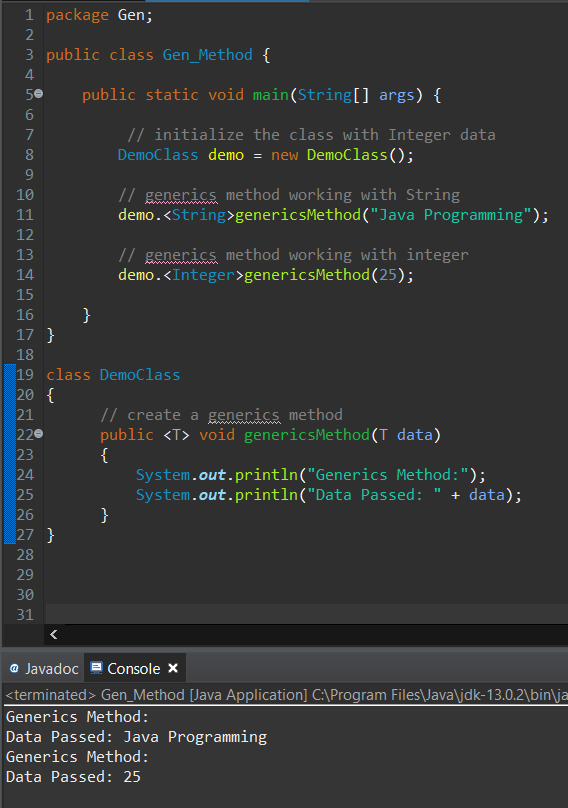


Here the **Generic Class** name is **GenericsClass.**

It has a constructor and a method.

We are creating the object and calling the constructor of the Generic Class of **Integer** type and then the **String** type.

Then we are calling the method which is instantiated.

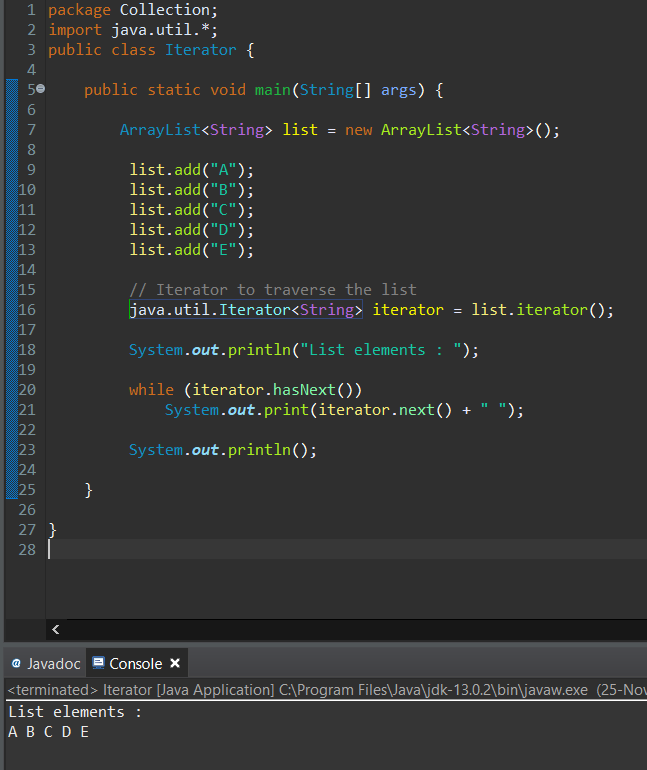


Similar to the generics class, we can also create a method that can be used with any type of data.

Such a method is known as Generics Method.

We have defined a generic method in **DemoClass.**

We have created the object of the demo class and then called the method with prefixing its data type.

