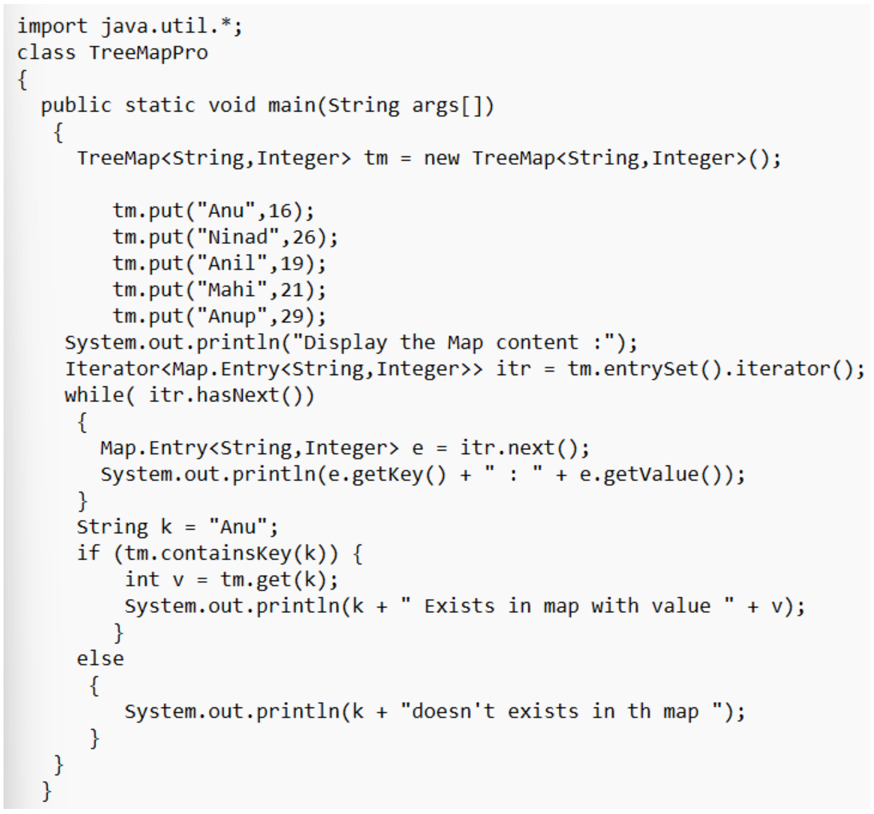
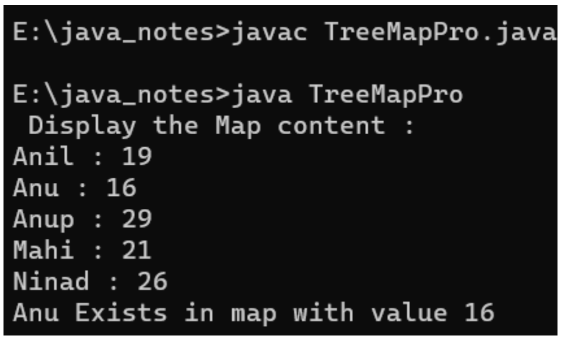
OBJECT ORIENTED PROGRAMMING WITH JAVA 8– LAB

Q1. Create a TreeMap and add some entries to it. Display the map contents using Iterator. Check whether a particular key exists in the map or not. If it is present, display its value.



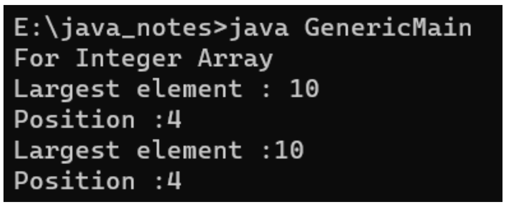
```java
import java.util.*;
class TreeMapPro
{
  public static void main(String args[])
   {
     TreeMap<String,Integer> tm = new TreeMap<String,Integer>();

        tm.put("Anu",16);
        tm.put("Ninad",26);
        tm.put("Anil",19);
        tm.put("Mahi",21);
        tm.put("Anup",29);
    System.out.println("Display the Map content :");
    Iterator<Map.Entry<String,Integer>> itr = tm.entrySet().iterator();
    while( itr.hasNext())
     {
       Map.Entry<String,Integer> e = itr.next();
       System.out.println(e.getKey() + " : " + e.getValue());
     }
     String k = "Anu";
     if (tm.containsKey(k)) {
         int v = tm.get(k);
         System.out.println(k + " Exists in map with value " + v);
        }
     else
      {
         System.out.println(k + "doesn't exists in th map ");
      }
   }
  }
```



```
E:\java_notes>javac TreeMapPro.java

E:\java_notes>java TreeMapPro
 Display the Map content :
Anil : 19
Anu : 16
Anup : 29
Mahi : 21
Ninad : 26
Anu Exists in map with value 16
```

