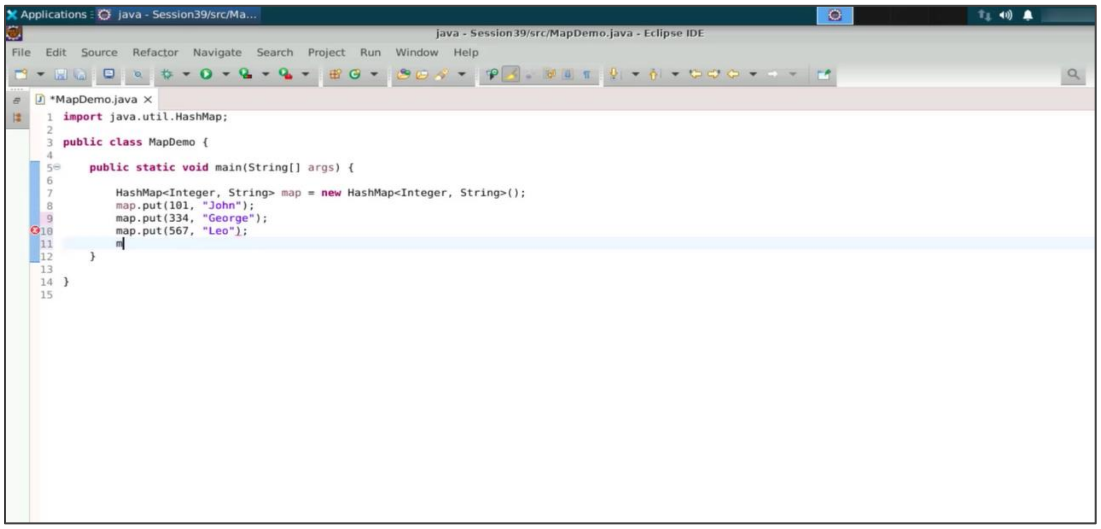
```java
import java.util.HashMap;

public class MapDemo {

    public static void main(String[] args) {

        HashMap<Integer, String> map = new HashMap<Integer, String>();
        map.put(null, null)

    }

}
```

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



```
import java.util.HashMap;

public class MapDemo {

    public static void main(String[] args) {

        HashMap<Integer, String> map = new HashMap<Integer, String>();
        map.put(101, "John")

    }

}
```

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

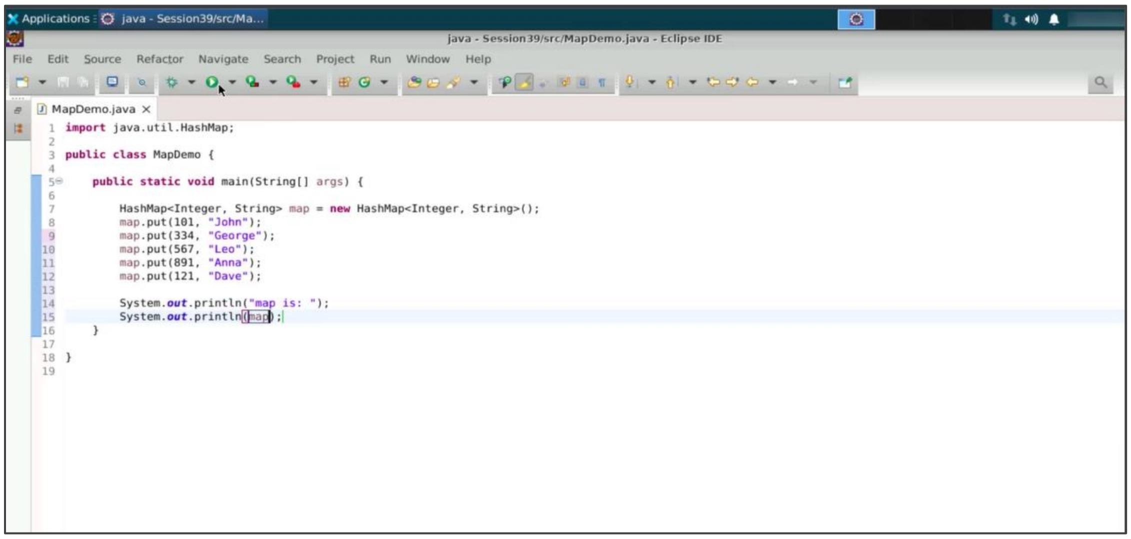


```
import java.util.HashMap;

public 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");
        m
    }

}
```

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.

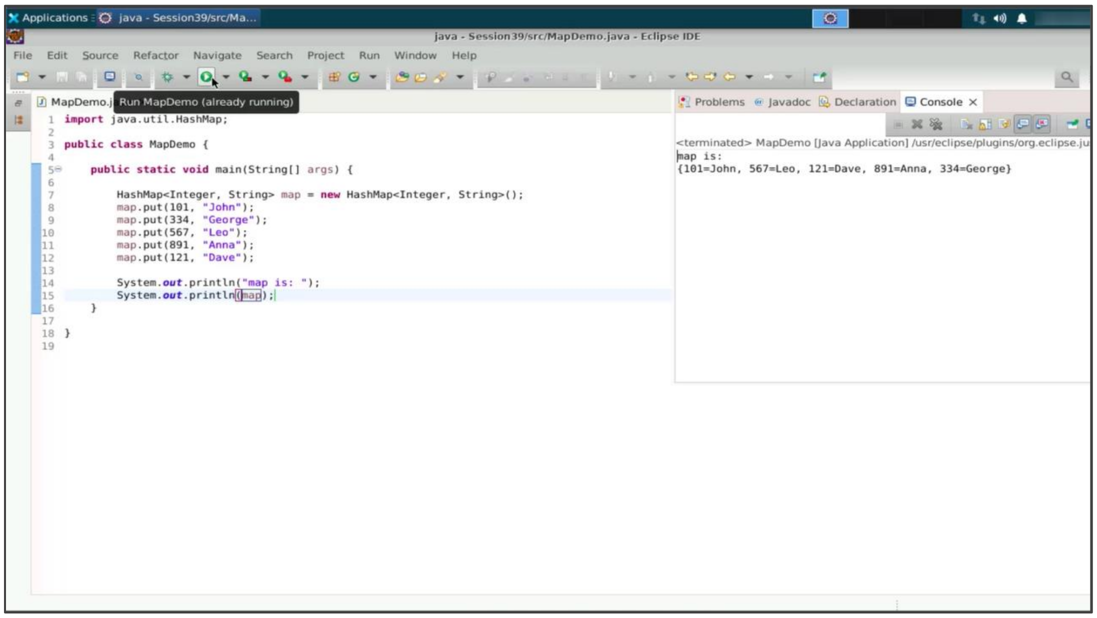
## Step 3: Execute the code with example data

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


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

```java
import java.util.HashMap;

public 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");

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

}
```

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

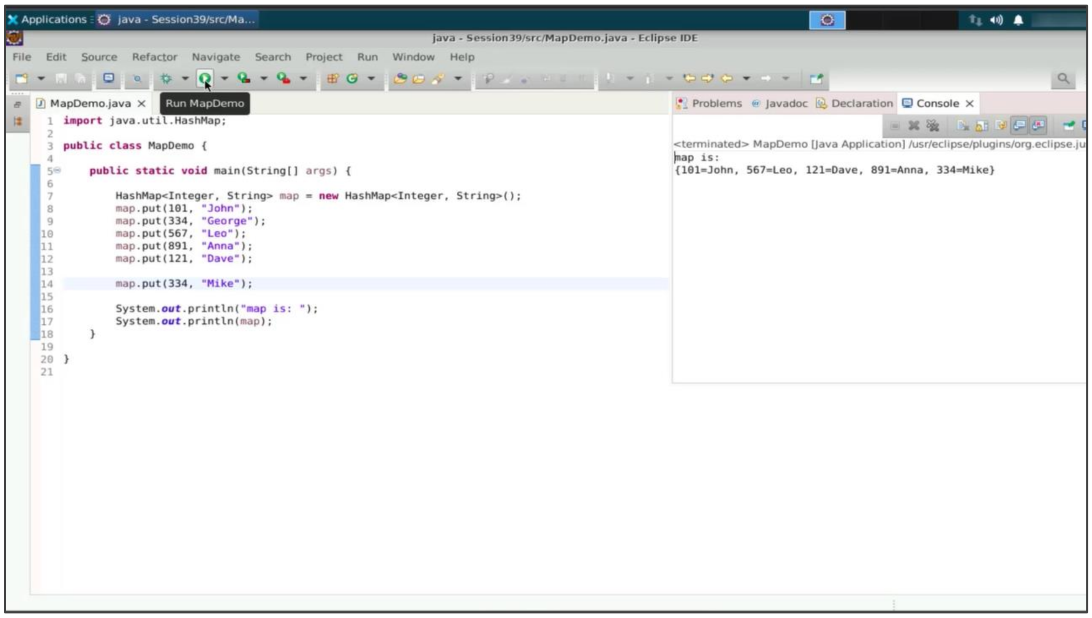

Applications
java - Session39/src/Ma...
java - Session39/src/MapDemo.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
MapDemo.j
Run MapDemo (already running)
Problems
Javadoc
Declaration
Console

```java
import java.util.HashMap;

public 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");

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

}
```

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

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

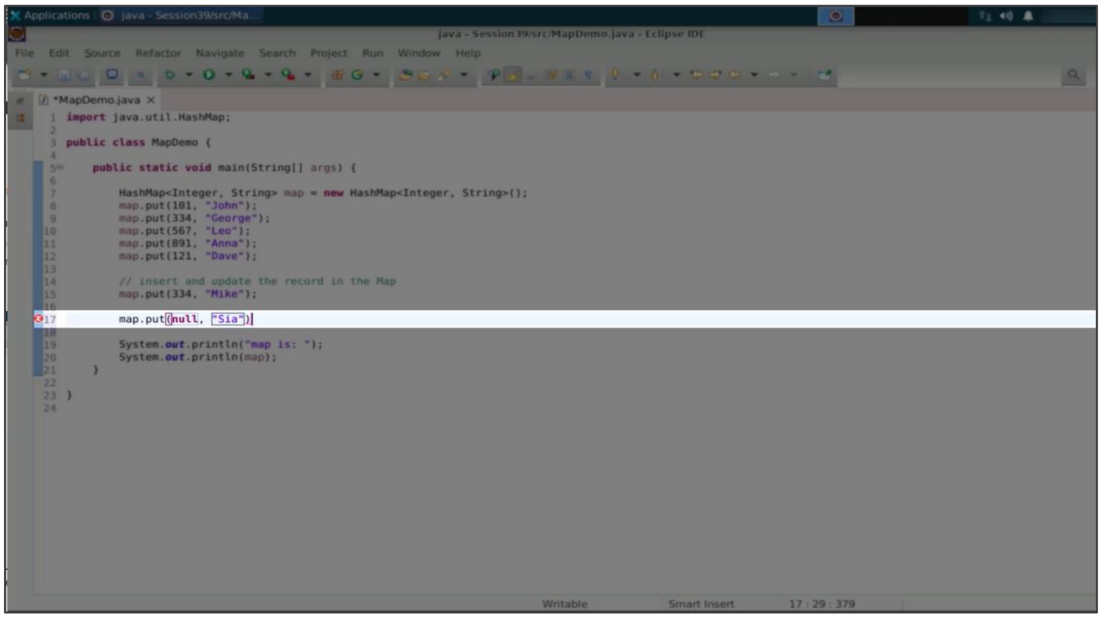
3.3 You can add new data using the **map.put()** method. If you try to add a duplicate key, it will update the corresponding value.



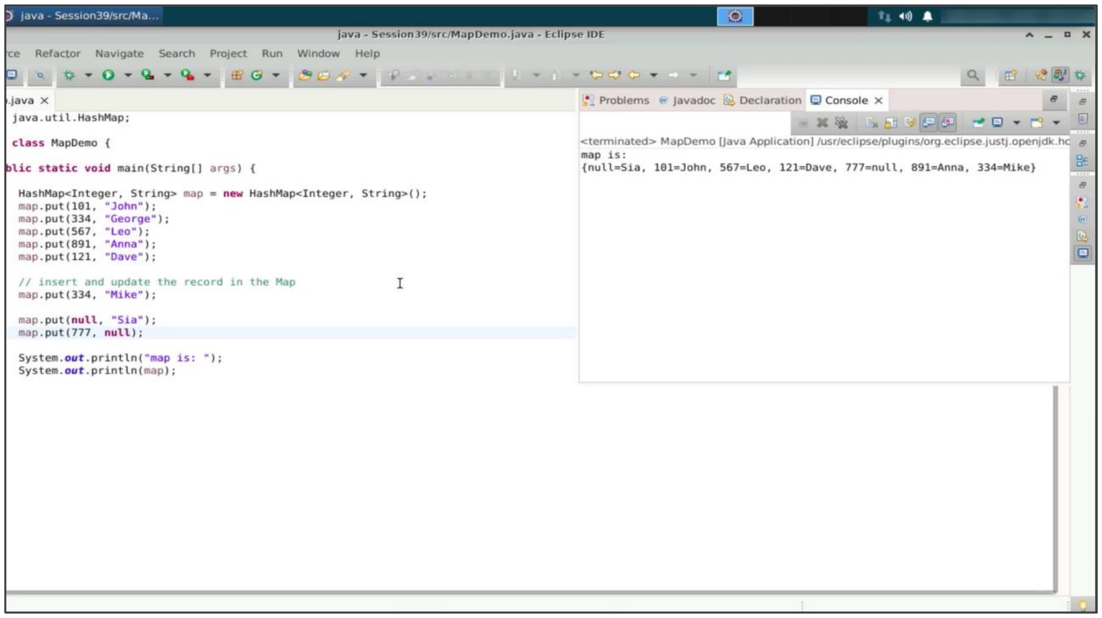
```
import java.util.HashMap;

public 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");

        map.put(334, "Mike");

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

}
```

```
<terminated> MapDemo [Java Application] /usr/eclipse/plugins/org.eclipse.ju
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



