Collections

Assignment

1) Write Java code to define List . Insert 5 floating point numbers in List, and using an iterator, find the sum of the numbers in List.

```
import java.util.ArrayList;

import java.util.Iterator;

public class ListEg {

   public static void main(String[] args){

       ArrayList<Float> lit = new ArrayList<>();

       lit.add(3.45F);

       lit.add(4.56F);

       lit.add(5.67F);

       lit.add(5.55F);

       lit.add(6.66F);

       float sum = 0;

       System.out.println("list items : ");

       Iterator it = lit.iterator();


       while(it.hasNext()) {

           sum += (float) ( it.next());

       }

       System.out.println(sum);


   }


}
```

```
/home/shreya/.jdks/openjdk-19.0.2/bin/java -javaagent:/snap/intellij-idea-commun
list items :
25.89

Process finished with exit code 0
```

2) Given the following class

Employee class{ Double Age; Double Salary; String Name}

Design the class in such a way that the default sorting should work on firstname and lastname.

Also, Write a program to sort Employee objects based on salary using Comparator.

```
public class Employee implements Comparable<Employee>{

   private String name;

  private double age,salary;

   public Employee(String name,double age,double salary){

       this.name=name;

       this.age=age;

       this.salary=salary;

   }

   public int compareTo(Employee o){

       String[] thisname=this.name.split(" ");

       String[] oname=o.name.split(" ");

       int c=thisname[0].compareTo(oname[0]);

       if(c!=0){

           return c;

       }
```

```
        else{
            return thisname[1].compareTo(oname[1]);
        }
    }
    public double getSalary(){
        return salary;
    }
    public double getAge(){
        return age;
    }
    public String getName(){
        return name;
    }


}
import java.util.*;
public class SalaryComparator implements  Comparator<Employee>{
    public int compare(Employee o, Employee o2){
        return Double.compare(o.getSalary(),o2.getSalary());
    }
}
import java.util.*;
public class Main {
    public static void main(String[] args) {
        List<Employee> e=new ArrayList<>();
        e.add(new Employee("Shruti Jain",23,455000));
```

```
        e.add(new Employee("Charu Sharma",45,650000));

        e.add(new Employee("Kavya Mohan",45,890000));

        e.add(new Employee("Shruti Sharma",54,650000));

        Collections.sort(e);

        System.out.println("Default Sorting by name : ");

        for(Employee em : e){

            System.out.println(em.getName()+ " "+em.getSalary()+"
"+em.getAge());

        }

        Collections.sort(e,new SalaryComparator());

        System.out.println("Sorting Based on Salary : ");

        for(Employee em : e){

            System.out.println(em.getName()+ " "+em.getSalary()+"
"+em.getAge());

        }

    }

}
```

```
Main
/home/shreya/.jdks/openjdk-19.0.2/bin/java -javaagent:/snap/intellij-idea-community/409/lib/idea_rt
Default Sorting by name :
Charu Sharma 650000.0 45.0
Kavya Mohan 890000.0 45.0
Shruti Jain 455000.0 23.0
Shruti Sharma 650000.0 54.0
Sorting Based on Salary :
Shruti Jain 455000.0 23.0
Charu Sharma 650000.0 45.0
Shruti Sharma 650000.0 54.0
Kavya Mohan 890000.0 45.0

Process finished with exit code 0
```

3) Design a Data Structure SpecialStack that supports all the stack operations like push(), pop(), isEmpty(), isFull() and an additional operation getMin() which should return minimum element from the SpecialStack. (Expected complexity O(1))

```
import java.util.*;

public class Specialstack {

  Stack<Integer> st= new Stack<>();

  Integer minval;

  public boolean isEmpty(){

      return st.isEmpty();

  }

  public boolean isFull(){

      return st.size()==Integer.MAX_VALUE;

  }

  public void push(int val){

    if(st.isEmpty()){
```

```
        minval=val;
        st.push(val);
    }
    else{
        if(val<minval){
            st.push((2*val)-minval);
            minval=val;
        }
       else st.push(val);
    }
  }
  public void pop(){
    if(st.isEmpty()){
        return;
    }
    int val=st.peek();
    if(val<minval){
        minval=(2*minval)-val;
    }
    st.pop();
  }
  public int top(){
      int val=st.peek();
      if(val<minval){
          return minval;
      }
```

```
        return val;
    }
    public int getMin(){
        if(st.isEmpty()){
            throw new IllegalStateException("Stack is Empty");
        }
        return minval;
    }
}
```

```
public class Mains {
    public static void main(String[] args){
        Specialstack s=new Specialstack();
        System.out.println("Stack isEmpty Method: "+s.isEmpty());
        System.out.println("Stack isFull Method : "+s.isFull());
        System.out.println("Minimum Values :");
        s.push(5);
        System.out.println(s.getMin());
        s.push(3);
        System.out.println(s.getMin());
        s.push(7);
        s.push(6);
        s.push(1);
        System.out.println(s.getMin());
        s.pop();
        System.out.println(s.getMin());
```

```
        System.out.println("Stack isEmpty Method: "+s.isEmpty());

        System.out.println("Stack isFull Method : "+s.isFull());

    }

}
```

```
Run:  User ×  Mains ×
/home/shreya/.jdks/openjdk-19.0.2/bin/java -javaagent:/snap/intellij-idea-community/409/lib/idea_rt
Stack isEmpty Method: true
Stack isFull Method : false
Minimum Values :
5
3
1
3
Stack isEmpty Method: false
Stack isFull Method : false

Process finished with exit code 0
```

4) Create class Employee with attributes name,age,designation and use instances of these class as keys in a Map and their salary as value

```
import java.util.*;



public class Employee {

   private String name,designation;

   private int age;

   public Employee(String name,String designation,int age){

       this.name=name;

       this.age=age;

       this.designation=designation;
```

```
    }

    public String getDetails(){

        return "Name :"+name+" Designation :"+designation+" Age :"+age;

    }

    @Override

    public boolean equals(Object e){

       if(this.equals(e)){

         return true;

       }

      if(!(e instanceof Employee)){

         return false;

      }

      Employee emp=(Employee) e;

      return
(emp.name.equals(this.name)&&emp.designation.equals(this.designation)&&
Objects.equals(emp.age, this.age));

    }

   @Override

   public int hashCode(){

      return Objects.hash(name,designation,age);

   }

    public static void main(String[] args){

        Map<Employee,Integer> emmap=new HashMap<>();

        Employee a=new Employee("Aakash","Manager",45);

        Employee b=new Employee("Shyam","Assistant",28);

        Employee c=new Employee("Ravina","Receptionist",29);

        Employee d=new Employee("Shruti","Developer",38);
```

```
        emmap.put(a,560000);

        emmap.put(b,780000);

        emmap.put(c,890000);

        emmap.put(d,330000);

        for(Map.Entry<Employee,Integer> entry : emmap.entrySet()){

            System.out.println(entry.getKey().getDetails()+" earns
"+entry.getValue());

        }

    }

}
```

```
Run:  Employee
/home/shreya/.jdks/openjdk-19.0.2/bin/java -javaagent:/snap/intellij-idea-community/409/lib/idea_rt
Name :Shyam Designation :Assistant Age :28 earns 780000
Name :Aakash Designation :Manager Age :45 earns 560000
Name :Ravina Designation :Receptionist Age :29 earns 890000
Name :Shruti Designation :Developer Age :38 earns 330000

Process finished with exit code 0
```