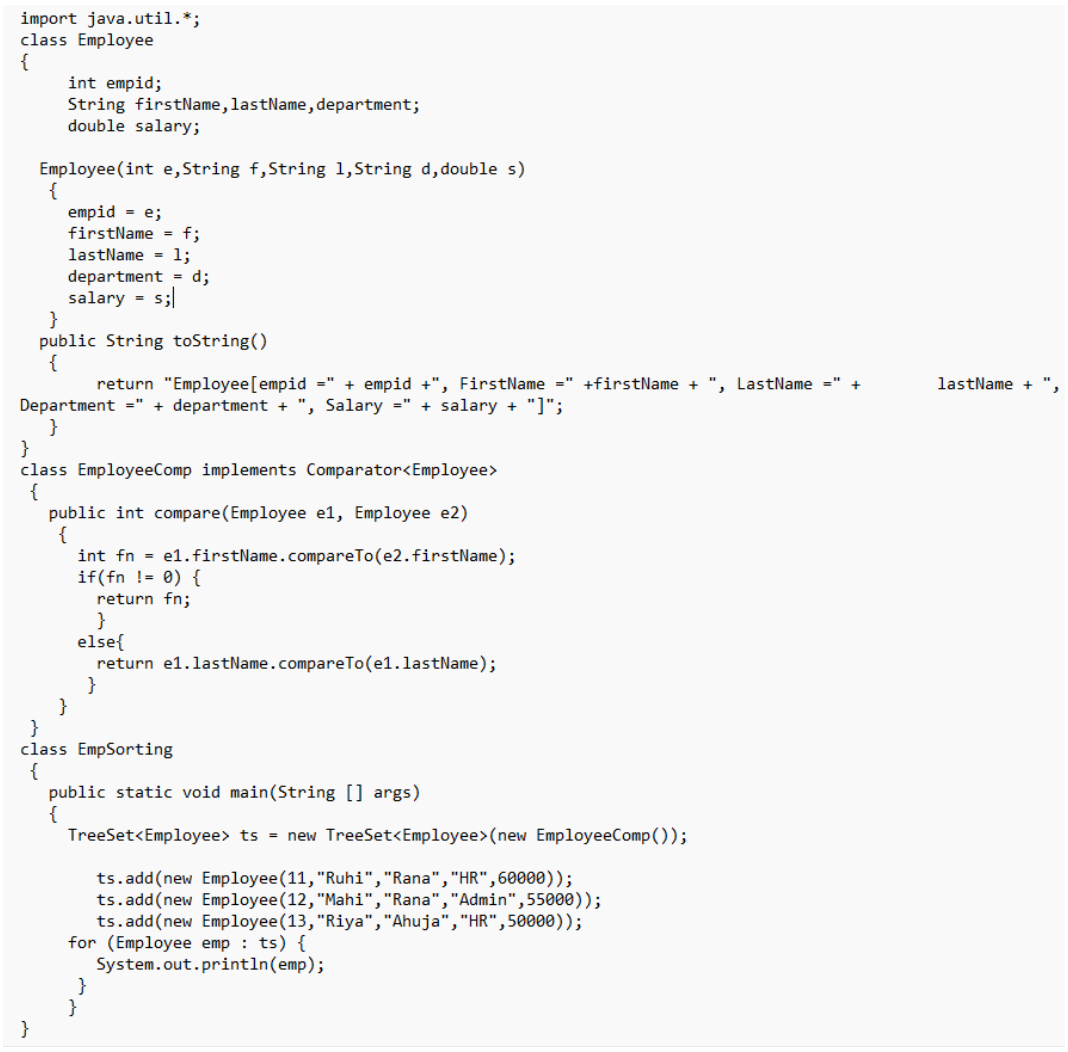
Q2. Create a class with a generic method to find the largest element in an array and its position.

```
import java.util.*;

class Pair<T,U>
{
  T first;
  U second;

  public Pair(T f, U s)
  {
    first = f;
    second = s;
  }
  public T getFirst()
  {
    return first;
  }
  public U getSecond()
  {
    return second;
  }
}
class GenericM<T extends Comparable<T>>
{
    public Pair<T,Integer>findLargestElement(T arr[])
     {
         if (arr.length == 0)
         return null;

         T largest = arr[0];
         int position = 0;

      for (int i = 1; i < arr.length ; i++) {
         if(arr[i].compareTo(largest) > 0 ){
            largest = arr[i];
            position = i;
           }
      }
         System.out.println("Largest element : "+ largest);
         System.out.println("Position :" + position);
         return new Pair<>(largest,position);
  }
}
 public class GenericMain
 {
   public static void main(String [] args)
      {
        GenericM<Integer> g = new GenericM<Integer>();
        Integer [] a = {4,5,6,8,10};
        System.out.println("For Integer Array");
        Pair<Integer,Integer>r=g.findLargestElement(a);
        if(r != null) {
          System.out.println("Largest element :" +r.getFirst());
          System.out.println("Position :" +r.getSecond());
        }
        else
         {
          System.out.println("The array is empty");


        }
      }
```