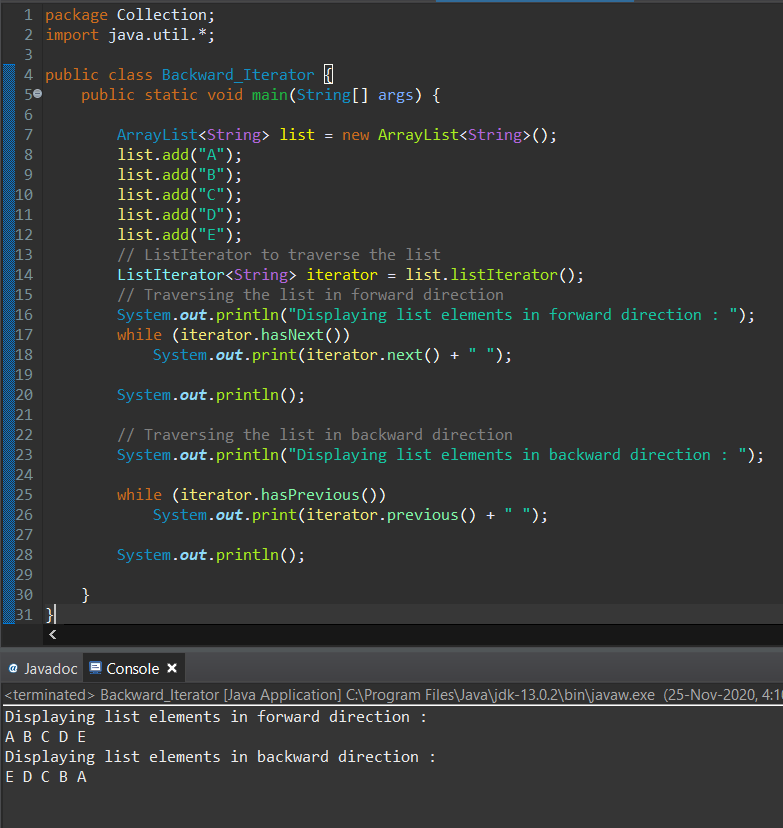


Iterator is commonly used to iterate through the data structure.

We have created the Iterator of type String.

We have used **hasNext()** to check whether the next element exist or not.

**Next()** is used to print the Next Element.



We have created the Iterator of type String.

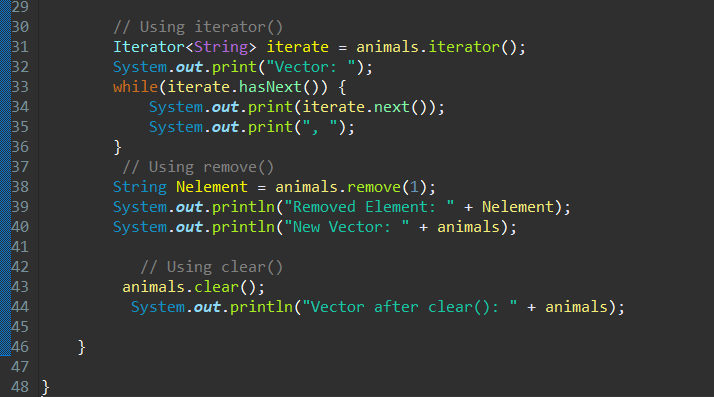
We have used **hasNext()** to check whether the next element exist or not.

**next()** is used to print the Next Element.

We have used **hasPrevious()** to check whether the previous element exist or not.

**previous()** is used to print the previous Element.





**Vector Class**Vector implements a dynamic array that means it can grow or shrink as required.

We have created two vectors **animals** and **mammals** of type String.

We have used **hasNext()** to check whether the next element exist or not.

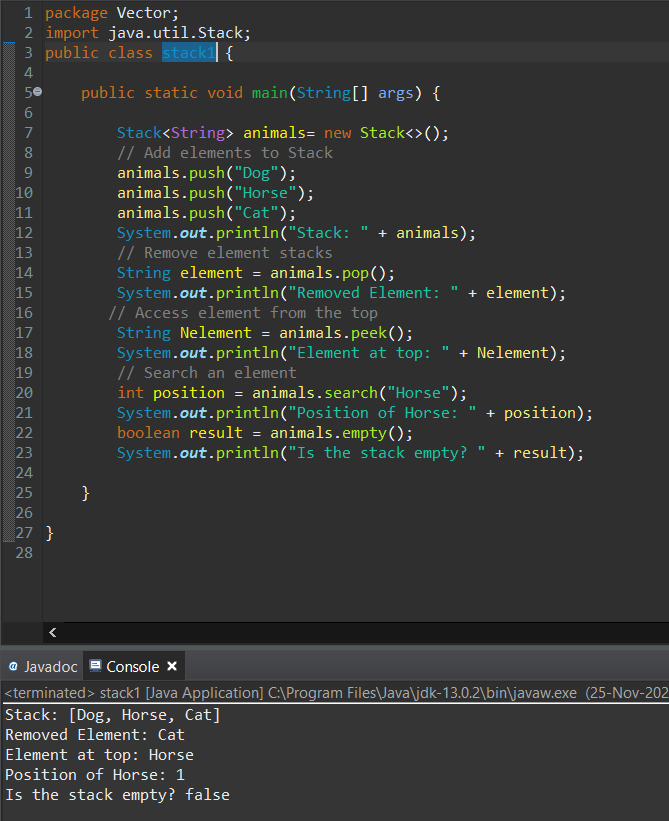
**next()** is used to print the Next Element.

add() method is used to add the Elements.