```
import java.util.HashMap;

public 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");

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

}
```

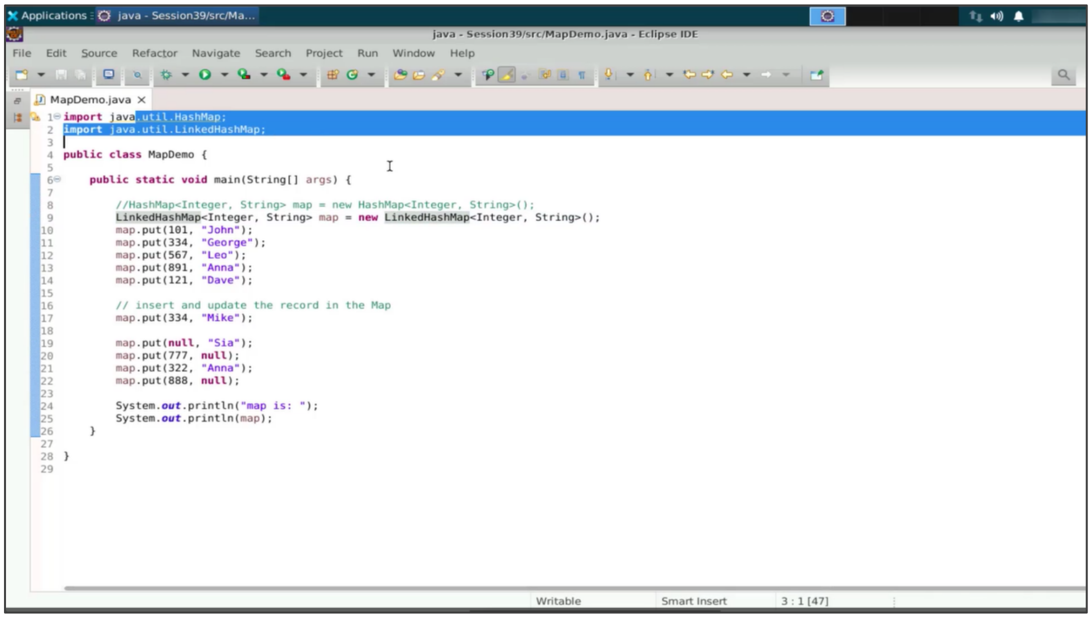


```
java.util.HashMap;

class MapDemo {

blic 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.justj.openjdk.hc
map is:
{null=Sia, 101=John, 567=Leo, 121=Dave, 777=null, 891=Anna, 334=Mike}
```

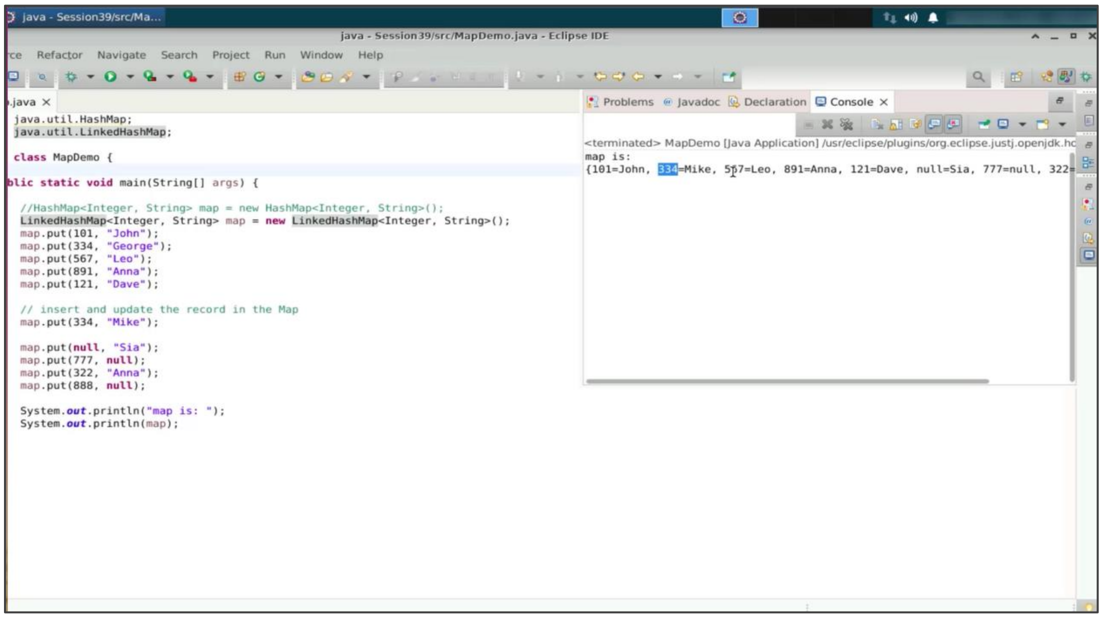
Duplicate values and multiple null values are allowed, but duplicate keys are not.

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



```java
import java.util.HashMap;
import java.util.LinkedHashMap;

public 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);
    }

}
```

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


java - Session39/src/Ma...
java - Session39/src/MapDemo.java - Eclipse IDE
rce Refactor Navigate Search Project Run Window Help
.java
Problems Javadoc Declaration Console

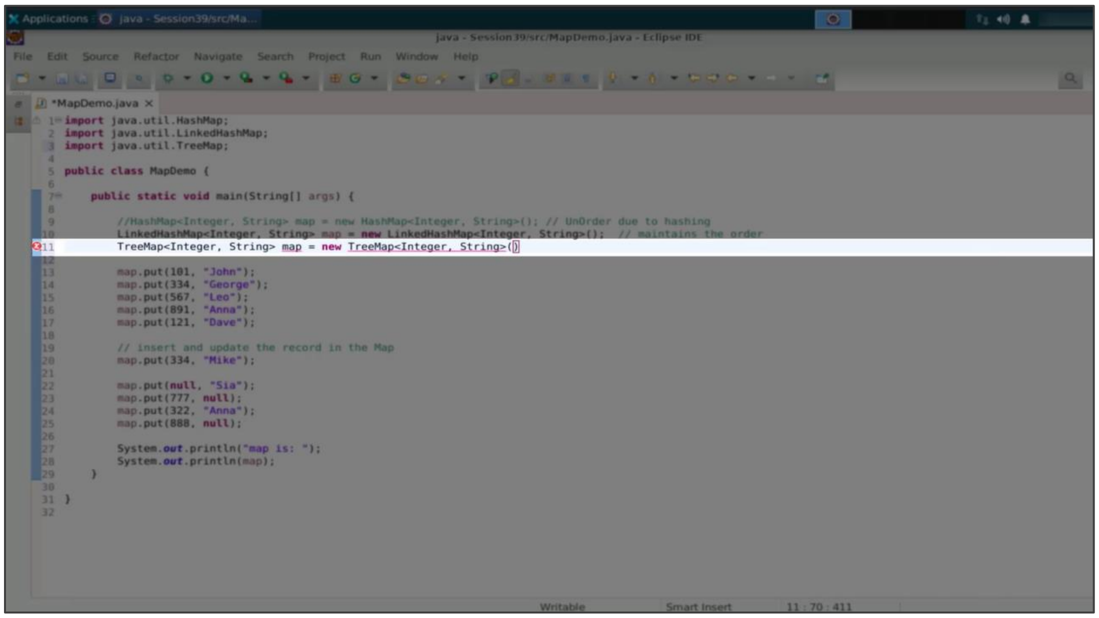
```
 java.util.HashMap;
 java.util.LinkedHashMap;

 class MapDemo {

blic 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);
```

```
<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=
```

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.

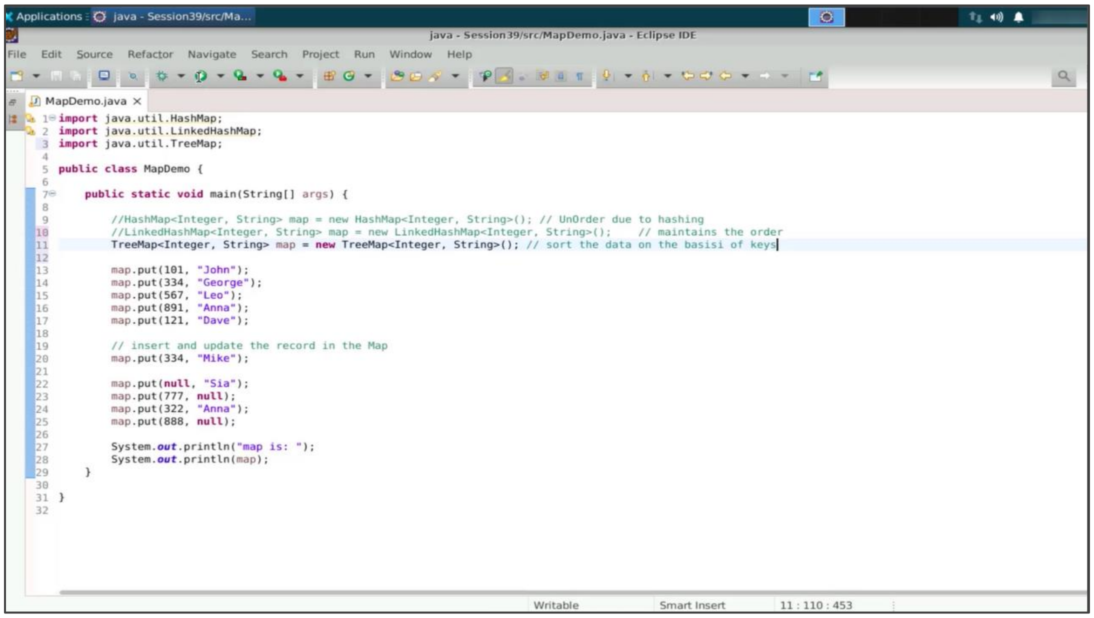
## Step 4: Sort the data based on keys

4.1 Create a new TreeMap



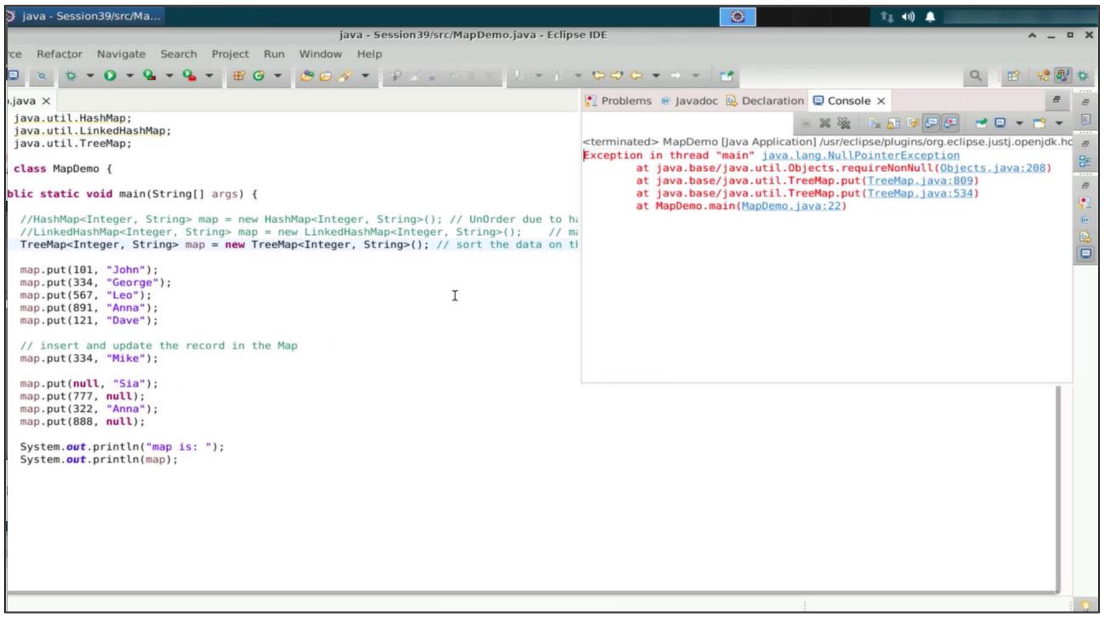
```java
import java.util.HashMap;
import java.util.LinkedHashMap;
import java.util.TreeMap;

public class MapDemo {

    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>()

        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);
    }

}
```

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



```java
import java.util.HashMap;
import java.util.LinkedHashMap;
import java.util.TreeMap;

public class MapDemo {

    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 basisi 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);
    }

