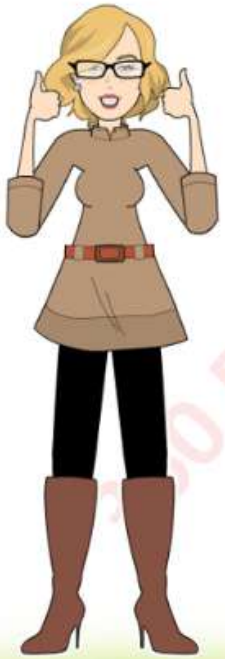
Set Interface

Click to Continue



Set is an interface that holds unique elements.

- It allows one element to be null.
- All the methods are inherited from Collection Interface.
- ***Sorted Sets*** maintains elements in sorted order.

Examples:

1. HashSet
2. TreeSet

HashSet

Click to Continue

HashSet stores values similar to **ArrayList** but does not store duplicate values.

Syntax:

```
Set ObjectName=new HashSet();
```

Illustration :

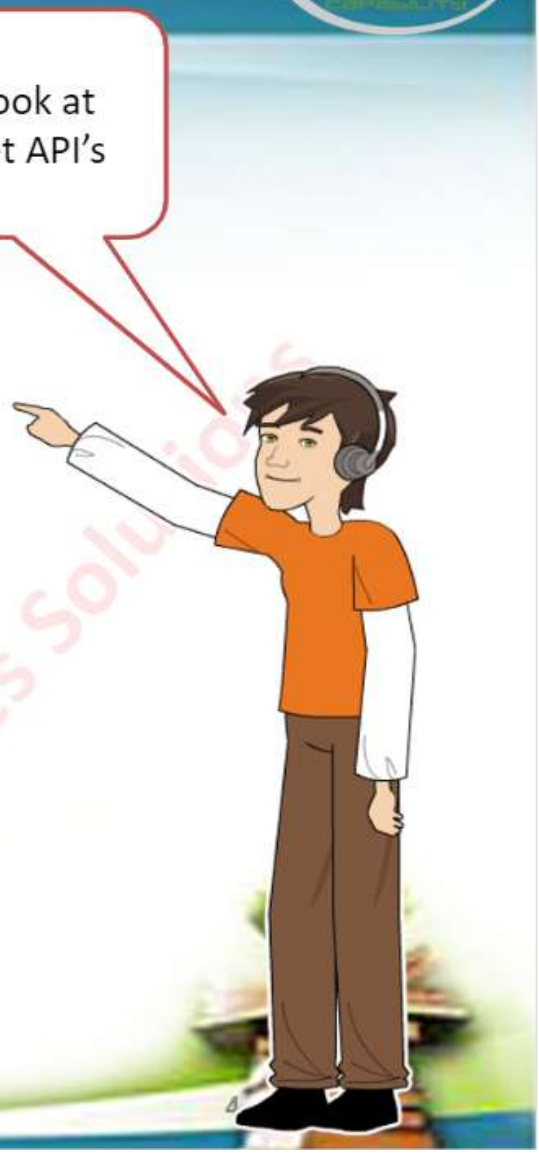
```
Set studentNames=new HashSet();
```

This creates a set which will store unique student names.

HashSet API

Click to Continue

Method	Description
boolean add(Object element)	Adds the specified element to this set if it is not already present.
void clear()	Removes all of the elements from this set.
boolean contains(Object o)	Returns true if this set contains the specified element.
boolean isEmpty()	Returns true if this set contains no elements.
boolean remove(Object o)	Removes the specified element from this set if it is present.
int size()	Returns the number of elements in this set.



Adding elements into HashSet

Click to Continue

add() method used to add an element. Use ***addAll()*** method to add a entire collection object.

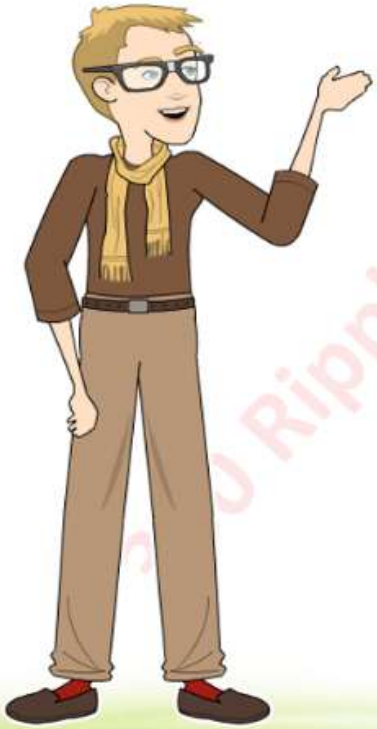


Illustration:

```
studentNames.add("Ramesh");
```

Adds two String names in the set

```
studentNames.add("Rajesh");
```

```
studentNames.add("Ramesh")
```

Since Ramesh already exists the name will not be added

```
students.addAll(studentNames);
```

Adds all the items of Set studentNames to **students**

Try it Out - HashSet

Click to Continue

Let us create a ***HashSet*** and add elements into it.

Scenario 1: Develop a method **loadStudentNames** that accepts the names of three students as three string parameter and add them to an HashSet.

Scenario 2: To the above scenario add the following names and print the Hashset and verify how many elements got added. ***Tim,Tim,John.***

Scenario 3 : Create an method ***loadEvenNumbers*** which accepts a int N and iterates through 'N' even numbers add each number in the ArrayList and returns the list..

Scenario 4 : Create a method ***copyEvenNumbers*** which needs to copy the even numbers created in scenario # 2 in another List

Try it out – Program Execution

Click to Continue

Create an object of the class and execute all the three methods

```
import java.util.Set;

public class HashSetMain {

    /**
     * @param args
     */
    public static void main(String[] args) {

        HashSetDemo exc = new HashSetDemo();
        exc.loadStudentNames("Tim", "John", "Tim");

        Set even = exc.populateEvenNumber(10);
        System.out.println("Set" + even);

        Set evenCopy = exc.copyEvenNumbers();
        System.out.println("Set Copy" + evenCopy);
    }

}
```

If the set variable exc is printed using Println statement the names will be printed. Tim will be printed only once, since set does not allow duplicates.

The populateEvenNumber method should be invoked which populates the evenNumber class variable.

Copy then copies the elements of the variable evenNumber into a new Set and returns it. The returned Set is stored in a new variable evenCopy and printed.