Add(i , element) adds element to the particular index.

Remove() is used to remove the element.

Clear() is used to delete all elements.



Stack provides the functionality of the stack data structure.

Push() is used to add elements.

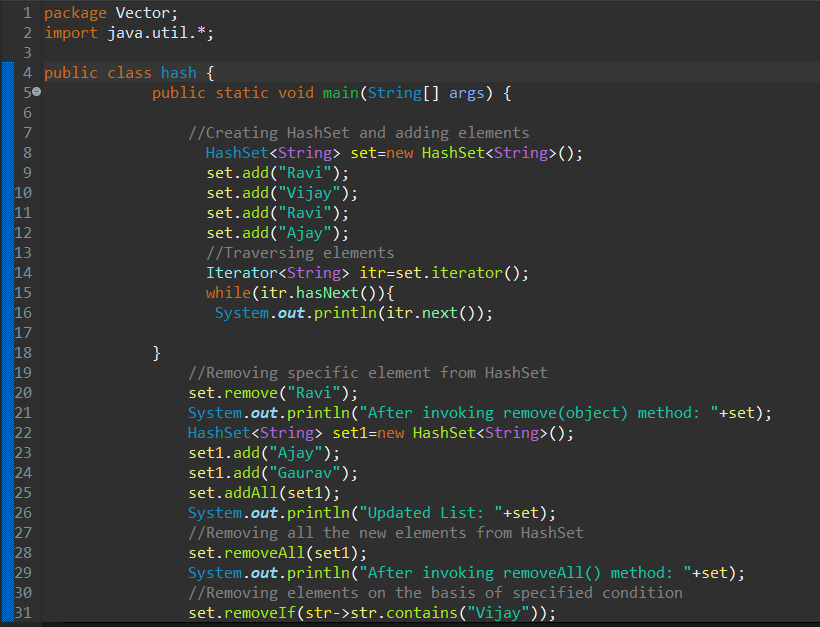
Pop() is used to remove elements.

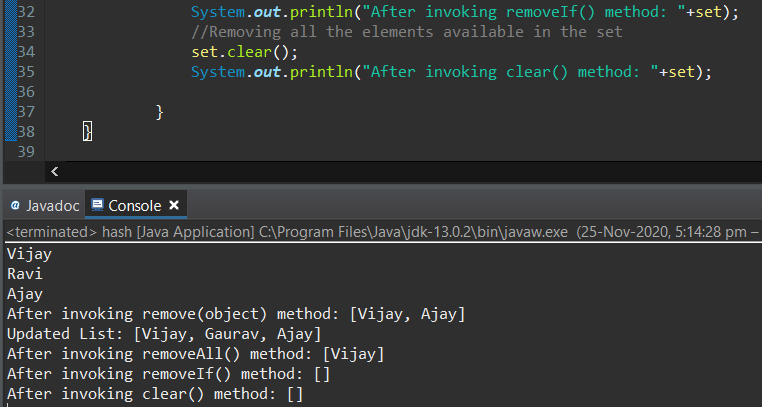
Peek() is used to get the top most element

Search() returns the index of the element which is passed as a parameter if its found in the Stack.

Empty() checks whether the Stack is empty or not.

Returns true if its empty else false.





If you don’t want to maintain insertion order but want store unique objects.

HashSet does not maintain insertion order.