}
```

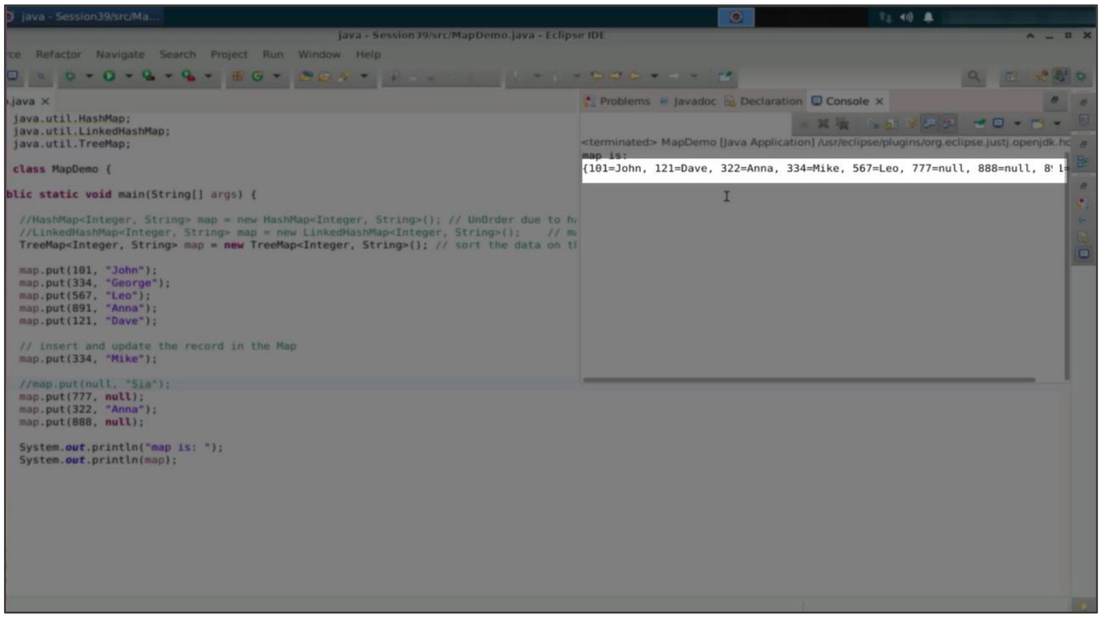
## 4.3 Resolve the null key issue



```
 java.util.HashMap;
 java.util.LinkedHashMap;
 java.util.TreeMap;

 class MapDemo {

blic static void main(String[] args) {

  //HashMap<Integer, String> map = new HashMap<Integer, String>(); // UnOrder due to ha
  //LinkedHashMap<Integer, String> map = new LinkedHashMap<Integer, String>();    // ma
  TreeMap<Integer, String> map = new TreeMap<Integer, String>(); // sort the data on tl

  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);
```

```
<terminated> MapDemo [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hc
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

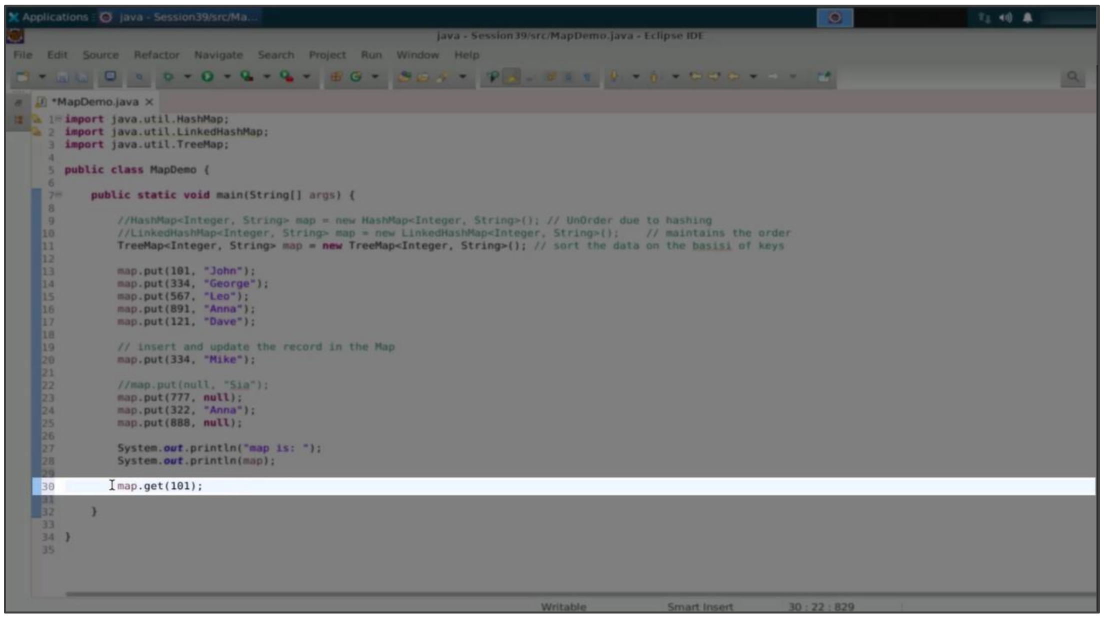


```
 java.util.HashMap;
 java.util.LinkedHashMap;
 java.util.TreeMap;

 class MapDemo {

blic static void main(String[] args) {

  //HashMap<Integer, String> map = new HashMap<Integer, String>(); // UnOrder due to ha
  //LinkedHashMap<Integer, String> map = new LinkedHashMap<Integer, String>();    // ma
  TreeMap<Integer, String> map = new TreeMap<Integer, String>(); // sort the data on tl

  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);
```

```
<terminated> MapDemo [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hc
map is:
{101=John, 121=Dave, 322=Anna, 334=Mike, 567=Leo, 777=null, 888=null, 8' l=
```

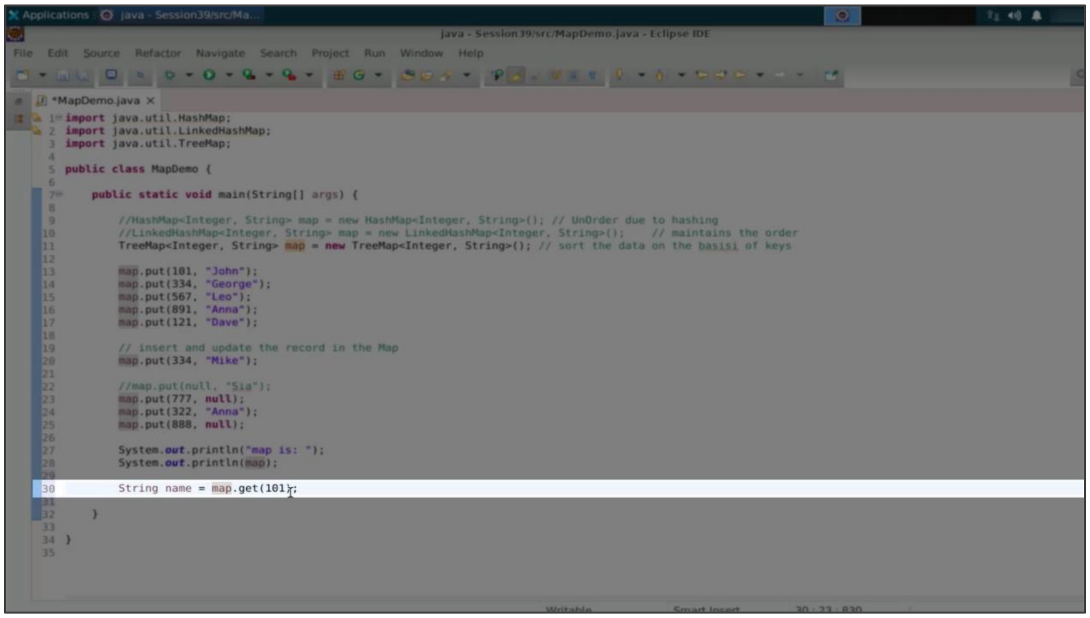
## Step 5: Iterate the data structure with example data

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



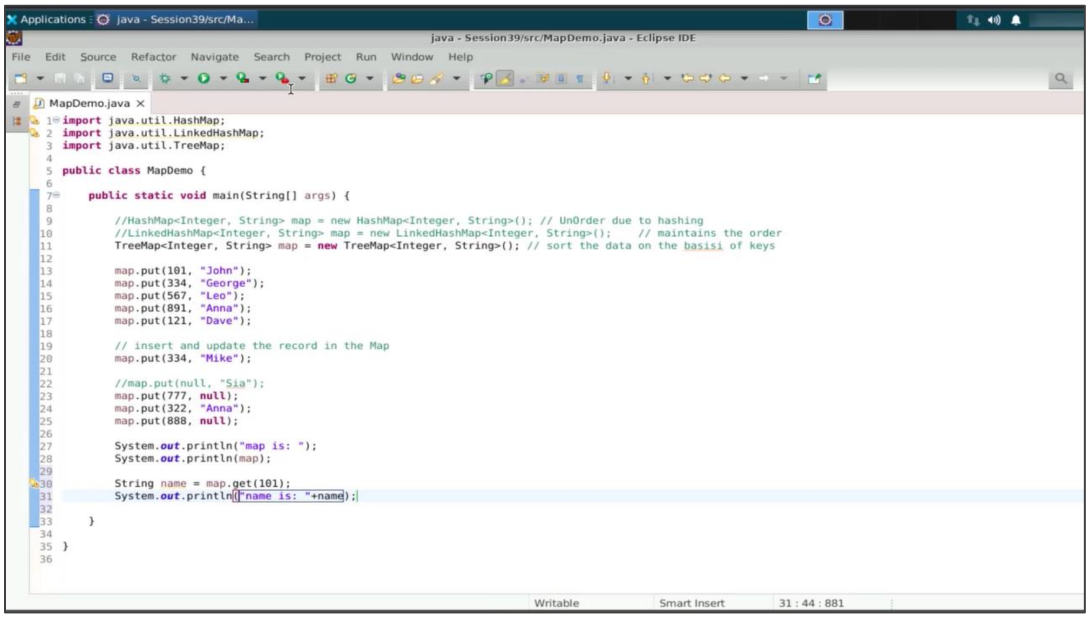
```java
import java.util.HashMap;
import java.util.LinkedHashMap;
import java.util.TreeMap;

public class MapDemo {

    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 basisi 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);

        map.get(101);

    }

}
```

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



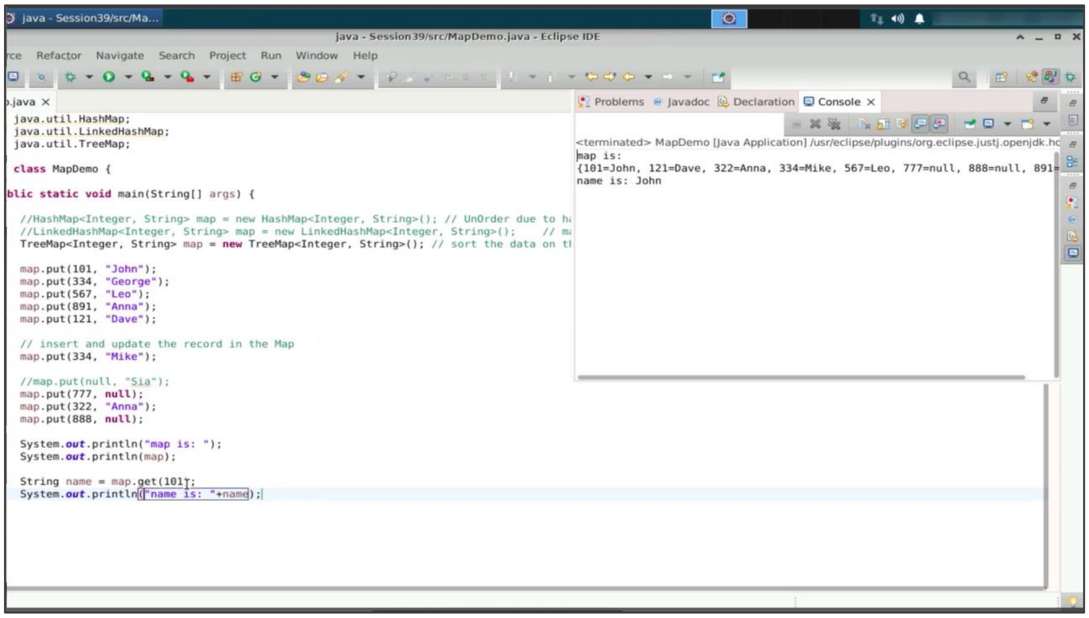
```java
import java.util.HashMap;
import java.util.LinkedHashMap;
import java.util.TreeMap;

public class MapDemo {

    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 basisi 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(101);

    }

}
```

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



```java
import java.util.HashMap;
import java.util.LinkedHashMap;
import java.util.TreeMap;

public class MapDemo {

    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 basisi 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(101);
        System.out.println("name is: "+name);

    }

}
```

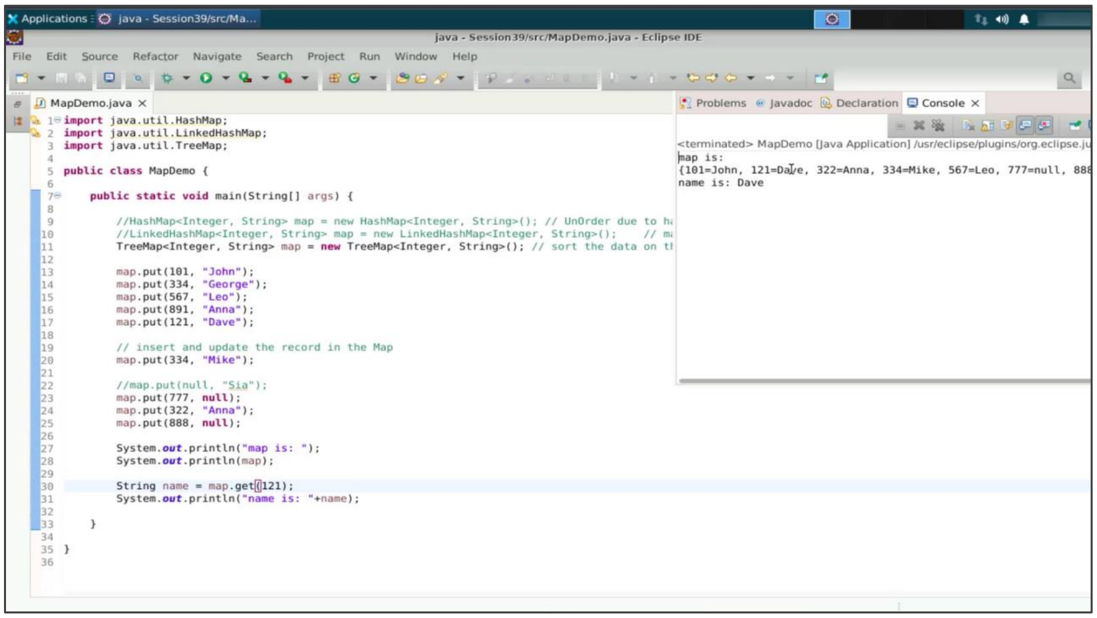
5.4 Run the code and observe the printed name



```java
java.util.HashMap;
java.util.LinkedHashMap;
java.util.TreeMap;

class MapDemo {

blic static void main(String[] args) {

  //HashMap<Integer, String> map = new HashMap<Integer, String>(); // UnOrder due to ha
  //LinkedHashMap<Integer, String> map = new LinkedHashMap<Integer, String>();    // ma
  TreeMap<Integer, String> map = new TreeMap<Integer, String>(); // sort the data on tl

  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(101);
  System.out.println("name is: "+name);
```

