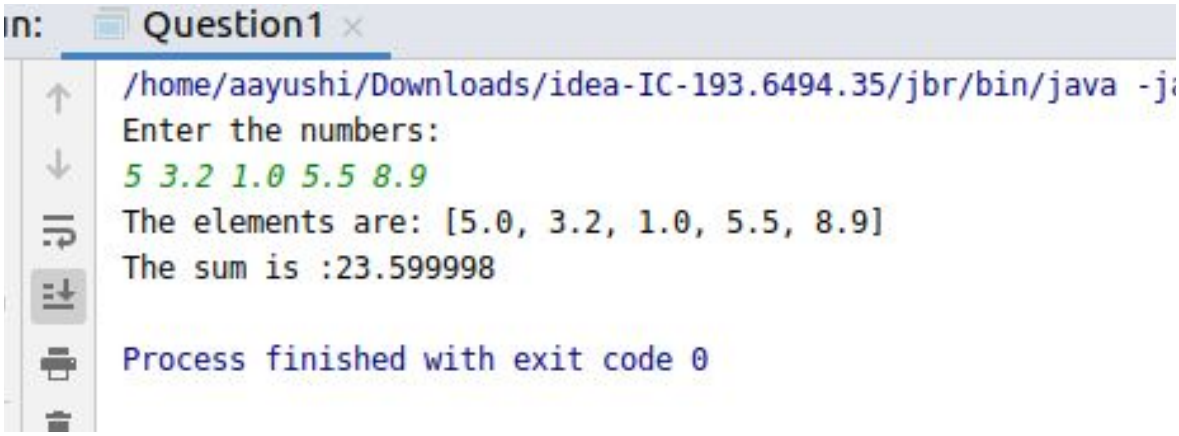
Session : Collections

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.**

CODE

```
package aayushi;
import java.util.ArrayList;
import java.util.List;
import java.util.ListIterator;
import java.util.Scanner;

public class Question1 {
  public static void main(String[] args) {
     float sum=0;
     List<Float> arrlist = new ArrayList<Float>();
     Scanner sc = new Scanner(System.in);

     System.out.println("Enter the numbers: ");
     for (int i=0;i<5;i++)
     {
        float n = sc.nextFloat();
        arrlist.add(n);
     }

     ListIterator<Float> iterator = arrlist.listIterator();
     while (iterator.hasNext())
     {
        sum = sum + iterator.next();
     }
     System.out.println("The elements are: "+arrlist);
     System.out.println("The sum is :"+ sum);
  }
}
```

OUTPUT



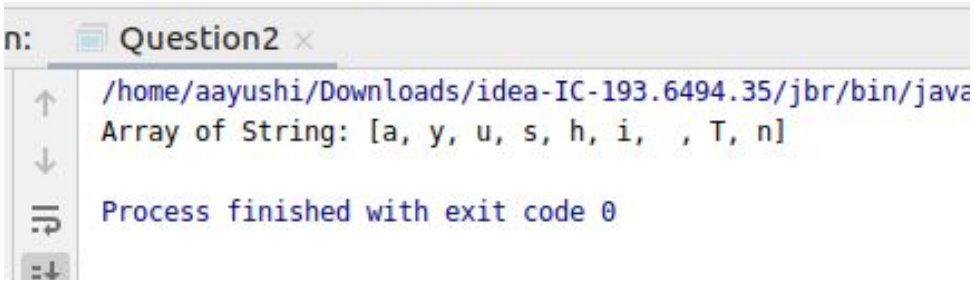
2. **Write a method that takes a string and returns the number of unique characters in the string.**

CODE

```
package aayushi;
import java.util.Arrays;
import java.util.LinkedHashSet;

//Write a method that takes a string and
// returns the number of unique characters in the string.

public class Question2 {
  public static String[] findunique(String str)
  {
     LinkedHashSet<String> set = new LinkedHashSet<String>();
     for(int i=0;i<str.length();i++)
     {
        set.add(String.valueOf(str.charAt(i)));
     }

     String[] arrayOfString = set.toArray(new String[0]);
     return arrayOfString;
  }

  public static void main(String[] args) {
     String str = "aayushi Thani";
     String[] arrayOfString =findunique(str);
     System.out.println("Array of String: " + Arrays.toString(arrayOfString));
```

```
  }
}
```