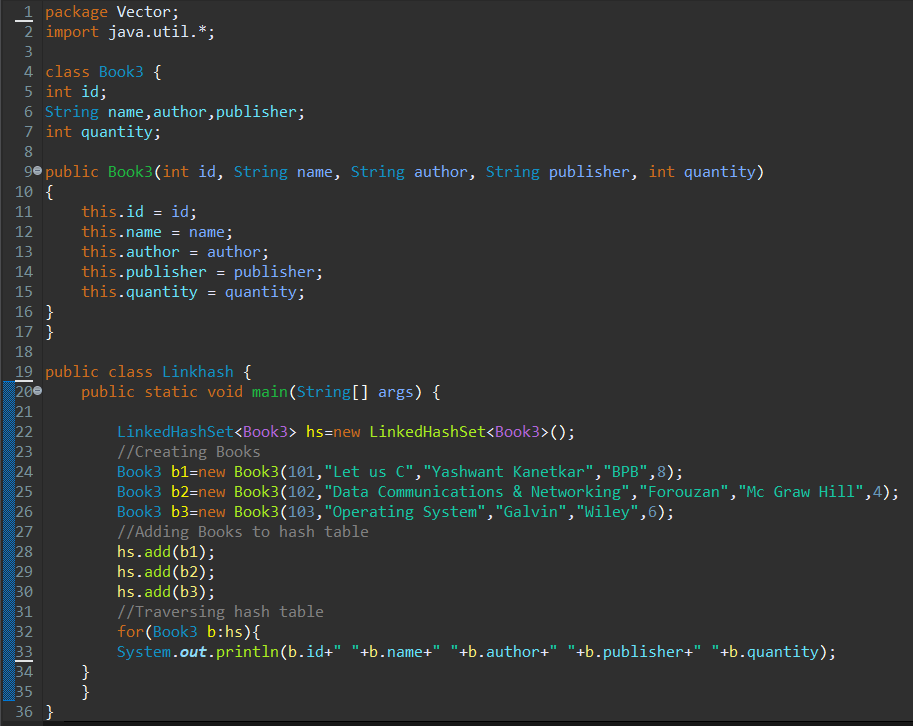
Add() is used to add the elements.

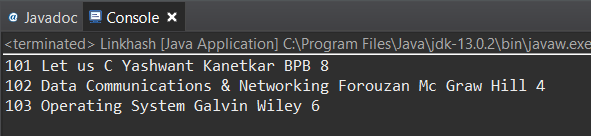
Addall() is used to add all elements from one set to another.

removeAll() removes all elements from the set1 if they are in set2.

Removeif() removes element from the set if its there.

Clear() removes all elements from the set.





If you want to maintain insertion order of elements then you can use LinkedHashSet.

LinkedHashSet maintains insertion order of objects.

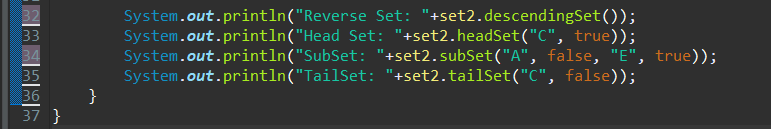
We have created the class Book with 5 members in it.

We have called the LinkHashSet and invoked it of the type class Book3.

We have created three more objects of class Book3 with different data and passed the arguments to the constructor.

Then we have added the Book objects to the LinkedHashSet.

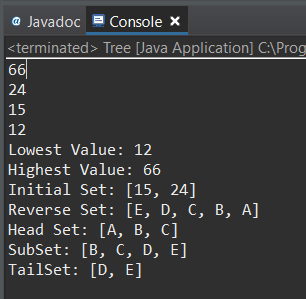


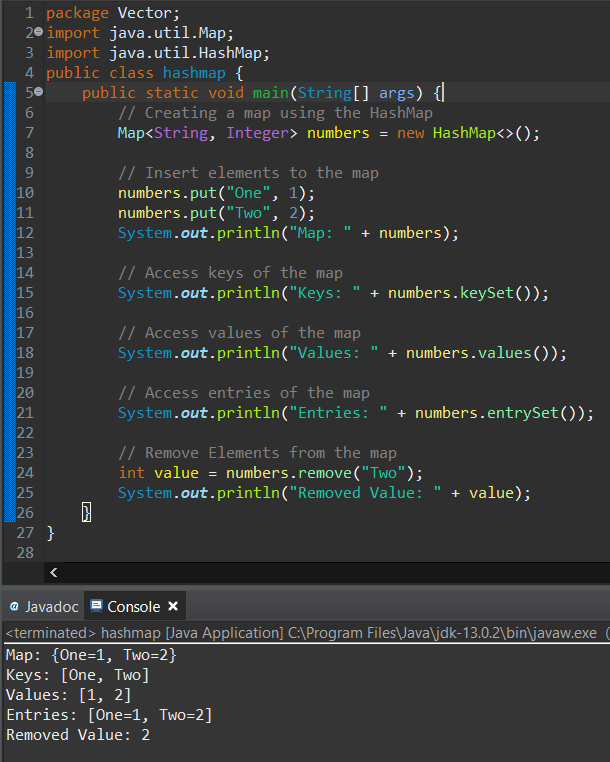


If you want to sort the elements according to some Comparator then use TreeSet.