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

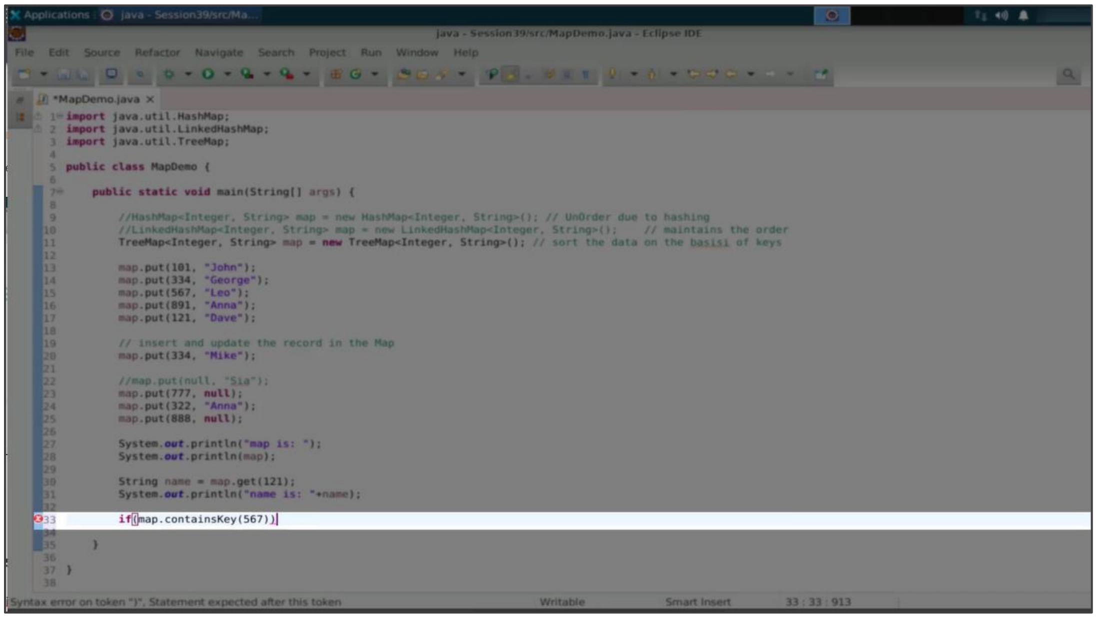
5.5 Repeat steps **9.1-9.4** with different keys to retrieve corresponding values



```
import java.util.HashMap;
import java.util.LinkedHashMap;
import java.util.TreeMap;

public class MapDemo {

    public static void main(String[] args) {

        //HashMap<Integer, String> map = new HashMap<Integer, String>(); // UnOrder due to ha
        //LinkedHashMap<Integer, String> map = new LinkedHashMap<Integer, String>();    // ma
        TreeMap<Integer, String> map = new TreeMap<Integer, String>(); // sort the data on th

        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);

    }
}
```

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

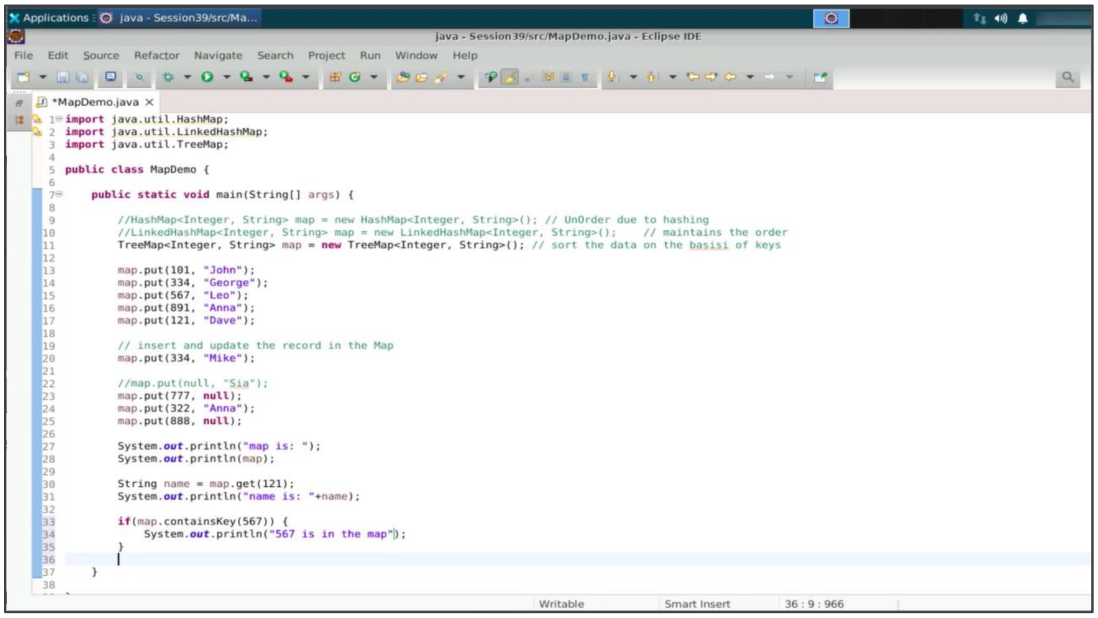
5.6 Use **map.containsKey(567)** to check if the key 567 exists



```
import java.util.HashMap;
import java.util.LinkedHashMap;
import java.util.TreeMap;

public class MapDemo {

    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 basisi 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))

    }
}
```

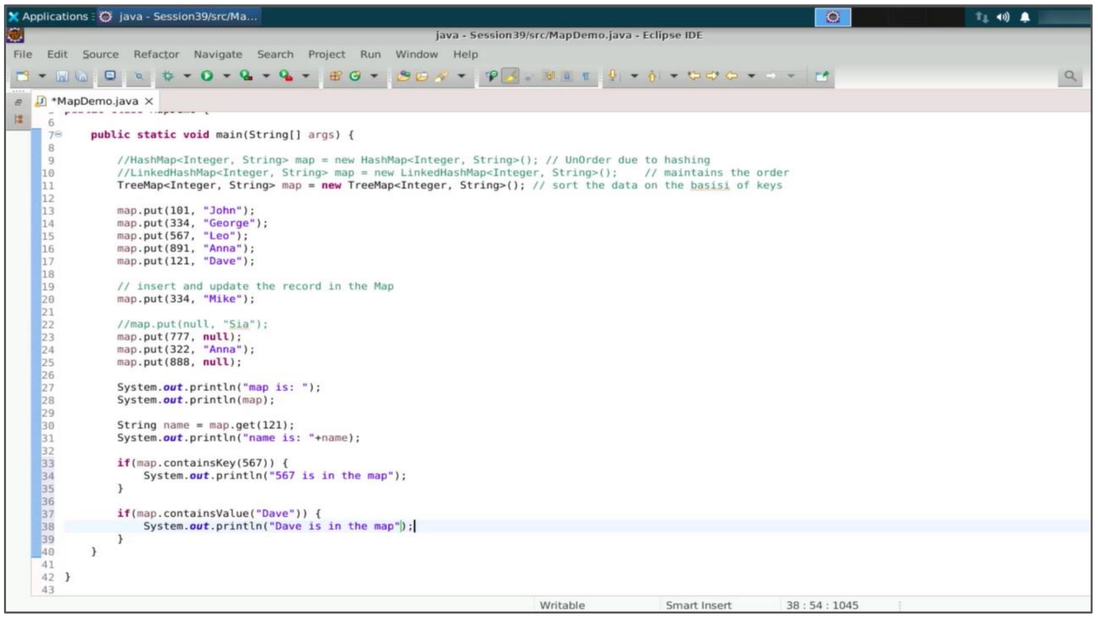
```
Syntax error on token ")", Statement expected after this token     Writable     Smart Insert     33 : 33 : 913
```

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



```java
import java.util.HashMap;
import java.util.LinkedHashMap;
import java.util.TreeMap;

public class MapDemo {

    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 basisi 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");
        }

    }
```

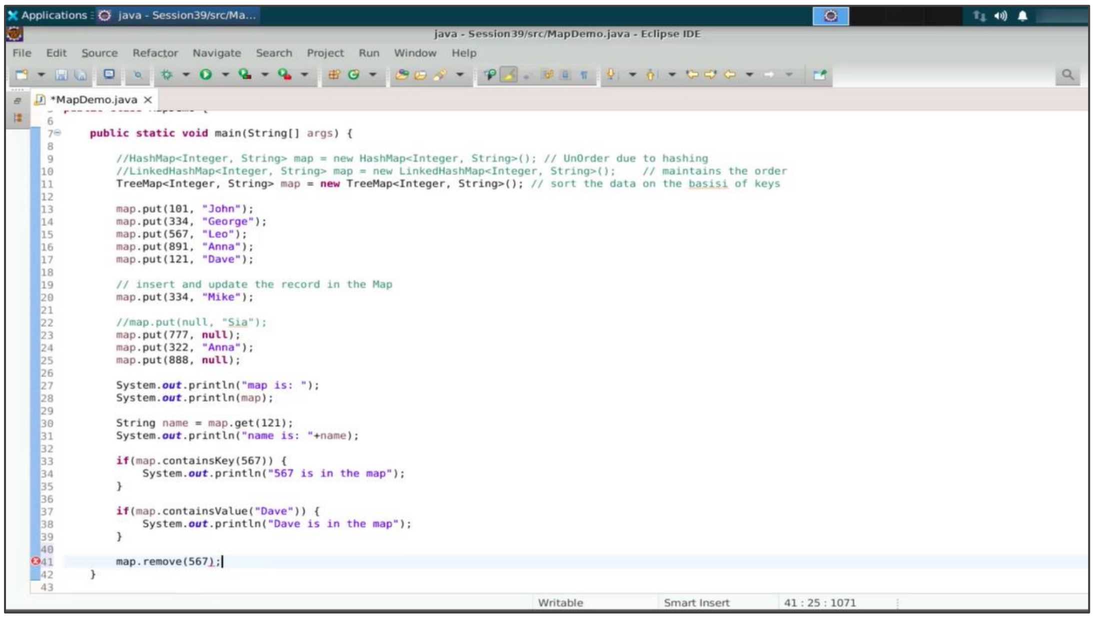
5.8 Use **map.containsValue("Dave")** to check if the value **Dave** exists



```java
    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 basisi 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");
        }
    }

}
```

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

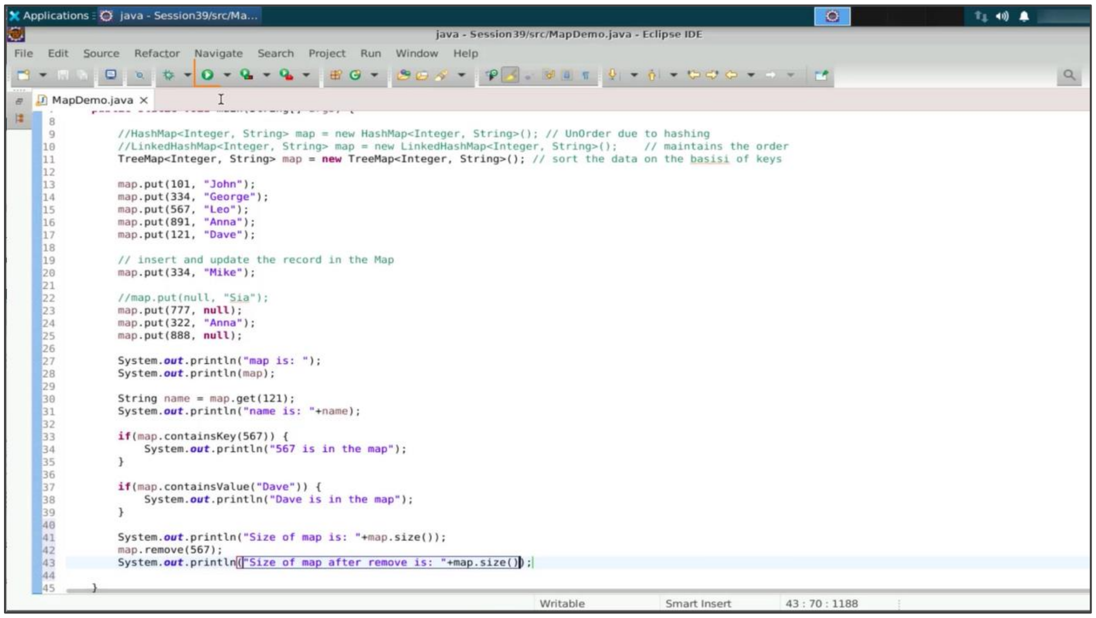
6.1 Use **map.remove(567)** to remove a key-value pair



Applications
java - Session39/src/Ma...
java - Session39/src/MapDemo.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
*MapDemo.java
Writable
Smart Insert
41 : 25 : 1071

```java
    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 basisi 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);
    }
```