OUTPUT



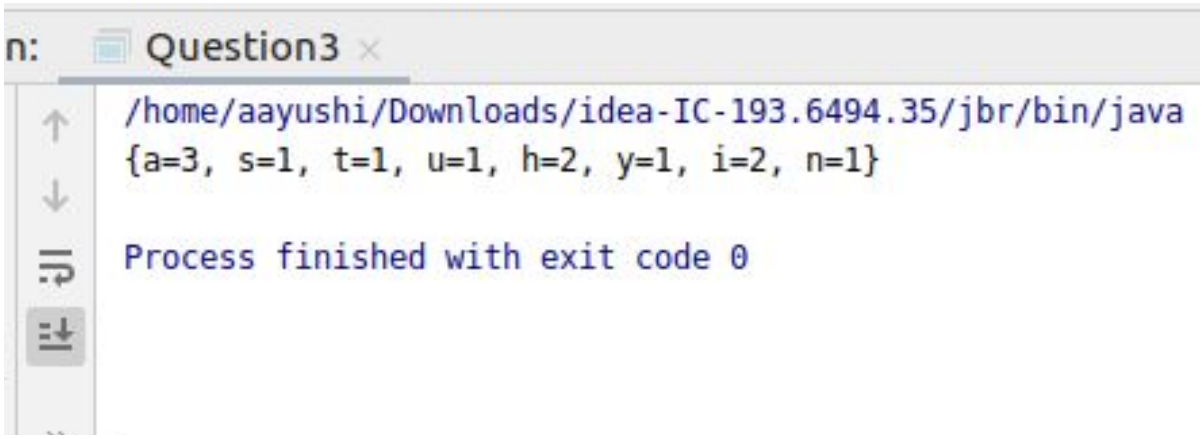
3. **Write a method that takes a string and print the number of occurrence of each character characters in the string.**

CODE

```
package aayushi;
import java.util.HashMap;

//Write a method that takes a string and print the number
// of occurrence of each character characters in the string.
public class Question3 {
  public static void occur(String str)
  {
     HashMap <Character, Integer> map = new HashMap<>();
     for (int i =0 ; i < str.length(); i++)
     {
        if (map.containsKey(str.charAt(i)))
        {
           int count = map.get(str.charAt(i));
           map.put(str.charAt(i), ++count);
        }
        else
        {
           map.put(str.charAt(i),1);
        }
     }
     System.out.println(map);
  }

  public static void main(String[] args) {
     String str = "aayushithani";
     occur(str);
```

```
  }
}
```

OUTPUT



4. **Write a program to sort HashMap by value.**

CODE

```
package aayushi;
import java.util.*;
import java.lang.*;

// Write a program to sort HashMap by value.
public class Question4 {
  public static HashMap<String, Integer> sortByValue(HashMap<String, Integer> hm) {
    // Create a list from elements of HashMap
    List<Map.Entry<String, Integer>> list =
          new LinkedList<Map.Entry<String, Integer>>(hm.entrySet());

    // Sort the list
    Collections.sort(list, new Comparator<Map.Entry<String, Integer>>() {
      public int compare(Map.Entry<String, Integer> o1,
                         Map.Entry<String, Integer> o2) {
        return (o1.getValue()).compareTo(o2.getValue());
      }
    });

    // put data from sorted list to hashmap
    HashMap<String, Integer> temp = new LinkedHashMap<String, Integer>();
    for (Map.Entry<String, Integer> aa : list) {
      temp.put(aa.getKey(), aa.getValue());
```

```
        }
        return temp;
    }


    public static void main(String[] args) {
        HashMap<String, Integer> hm = new HashMap<String, Integer>();

        hm.put("Aayushi", 95);
        hm.put("Komal", 85);
        hm.put("Aakash", 91);
        hm.put("Shobhit", 95);
        hm.put("Aastha", 79);
        hm.put("Pragya", 80);
        Map<String, Integer> hm1 = sortByValue(hm);

        // print the sorted hashmap
        for (Map.Entry<String, Integer> en : hm1.entrySet()) {
            System.out.println("Key = " + en.getKey() +
                    ", Value = " + en.getValue());
        }
    }
}
```