TreeSet orders of the elements according to supplied Comparator.

Default, It’s objects will be placed in their natural ascending order.





A map contains values on the basis of key, i.e. key and value pair. Each key and value pair is known as an entry.

A Map contains unique keys.

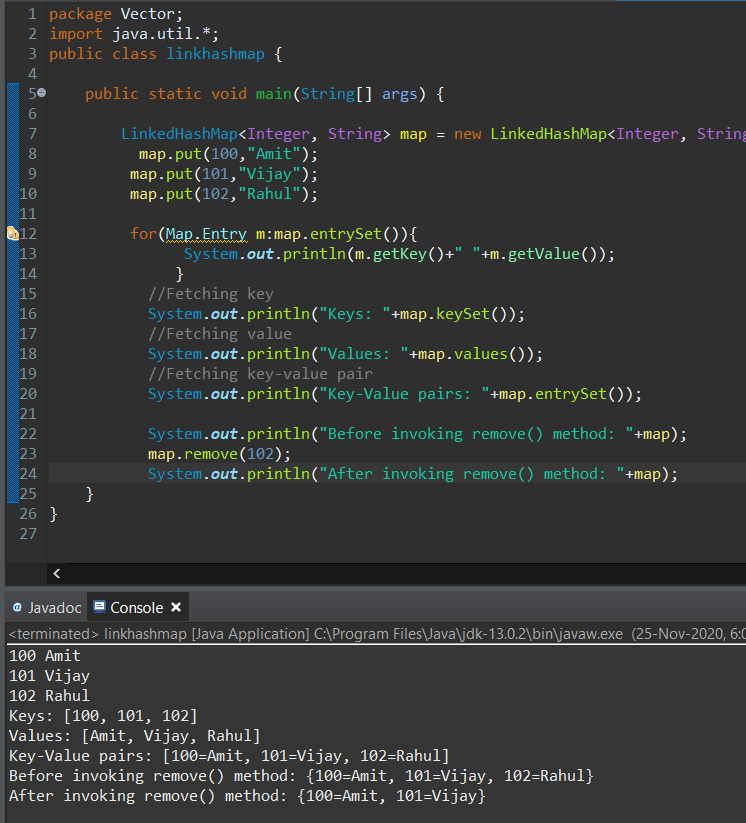
A Map is useful if you have to search, update or delete elements on the basis of a key.

Keyset() is used to display the keys.

Values() is used to display the values.

Entryset() is used to display the entries.

Remove(key) is used to delete the value along with the key.



This class extends HashMap and maintains a linked list of the entries in the map, in the order in which they were inserted.

This allows insertion-order iteration over the map.

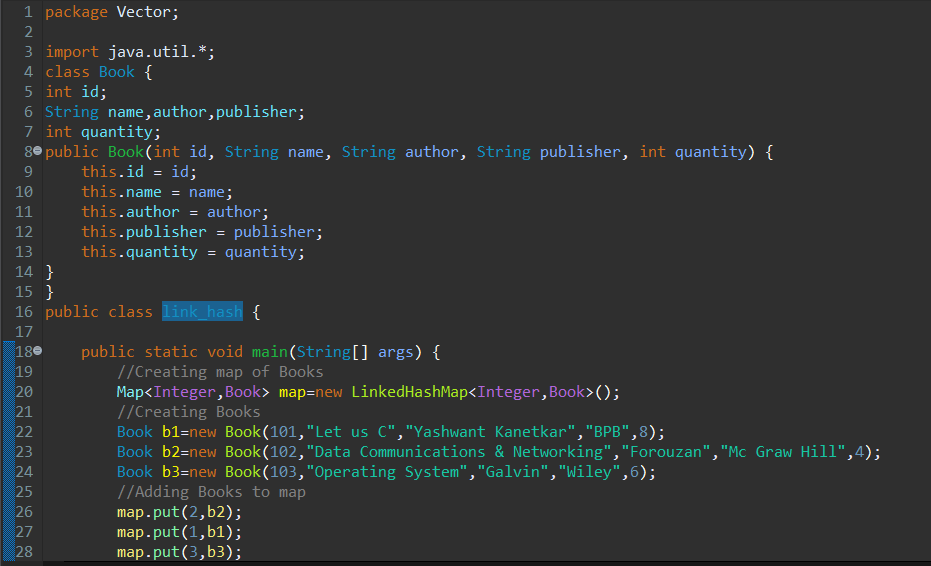
That is, when iterating a LinkedHashMap, the elements will be returned in the order in which they were inserted.