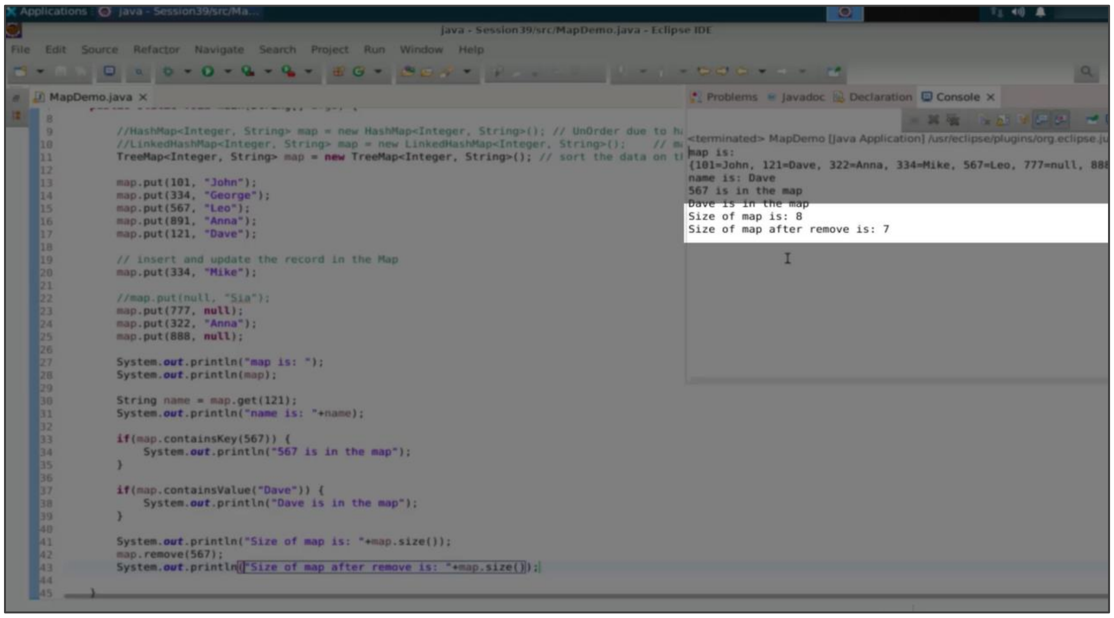
6.2 Print the size of the map before and after using **remove()**:



Applications
java - Session39/src/Ma...
java - Session39/src/MapDemo.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
MapDemo.java
Writable
Smart Insert
43 : 70 : 1188

```java
        //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 basisi 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

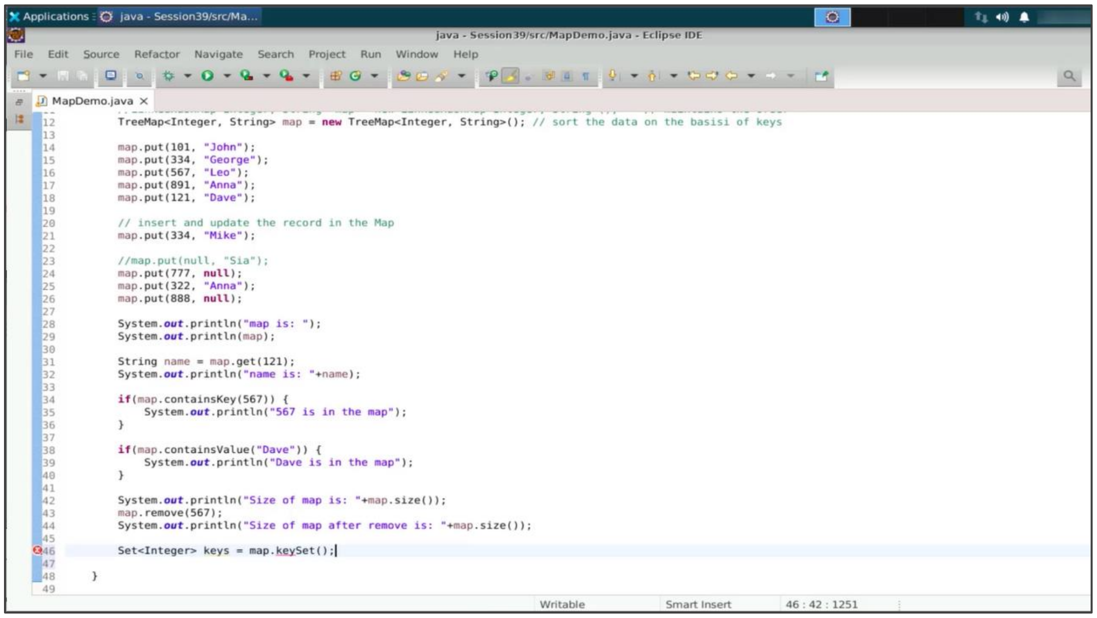


```java
        //HashMap<Integer, String> map = new HashMap<Integer, String>(); // UnOrder due to h
        //LinkedHashMap<Integer, String> map = new LinkedHashMap<Integer, String>();    // m
        TreeMap<Integer, String> map = new TreeMap<Integer, String>(); // sort the data on t

        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());

    }
```

```
<terminated> MapDemo [Java Application] /usr/eclipse/plugins/org.eclipse.ju
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
```

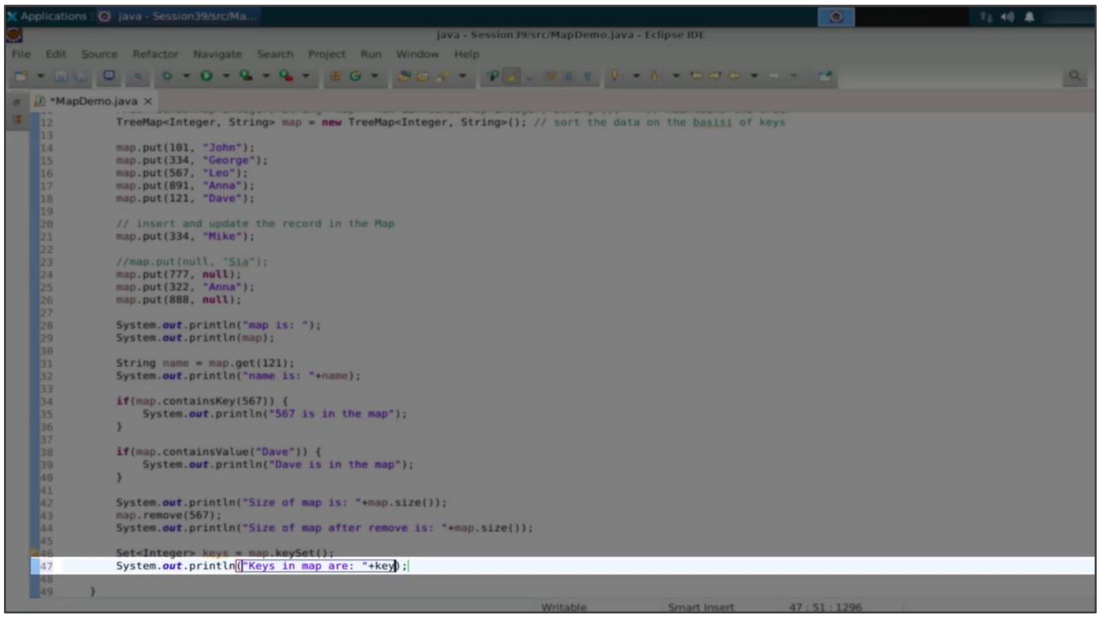
## Step 7: Implement iteration to obtain all the keys from the map

7.1 Obtain all the keys from the map using **map.keySet()** and a:ssign the keys to a set:
**Set<Integer> keys = map.keySet();**



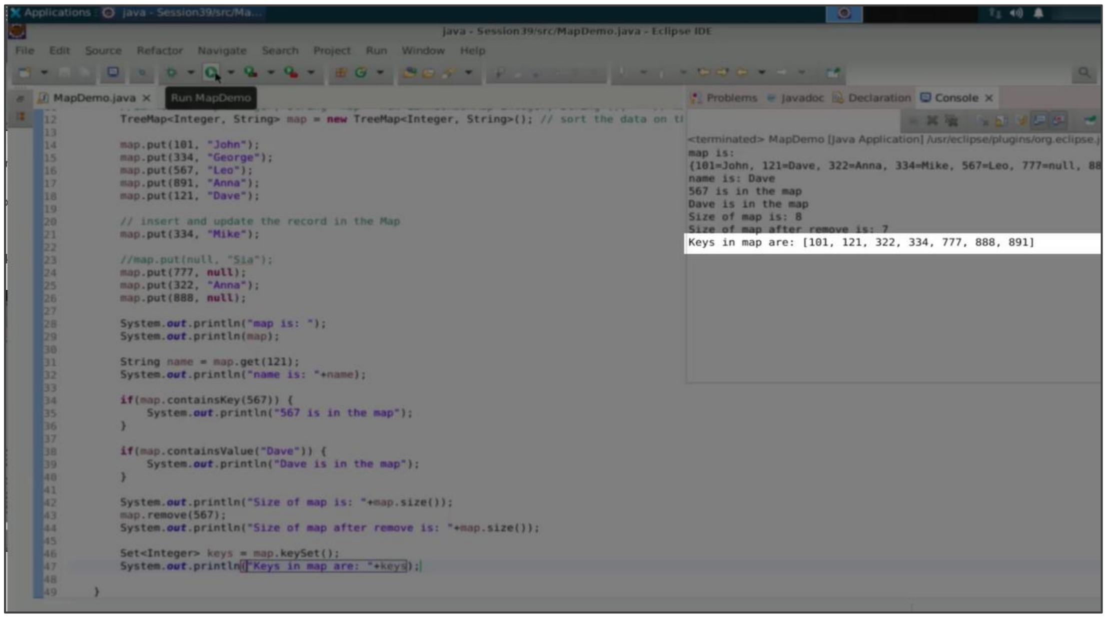
```java
        TreeMap<Integer, String> map = new TreeMap<Integer, String>(); // sort the data on the basisi of keys

        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());

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

    }
```

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



```java
        TreeMap<Integer, String> map = new TreeMap<Integer, String>(); // sort the data on the basisi 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());

        Set<Integer> keys = map.keySet();
        System.out.println("Keys in map are: "+keys);

    }
```

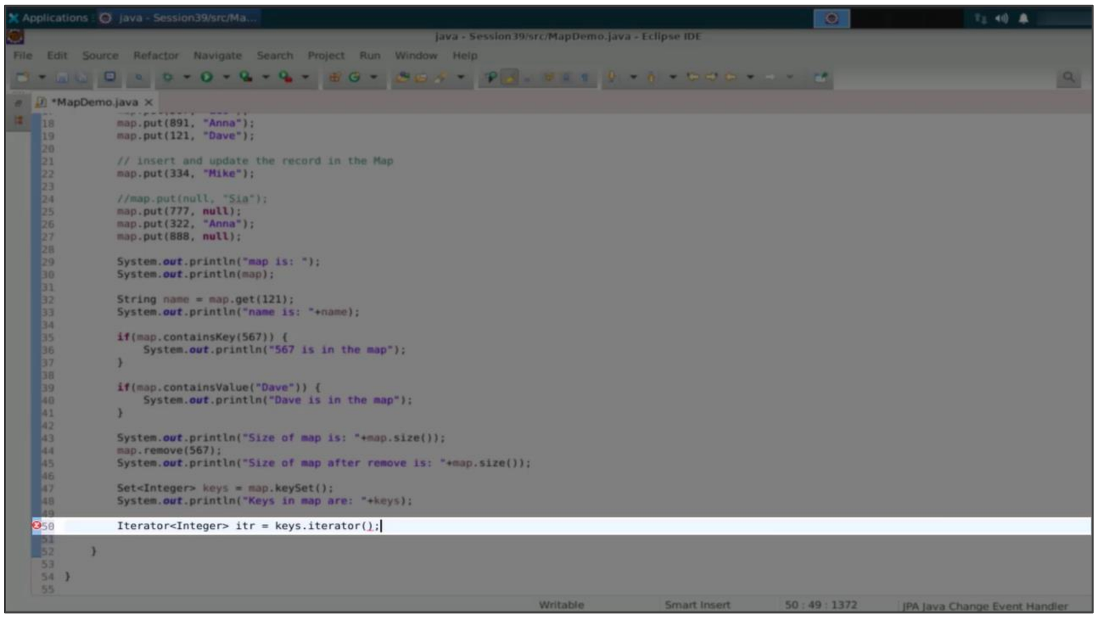
7.3 Run the code and observe the printed keys



```java
        TreeMap<Integer, String> map = new TreeMap<Integer, String>(); // sort the data on tl

        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());

        Set<Integer> keys = map.keySet();
        System.out.println("Keys in map are: "+keys);