OUTPUT

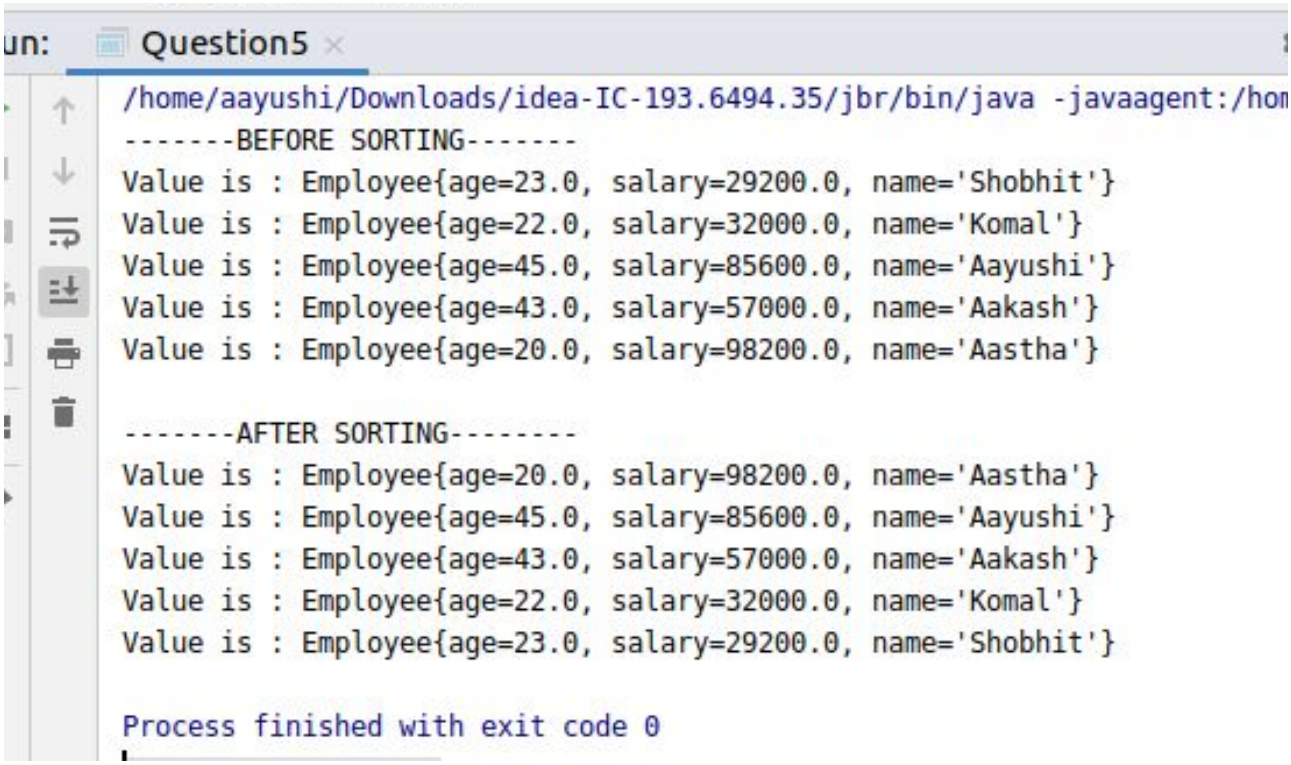
5. **Write a program to sort Employee objects based on highest salary using Comparator. Employee class{ Double Age; Double Salary; String Name}.**

CODE

```
package aayushi;
import java.util.*;

class Employee
{
  private double age;
  private double salary;
  private String name;

  public Employee(double age,double salary,String name)
  {
     this.age=age;
     this.name=name;
     this.salary=salary;
  }


  public double getSalary() {
     return salary;
  }

  @Override
  public String toString() {
     return "Employee{" +
           "age=" + age +
           ", salary=" + salary +
           ", name='" + name + '\'' +
           '}';
  }
}

class SalarySorter implements Comparator<Employee>
{
  @Override
  public int compare(Employee t1, Employee t2) {
     return (int) (t1.getSalary() - t2.getSalary());
  }
```

```
}

public class Question5 {
  public static void main(String[] args) {
     Employee e1 = new Employee(45, 85600, "Aayushi");
     Employee e2 = new Employee(22, 32000, "Komal");
     Employee e3 = new Employee(43, 57000, "Aakash");
     Employee e4 = new Employee(20, 98200, "Aastha");
     Employee e5 = new Employee(23, 29200, "Shobhit");


     List<Employee> employees = new ArrayList<Employee>();
     employees.add(e5);
     employees.add(e2);
     employees.add(e1);
     employees.add(e3);
     employees.add(e4);

     ListIterator<Employee> itr = employees.listIterator();
     System.out.println("-------BEFORE SORTING-------");

     while (itr.hasNext())
     {
        System.out.println("Value is : " + itr.next());
     }

     System.out.println();
     Collections.sort(employees,Collections.reverseOrder(new SalarySorter()));
     ListIterator<Employee> iterator = employees.listIterator();

     System.out.println("-------AFTER SORTING--------");
     while (iterator.hasNext())
     {
        System.out.println("Value is : " + iterator.next());
     }
  }
}
```

OUTPUT



6. **Write a program to sort the Student objects based on Score , if the score are same then sort on First Name . Class Student{ String Name; Double Score; Double Age.**