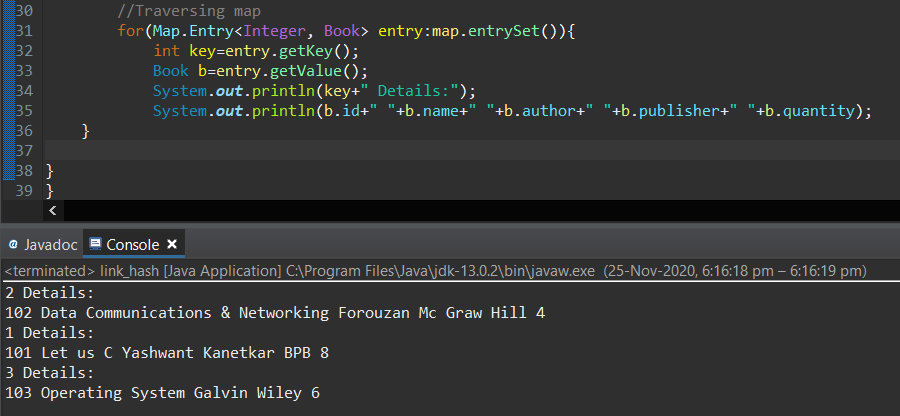
Keyset() is used to display the keys.

Values() is used to display the values.

Entryset() is used to display the entries.

Remove(key) is used to delete the value along with the key.





This class extends HashMap and maintains a linked list of the entries in the map, in the order in which they were inserted.

This allows insertion-order iteration over the map.

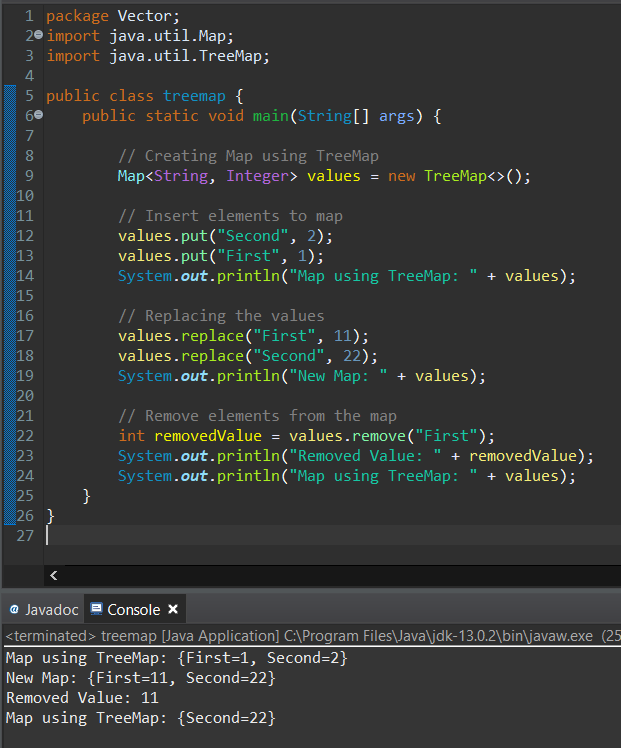
We have created the class Book with 5 members in it.

We have called the LinkHashMap and invoked it of the type class Book.

We have created three more objects of class Book with different data and passed the arguments to the constructor.

Then we have added the Book objects to the LinkedHashMap.