    }
```

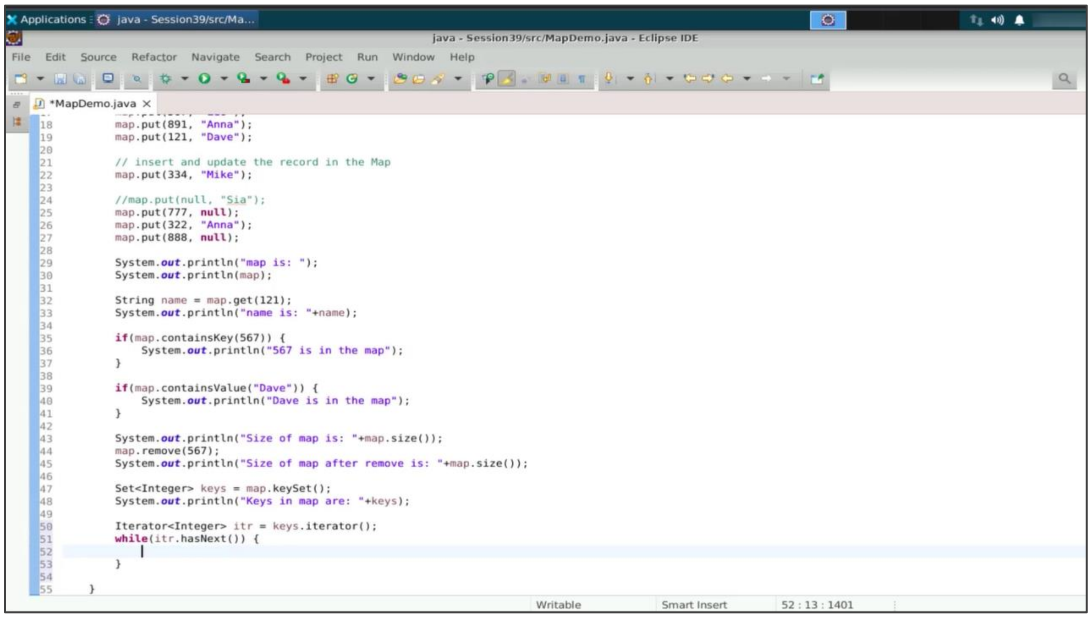
```
<terminated> MapDemo [Java Application] /usr/eclipse/plugins/org.eclipse.j
map is:
{101=John, 121=Dave, 322=Anna, 334=Mike, 567=Leo, 777=null, 88
name is: Dave
567 is in the map
Dave is in the map
Size of map is: 8
Size of map after remove is: 7
Keys in map are: [101, 121, 322, 334, 777, 888, 891]
```

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


Applications
java - Session39/src/Ma...
java - Session39/src/MapDemo.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
*MapDemo.java
Writable
Smart Insert
50 : 49 : 1372
JPA Java Change Event Handler

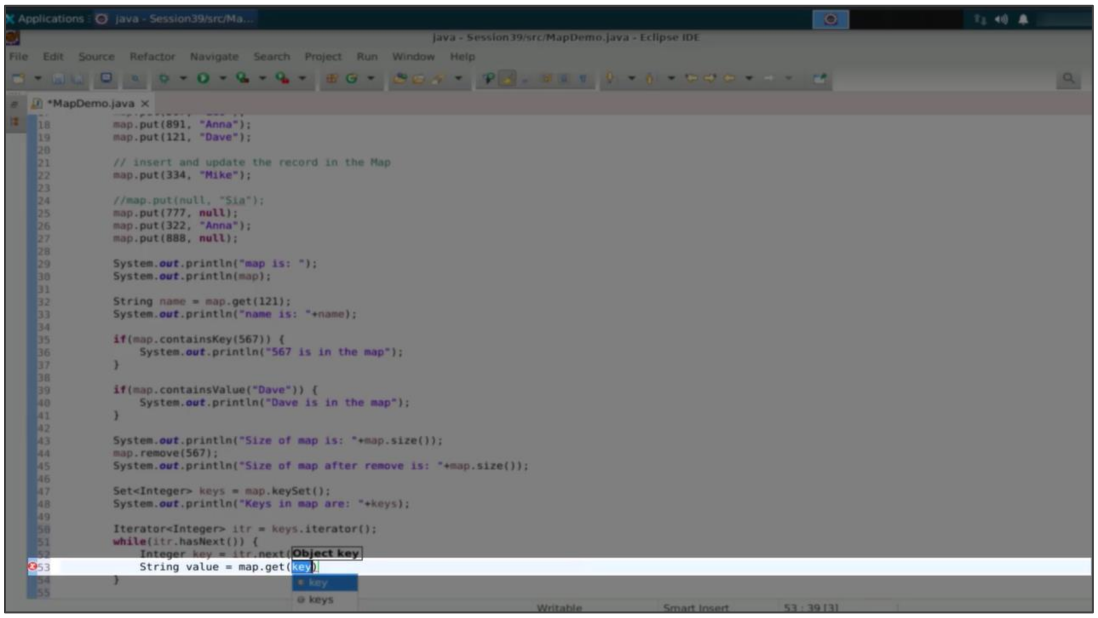
```
        map.put(891, "Anna");
        map.put(121, "Dave");

        // 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());

        Set<Integer> keys = map.keySet();
        System.out.println("Keys in map are: "+keys);

        Iterator<Integer> itr = keys.iterator();

    }

}
```

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


Applications
java - Session39/src/Ma...
java - Session39/src/MapDemo.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
*MapDemo.java
Writable
Smart Insert
52 : 13 : 1401

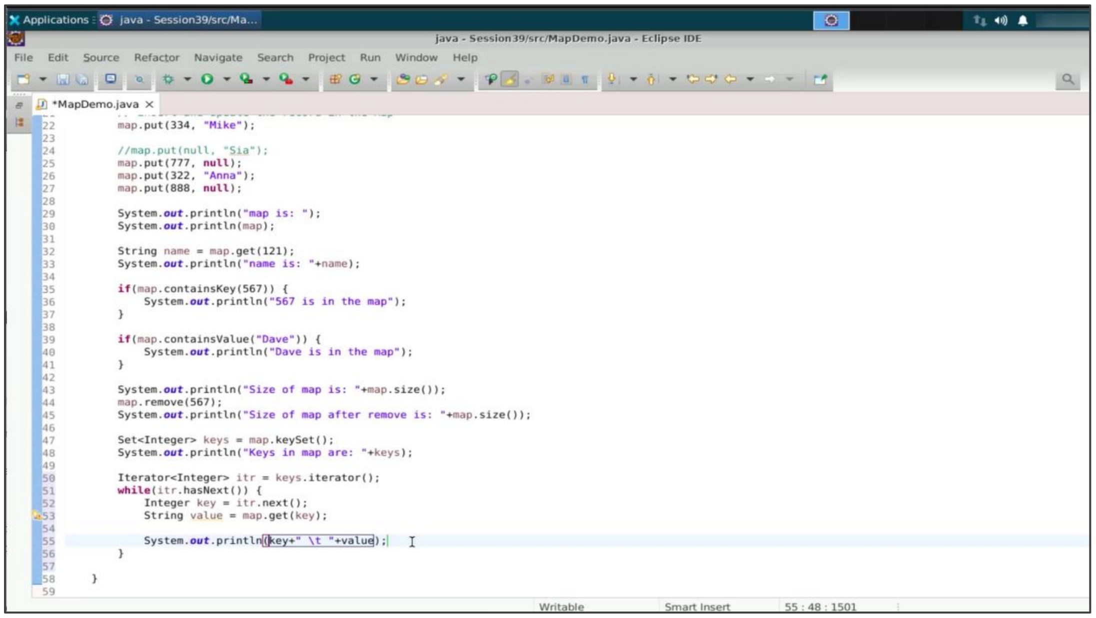
```
        map.put(891, "Anna");
        map.put(121, "Dave");

        // 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());

        Set<Integer> keys = map.keySet();
        System.out.println("Keys in map are: "+keys);

        Iterator<Integer> itr = keys.iterator();
        while(itr.hasNext()) {

        }


    }
```

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



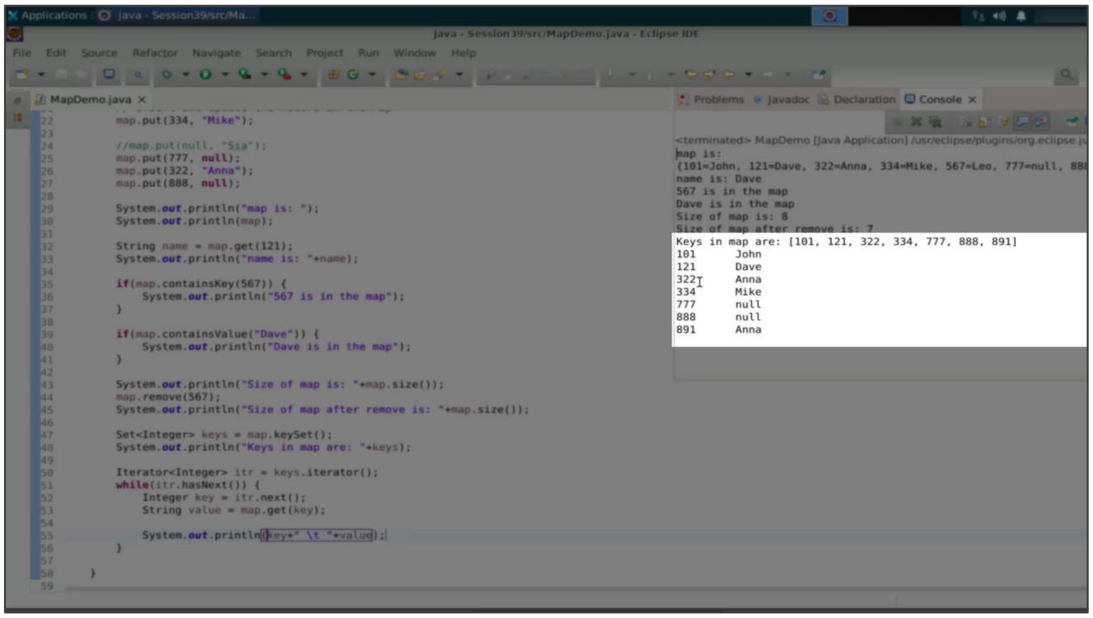
```
        map.put(891, "Anna");
        map.put(121, "Dave");

        // 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());

        Set<Integer> keys = map.keySet();
        System.out.println("Keys in map are: "+keys);

        Iterator<Integer> itr = keys.iterator();
        while(itr.hasNext()) {
            Integer key = itr.next(
            String value = map.get(key)
        }
```

7.7 Print the key-value pairs



```
        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());

        Set<Integer> keys = map.keySet();
        System.out.println("Keys in map are: "+keys);

        Iterator<Integer> itr = keys.iterator();
        while(itr.hasNext()) {
            Integer key = itr.next();
            String value = map.get(key);

            System.out.println(key+" \t "+value);
        }

    }
```

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

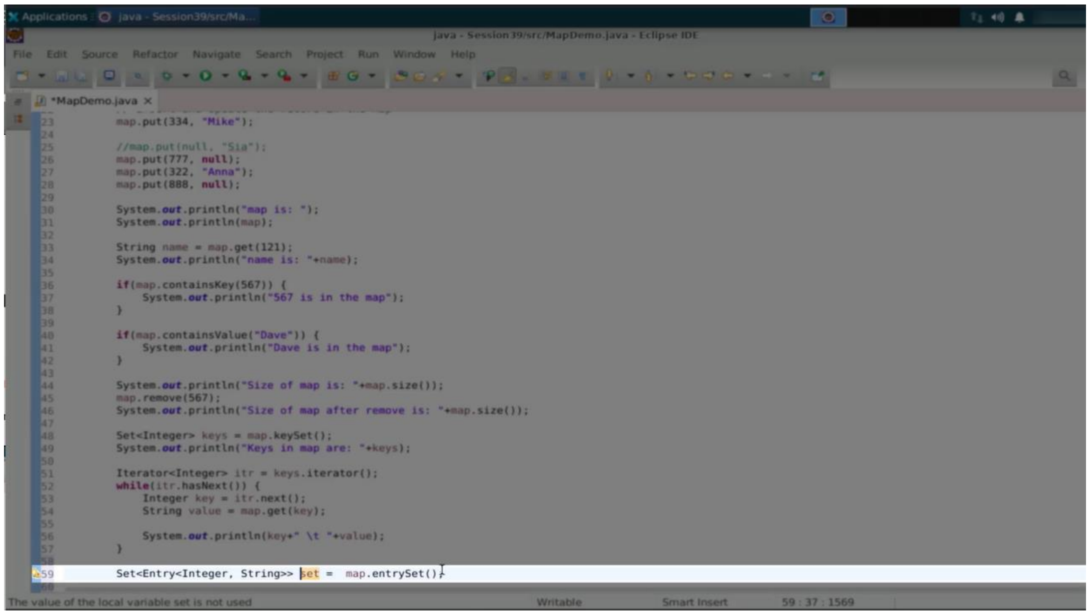


```java
        map.put(334, "Mike");

        //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());

        Set<Integer> keys = map.keySet();
        System.out.println("Keys in map are: "+keys);

        Iterator<Integer> itr = keys.iterator();
        while(itr.hasNext()) {
            Integer key = itr.next();
            String value = map.get(key);

            System.out.println(key+" \t "+value);
        }

    }
```

```
<terminated> MapDemo [Java Application] /usr/eclipse/plugins/org.eclipse.ju
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()**

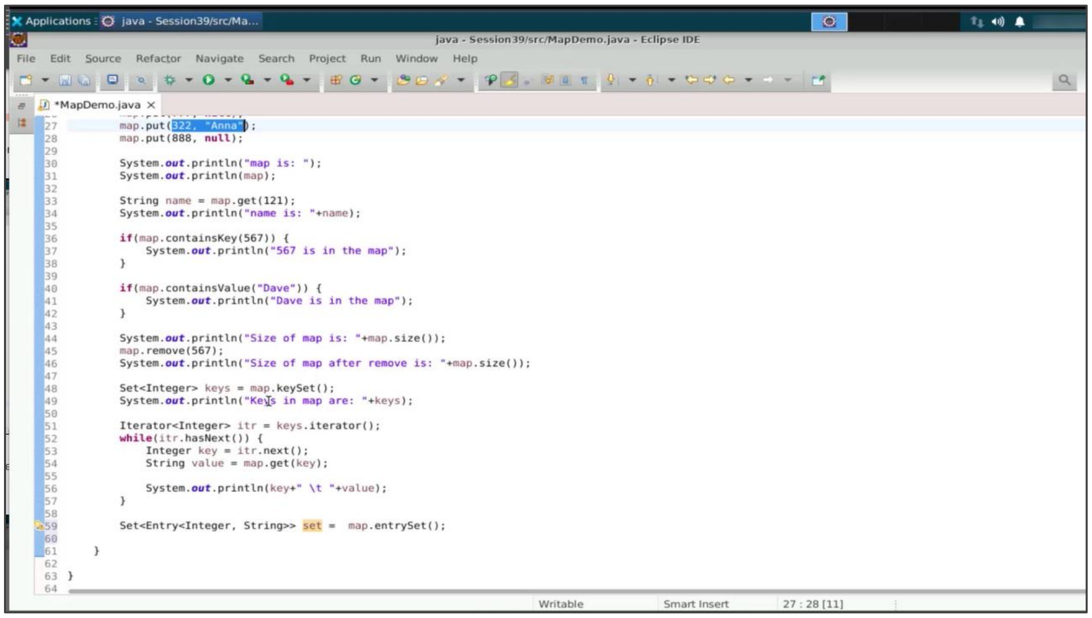


```java
        map.put(334, "Mike");

        //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());

        Set<Integer> keys = map.keySet();
        System.out.println("Keys in map are: "+keys);

        Iterator<Integer> itr = keys.iterator();
        while(itr.hasNext()) {
            Integer key = itr.next();
            String value = map.get(key);

            System.out.println(key+" \t "+value);
        }

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