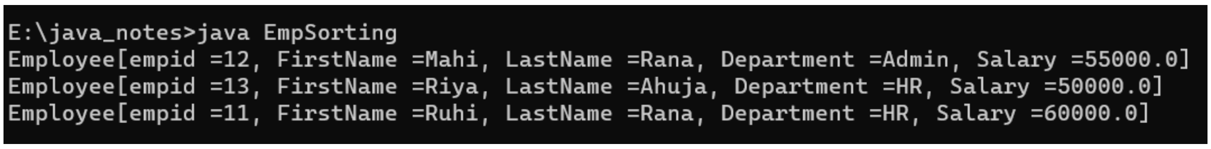


```
E:\java_notes>java GenericMain
For Integer Array
Largest element : 10
Position :4
Largest element :10
Position :4
```

Q3. Create an Employee class with data members empid, first name, last name, dept and salary. Create a Treeset of Employee objects and sort objects using first name. If two employees have the same first name, then sort them by last name using Comparator.



```
import java.util.*;
class Employee
{
     int empid;
     String firstName,lastName,department;
     double salary;

  Employee(int e,String f,String l,String d,double s)
   {
     empid = e;
     firstName = f;
     lastName = l;
     department = d;
     salary = s;
   }
  public String toString()
   {
       return "Employee[empid =" + empid +", FirstName =" +firstName + ", LastName =" +        lastName + ",
Department =" + department + ", Salary =" + salary + "]";
   }
}
class EmployeeComp implements Comparator<Employee>
 {
   public int compare(Employee e1, Employee e2)
    {
      int fn = e1.firstName.compareTo(e2.firstName);
      if(fn != 0) {
        return fn;
        }
      else{
        return e1.lastName.compareTo(e1.lastName);
       }
    }
 }
class EmpSorting
 {
   public static void main(String [] args)
   {
     TreeSet<Employee> ts = new TreeSet<Employee>(new EmployeeComp());

        ts.add(new Employee(11,"Ruhi","Rana","HR",60000));
        ts.add(new Employee(12,"Mahi","Rana","Admin",55000));
        ts.add(new Employee(13,"Riya","Ahuja","HR",50000));
     for (Employee emp : ts) {
        System.out.println(emp);
      }
     }
}
```



```
E:\java_notes>java EmpSorting
Employee[empid =12, FirstName =Mahi, LastName =Rana, Department =Admin, Salary =55000.0]
Employee[empid =13, FirstName =Riya, LastName =Ahuja, Department =HR, Salary =50000.0]
Employee[empid =11, FirstName =Ruhi, LastName =Rana, Department =HR, Salary =60000.0]
```