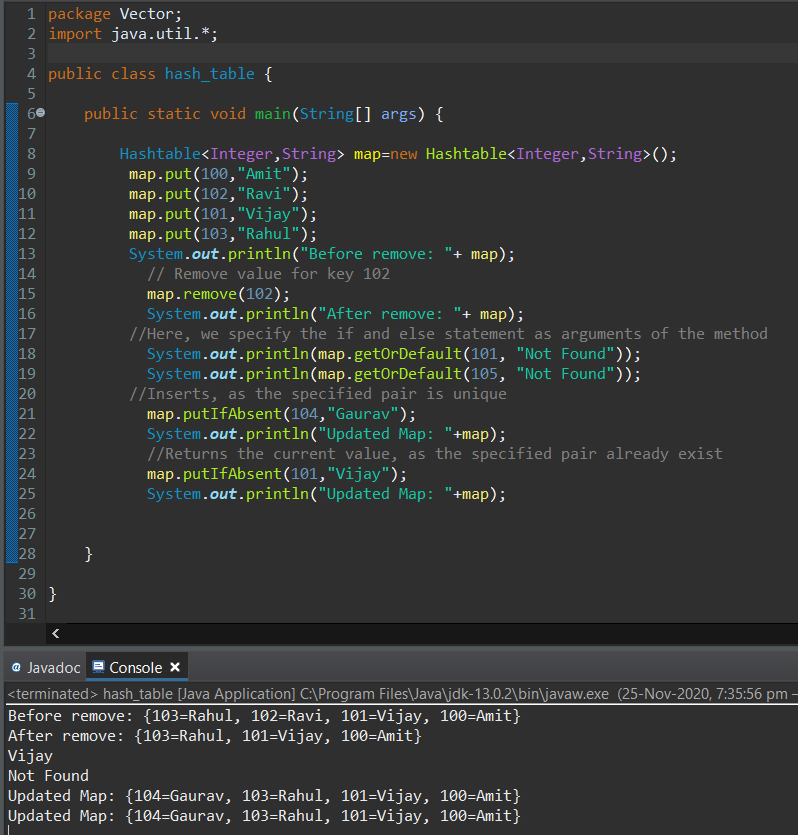
Though we have created the object of Book in order we are inserting it with some other key value.



Put() inserts the element in the key.

Replace() replaces the key with the given value.

Remove() removes the key and the value and it returns the value specific to the given key.



Hash function is used to map a given value with a particular key for faster access of elements.

The efficiency of mapping depends of the efficiency of the hash function used.

Put() inserts the element in the key.

Remove() removes the key and the value.

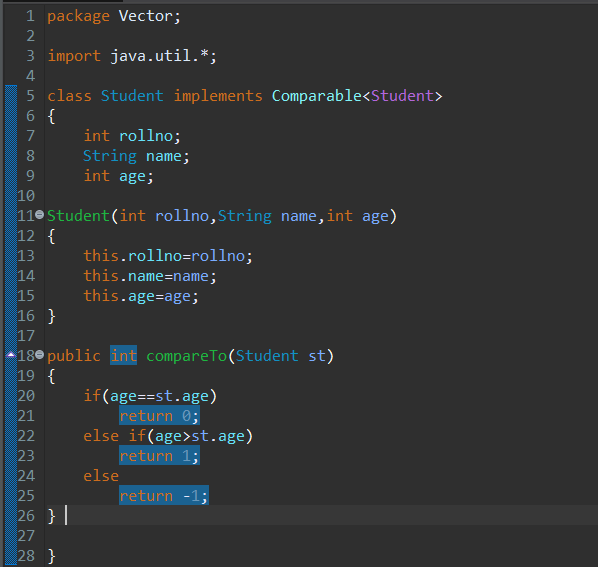
The getOrDefault(Object key, V defaultValue) method of [Map interface](https://www.geeksforgeeks.org/map-interface-java-examples/), implemented by [HashMap class](https://www.geeksforgeeks.org/java-util-hashmap-in-java/) is used to get the value mapped with specified key.

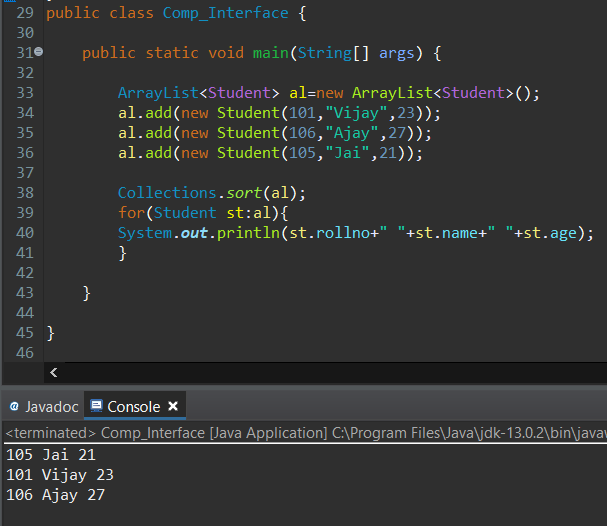
If no value is mapped with the provided key then the default value is returned.