```
The value of the local variable set is not used     Writable     Smart Insert     59 : 37 : 1569
```

8.2 Assign the entry set to a variable:

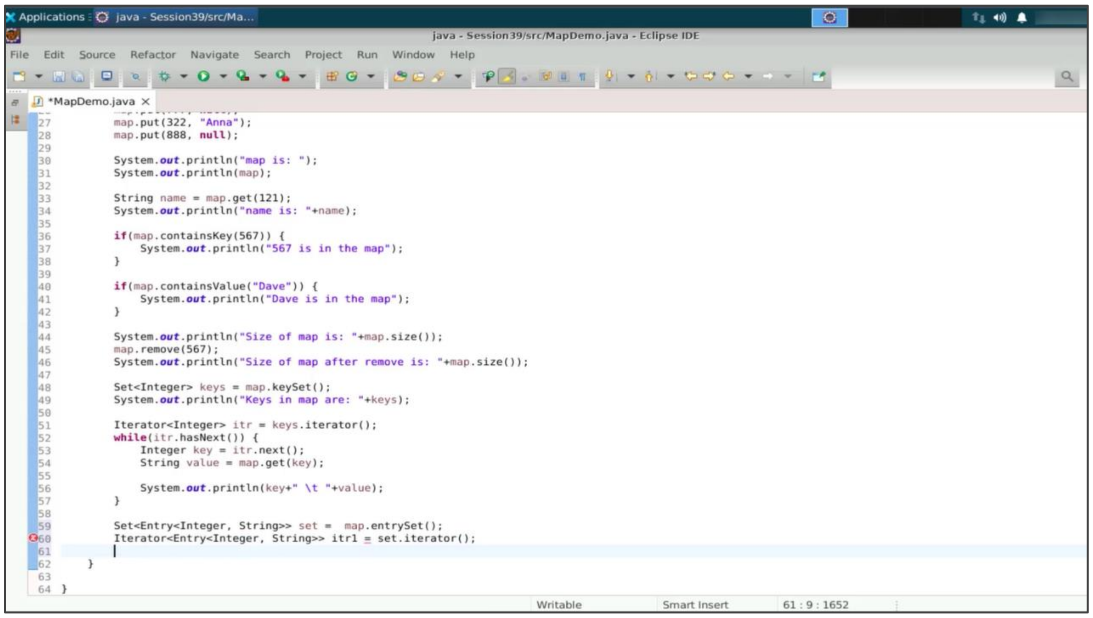
**Set<Map.Entry<Integer, String>> entrySet = map.entrySet();**


Applications
java - Session39/src/Ma...
java - Session39/src/MapDemo.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
*MapDemo.java
Writable
Smart Insert
27 : 28 [11]

```
        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());

        Set<Integer> keys = map.keySet();
        System.out.println("Keys in map are: "+keys);

        Iterator<Integer> itr = keys.iterator();
        while(itr.hasNext()) {
            Integer key = itr.next();
            String value = map.get(key);

            System.out.println(key+" \t "+value);
        }

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

    }

}
```

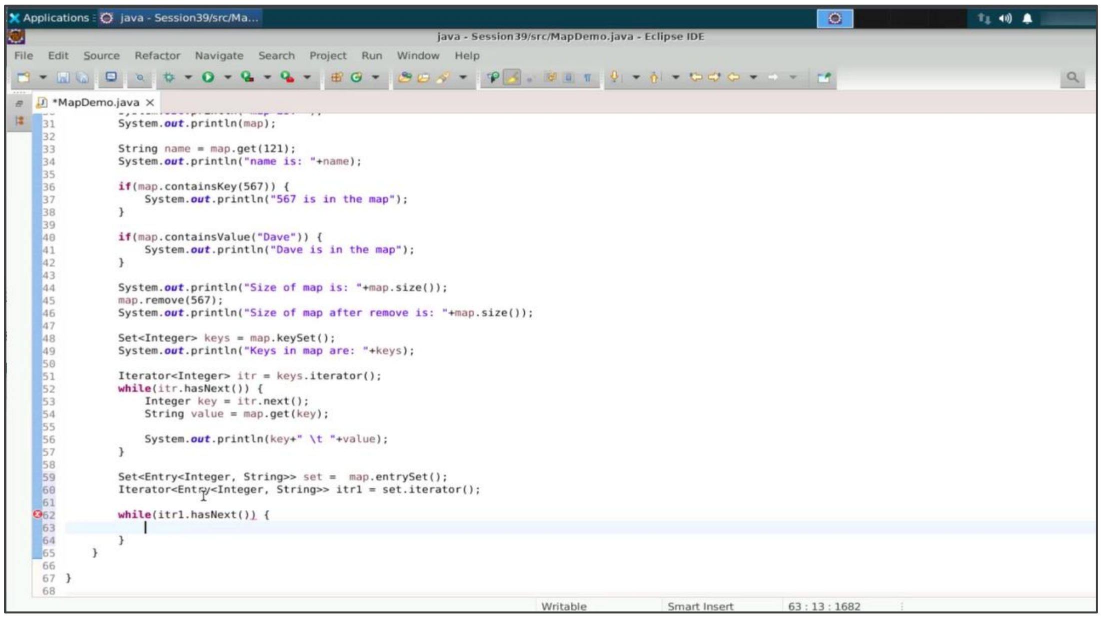
8.3 Create an iterator for the entry set:

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


Applications
java - Session39/src/Ma...
java - Session39/src/MapDemo.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
*MapDemo.java
Writable
Smart Insert
61 : 9 : 1652

```
        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());

        Set<Integer> keys = map.keySet();
        System.out.println("Keys in map are: "+keys);

        Iterator<Integer> itr = keys.iterator();
        while(itr.hasNext()) {
            Integer key = itr.next();
            String value = map.get(key);

            System.out.println(key+" \t "+value);
        }

        Set<Entry<Integer, String>> set =  map.entrySet();
        Iterator<Entry<Integer, String>> itr1 = set.iterator();

    }

}
```

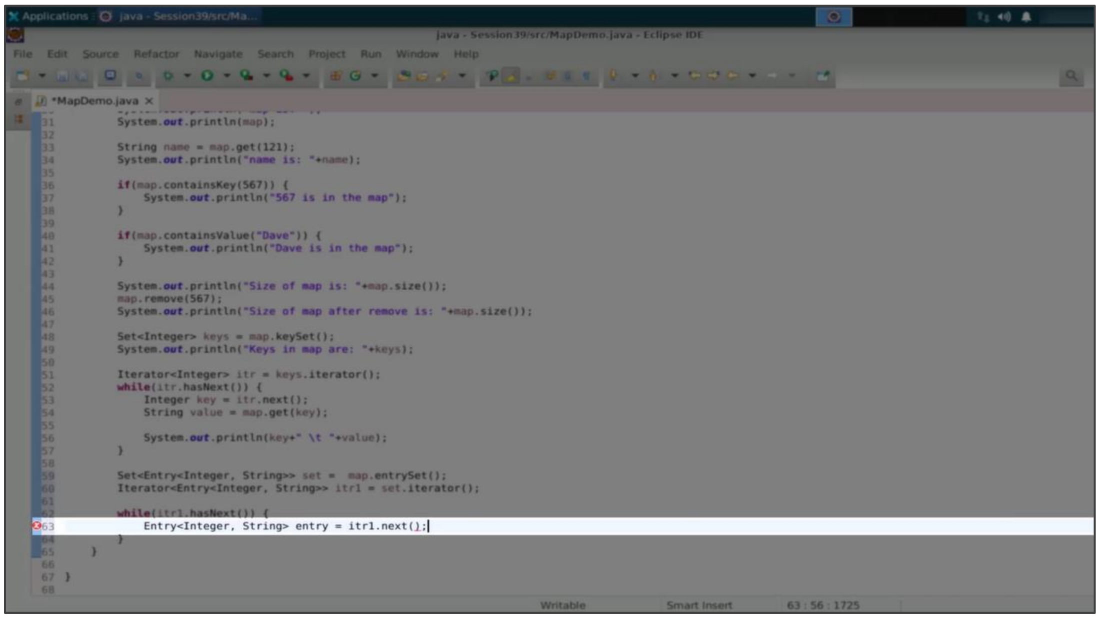
8.4 Use a while loop with **itr1.hasNext()**



```
        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());

        Set<Integer> keys = map.keySet();
        System.out.println("Keys in map are: "+keys);

        Iterator<Integer> itr = keys.iterator();
        while(itr.hasNext()) {
            Integer key = itr.next();
            String value = map.get(key);

            System.out.println(key+" \t "+value);
        }

        Set<Entry<Integer, String>> set =  map.entrySet();
        Iterator<Entry<Integer, String>> itr1 = set.iterator();

        while(itr1.hasNext()) {

        }
    }

}
```

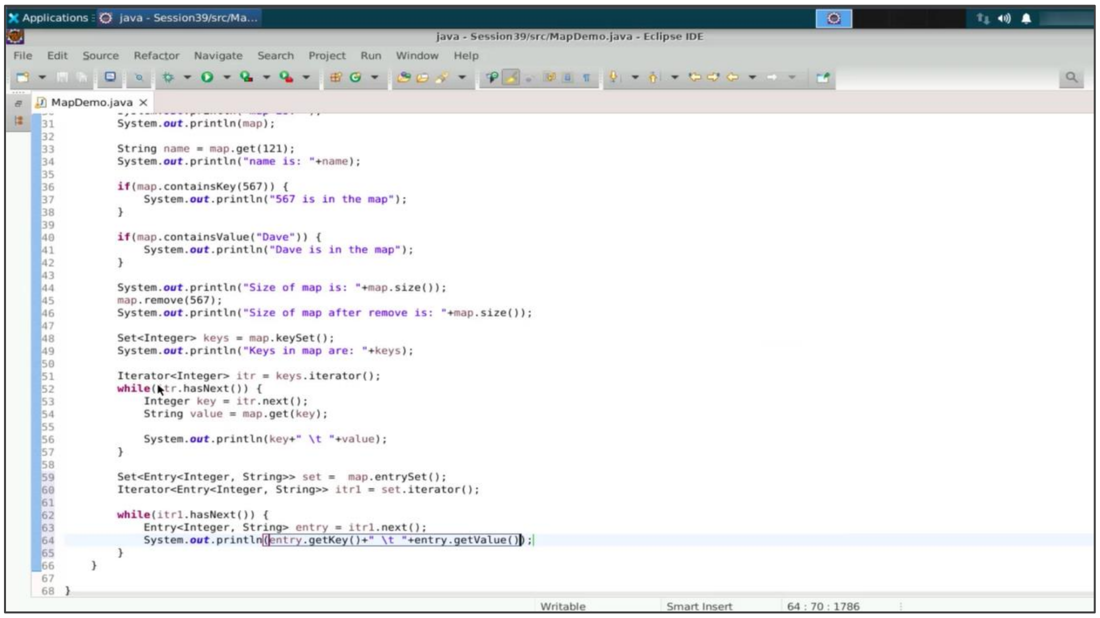
8.5 Retrieve each entry Map:

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



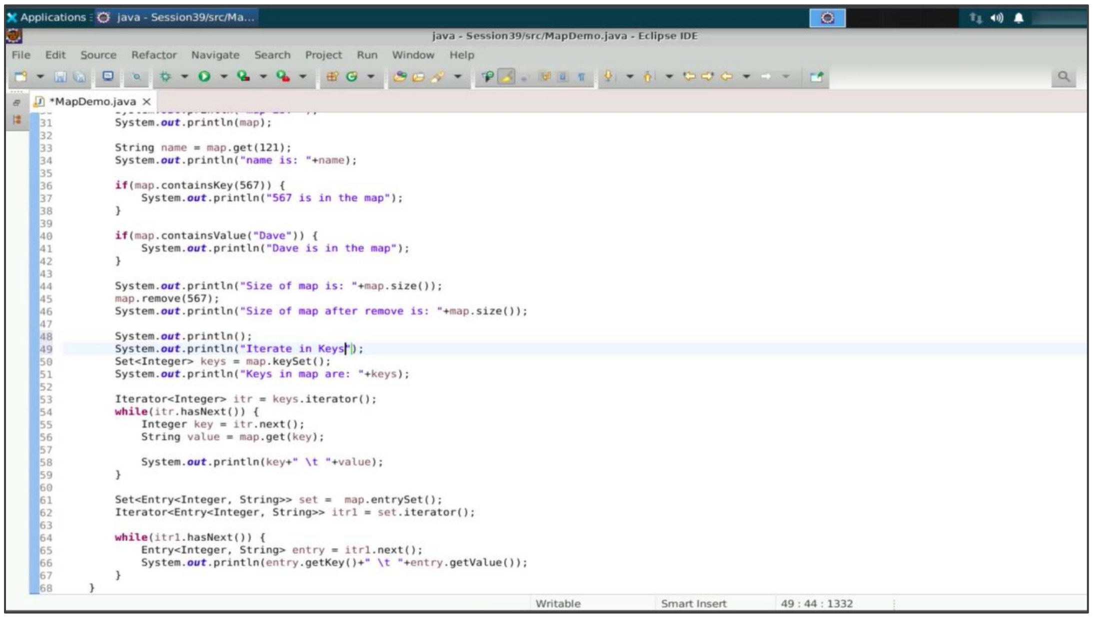
```
        System.out.println(map);

