HashSet programs :

```java
package com.ravi.hash_set_demo;

import java.util.HashSet;

public class HashSetDemo2
 {
    public static void main(String[] argv)
    {
        HashSet<String> hs = new HashSet<>();
        hs.add("Ravi");
        hs.add("Vijay");
        hs.add("Ravi");
        hs.add("Ajay");
        hs.add("Palavi");
        hs.add("Sweta");
        hs.add(null);
        hs.add(null);
        hs.forEach(str -> System.out.println(str));

    }
}
```

HashSet does not maintain any order while iterating the elements from the Collection.

```java
package com.ravi.hash_set_demo;

import java.util.Arrays;
import java.util.HashSet;

public class HashSetDemo3
{
    public static void main(String[] args)
    {
        Boolean values[] = new Boolean[5];

        HashSet<Object> hs = new HashSet<>();
        values[0] = hs.add(12);
        values[1] = hs.add(12);
        values[2] = hs.add(13);
        values[3] = hs.add(13);
        values[4] = hs.add(new String("Java"));

        System.out.println(Arrays.toString(values));

    }

}
```

```
Note : List interface add() method will always return true but Set interface add() method
       method will return true Or false based on the type of Object i.e Object is
       identical OR not
```

```java
 package com.ravi.hash_set_demo;

 //add, delete, display and exit

 import java.util.HashSet;
 import java.util.Scanner;

 public class HashSetDemo4
 {
     public static void main(String[] args)
     {
      HashSet<String> hashSet = new HashSet<>();
      Scanner scanner = new Scanner(System.in);

      while (true)
      {
          System.out.println("Options:");
          System.out.println("1. Add element");
          System.out.println("2. Delete element");
          System.out.println("3. Display HashSet");
          System.out.println("4. Exit");

          System.out.print("Enter your choice (1/2/3/4): ");
          int choice = Integer.parseInt(scanner.nextLine());

          switch (choice)
         {
              case 1:
                  System.out.print("Enter the element to add: ");
                  String elementToAdd = scanner.nextLine();
                  if (hashSet.add(elementToAdd))
                {
                     System.out.println("Element added successfully.");
                 }
                else
                {
                     System.out.println("Element already exists in the HashSet.");
                 }
                 break;
                 case 2:
                 System.out.print("Enter the element to delete: ");
                 String elementToDelete = scanner.nextLine();
                 if (hashSet.remove(elementToDelete))
                {
                     System.out.println("Element deleted successfully.");
                 }
                else
                {
                     System.out.println("Element not found in the HashSet.");
                 }
                 break;
                 case 3:
                 System.out.println("Elements in the HashSet:");
                 hashSet.forEach(System.out::println);
                 break;
                 case 4:
                 System.out.println("Exiting the program.");
                 scanner.close();
                 System.exit(0);
                 default:
                 System.out.println("Invalid choice. Please try again.");
          }

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

```java
package com.ravi.hash_set_demo;

import java.util.HashSet;

public class HashSetDemo5
{
    public static void main(String[] args)
    {
        HashSet<String> hs1 = new HashSet<String>();
        hs1.add(new String("India"));
        hs1.add(new String("India"));
        System.out.println(hs1.size()); //1


        HashSet<StringBuffer> hs2 = new HashSet<StringBuffer>();
        hs2.add(new StringBuffer("Hyd"));
        hs2.add(new StringBuffer("Hyd"));
        System.out.println(hs2.size()); //2

    }
}
```

From the above program, It is clear that hashCode() and equals() methods are overridden in String class but not overridden in StringBuffer class.

Map<K,V> interface :

----

* It is a separate interface, It does not have any concern with Collection<E> interface.

* It does not have any concern so It does not support any method of Collection<E> interface.

* It is mainly used to work with **Group of objects** i.e key and value pair.

* Here key and value both are objects.

* Map<K,V> does not accept **duplicate key but value may be duplicate.**

key	value
raj@gmail.com	raj$&^&^*
ravi@gmail.com	ravi*&^$%
scott@gmail.com	scott*&^&*
aryan@gmail.com	aryan~~!#$#

Entry

* In this diagram total 4 entries are available

* An Entry in the map reprsents the key and value pair so, Each entry represents a paricular key and value combination.

* Actually Entry is an interface which is defined inside Map interface.

Example :

```
public interface Map<K,V>
{
    public interface Entry<K,V>
    {
        //key and value pair
    }
}
```

* In .java file, We can represent the Entry by using Map.Entry on the other hand the same Entry we can represent in the .class file by using Map$Entry.class.

* In Map, If we accept **duplicate key** then old key value will be replaced by new key value.

* We cannot use Iterator<E> and Spliterator<E> directly on the Map (Group of Objects) but If we convert the Map into Collection then we can use Iterator and Spliterator.

* default forEach(BiConsumer<T,U> b) methdo is added in Map<K,V> interface from JDK 1.8V to fetch key and value as an Object.

Hierarchy of Map<K,V> interface :