The putIfAbsent(key, value) method of [HashMap](http://www.geeksforgeeks.org/java-util-hashmap-in-java/) class is used map the specified key with the specified value, only if no such mapping exists in this HashMap instance.

104 key is inserted since its not present in the hashmap.

101 already exist. So it wont insert any new key and returns the current vaue.

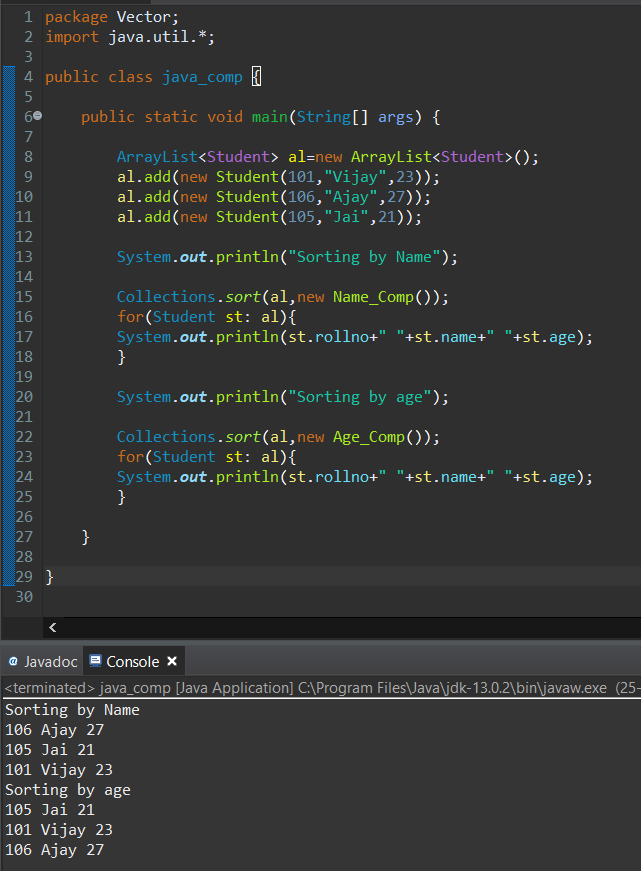


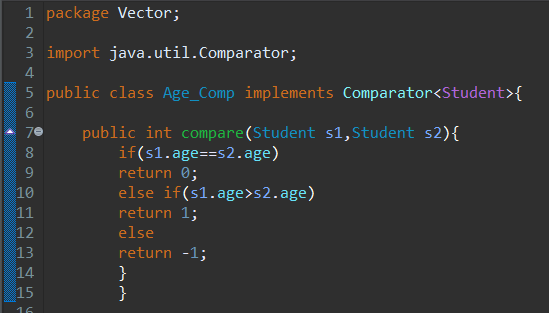


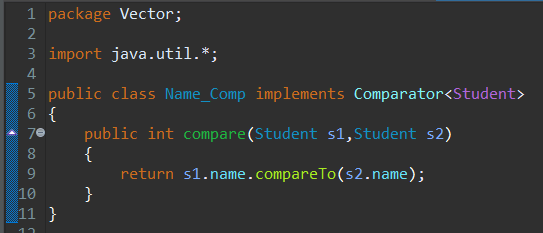
Here we have used Comparable Interface.

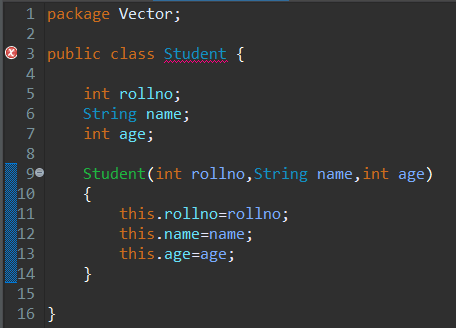
We are sorting it using Age.

We have implemented CompareTo() method and based on the value passed we are comparing and returning the Boolean.









Here we have created 4 classes.

Student class contains 3 data members.

Class Age\_Comp implements Comparator.

Here we have used Comparable Interface.

We are sorting it using Students Age.

We have implemented Compare() method and based on the value passed we are comparing and returning the Boolean.

Class Name\_Comp implements Comparator.

Here we have used Comparable Interface.

We are sorting it using Students Name.

We have implemented CompareTo() method and based on the value passed we are comparing and returning the Boolean.

In Class java\_comp we have created the object of other classes and implemented their constructors.

We are sorting using Name and then using Age.

**Thankyou!!**