        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());

        Set<Integer> keys = map.keySet();
        System.out.println("Keys in map are: "+keys);

        Iterator<Integer> itr = keys.iterator();
        while(itr.hasNext()) {
            Integer key = itr.next();
            String value = map.get(key);

            System.out.println(key+" \t "+value);
        }

        Set<Entry<Integer, String>> set =  map.entrySet();
        Iterator<Entry<Integer, String>> itr1 = set.iterator();

        while(itr1.hasNext()) {
            Entry<Integer, String> entry = itr1.next();
        }
    }

}
```

## 8.6 Print the key-value pairs



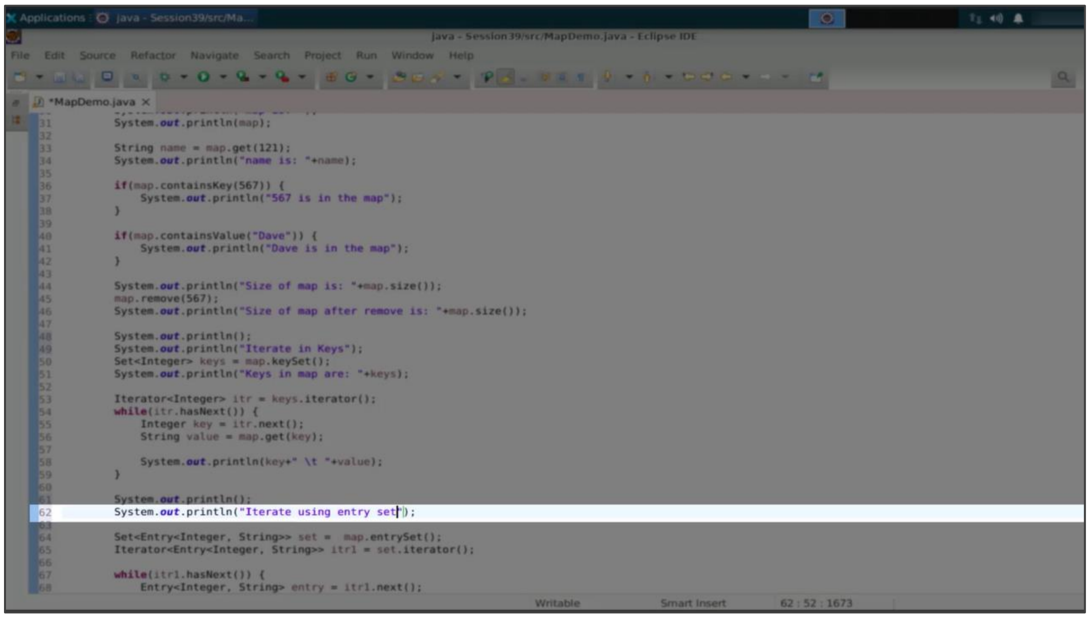
```
        System.out.println(map);

        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());

        Set<Integer> keys = map.keySet();
        System.out.println("Keys in map are: "+keys);

        Iterator<Integer> itr = keys.iterator();
        while(itr.hasNext()) {
            Integer key = itr.next();
            String value = map.get(key);

            System.out.println(key+" \t "+value);
        }

        Set<Entry<Integer, String>> set =  map.entrySet();
        Iterator<Entry<Integer, String>> itr1 = set.iterator();

        while(itr1.hasNext()) {
            Entry<Integer, String> entry = itr1.next();
            System.out.println(entry.getKey()+" \t "+entry.getValue());
        }
    }

}
```

## 8.7 Print **Iterate in Keys**



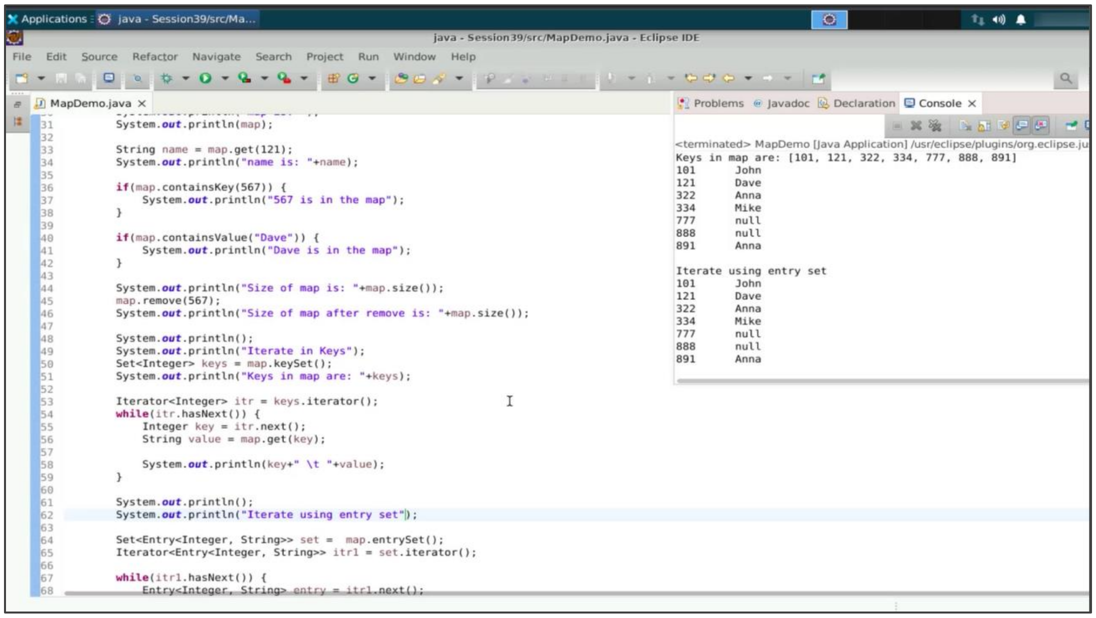
```
        System.out.println(map);

        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());

        System.out.println();
        System.out.println("Iterate in Keys");
        Set<Integer> keys = map.keySet();
        System.out.println("Keys in map are: "+keys);

        Iterator<Integer> itr = keys.iterator();
        while(itr.hasNext()) {
            Integer key = itr.next();
            String value = map.get(key);

            System.out.println(key+" \t "+value);
        }

        Set<Entry<Integer, String>> set =  map.entrySet();
        Iterator<Entry<Integer, String>> itr1 = set.iterator();

        while(itr1.hasNext()) {
            Entry<Integer, String> entry = itr1.next();
            System.out.println(entry.getKey()+" \t "+entry.getValue());
        }
    }
```

8.8 Print **Iterate using entry set**



```
        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());

        System.out.println();
        System.out.println("Iterate in Keys");
        Set<Integer> keys = map.keySet();
        System.out.println("Keys in map are: "+keys);

        Iterator<Integer> itr = keys.iterator();
        while(itr.hasNext()) {
            Integer key = itr.next();
            String value = map.get(key);

            System.out.println(key+" \t "+value);
        }

        System.out.println();
        System.out.println("Iterate using entry set");

        Set<Entry<Integer, String>> set =  map.entrySet();
        Iterator<Entry<Integer, String>> itr1 = set.iterator();

        while(itr1.hasNext()) {
            Entry<Integer, String> entry = itr1.next();
```

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

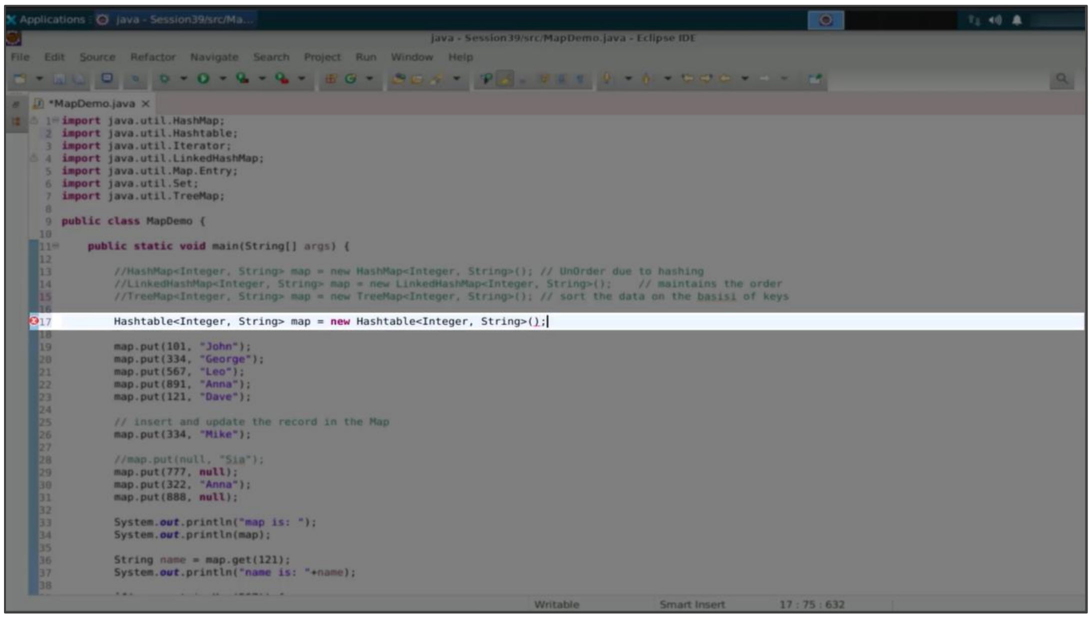


```
        System.out.println(map);

        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());

        System.out.println();
        System.out.println("Iterate in Keys");
        Set<Integer> keys = map.keySet();
        System.out.println("Keys in map are: "+keys);

        Iterator<Integer> itr = keys.iterator();
        while(itr.hasNext()) {
            Integer key = itr.next();
            String value = map.get(key);

            System.out.println(key+" \t "+value);
        }

        System.out.println();
        System.out.println("Iterate using entry set");

        Set<Entry<Integer, String>> set =  map.entrySet();
        Iterator<Entry<Integer, String>> itr1 = set.iterator();

        while(itr1.hasNext()) {
            Entry<Integer, String> entry = itr1.next();
```

```
<terminated> MapDemo [Java Application] /usr/eclipse/plugins/org.eclipse.ju
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

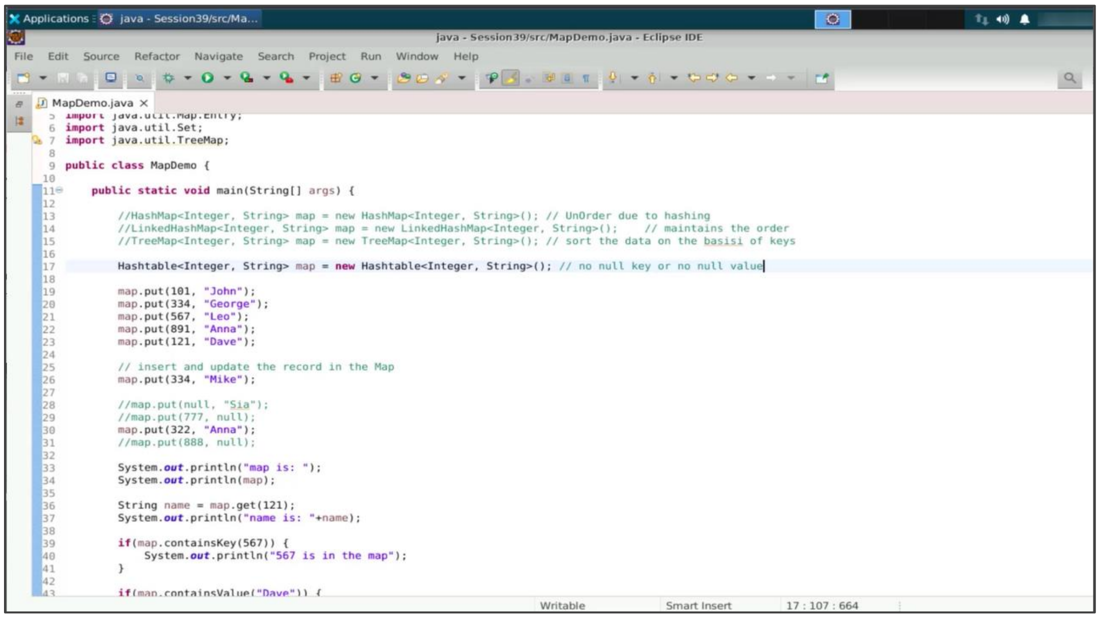
9.1 Create a Hashtable named **map**


Applications
java - Session39/src/Ma...
java - Session39/src/MapDemo.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
*MapDemo.java
Writable
Smart Insert
17 : 75 : 632

```java
import java.util.HashMap;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.Map.Entry;
import java.util.Set;
import java.util.TreeMap;

public class MapDemo {

    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 basisi of keys

        Hashtable<Integer, String> map = new Hashtable<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);

        String name = map.get(121);
        System.out.println("name is: "+name);
```

Note: Hashtable does not allow null keys or values.

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



```java
import java.util.Map.Entry;
import java.util.Set;
import java.util.TreeMap;

public class MapDemo {

    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 basisi of keys

        Hashtable<Integer, String> map = new Hashtable<Integer, String>(); // no null key or no null value

        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")) {
```

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