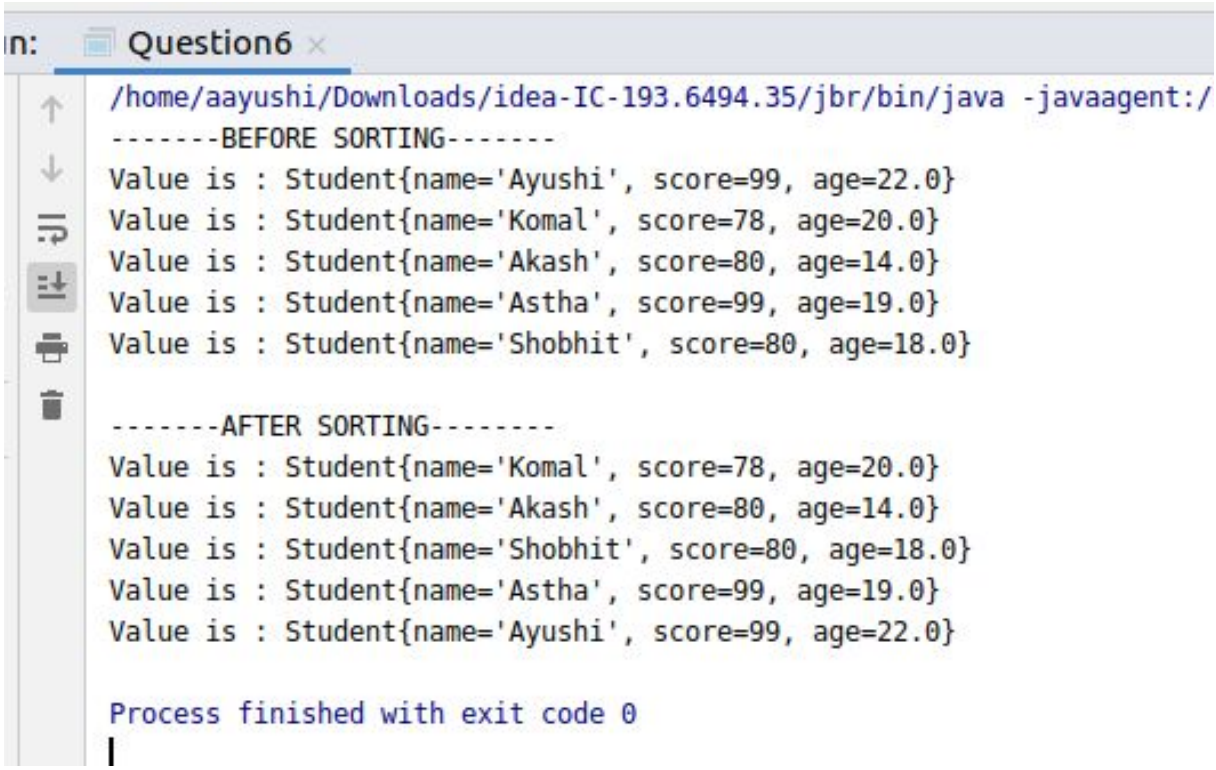
CODE

```
package aayushi;
import java.util.*;

class Student
{
  private String name;
  private int score;
  private double age;

  public Student(String name,int score,double age)
  {
     this.name=name;
     this.score=score;
     this.age=age;
  }
```

```
  public String getName() {
     return name;
  }

  public double getScore() {
     return score;
  }

  @Override
  public String toString() {
     return "Student{" +
           "name='" + name + '\'' +
           ", score=" + score +
           ", age=" + age +
           '}';
  }
}

class StudentSorted implements Comparator<Student>{
  public int compare(Student t1, Student t2) {

     // for comparison
     int ScoreCompare = Double.compare(t1.getScore(), t2.getScore());
     int NameCompare = t1.getName().compareTo(t2.getName());

     // 2-level comparison using if-else block
     if (ScoreCompare == 0) {
        return ((NameCompare == 0) ? ScoreCompare : NameCompare);
     } else {
        return ScoreCompare;
     }
  }
}

public class Question6 {
  public static void main(String[] args) {
     Student s1 = new Student("Ayushi",99,22);
     Student s2 = new Student("Komal",78,20);
     Student s3 = new Student("Akash",80,14);
     Student s4 = new Student("Astha",99,19);
     Student s5 = new Student("Shobhit",80,18);
```

```
        List<Student> student = new ArrayList<Student>();
        student.add(s1);
        student.add(s2);
        student.add(s3);
        student.add(s4);
        student.add(s5);

        ListIterator<Student> itr = student.listIterator();
        System.out.println("-------BEFORE SORTING-------");

        while (itr.hasNext())
        {
           System.out.println("Value is : " + itr.next());
        }

        System.out.println();

        Collections.sort(student,new StudentSorted());

        ListIterator<Student> iterator = student.listIterator();

        System.out.println("-------AFTER SORTING--------");
        while (iterator.hasNext())
        {
           System.out.println("Value is : " + iterator.next());
        }
    }
}
```

OUTPUT



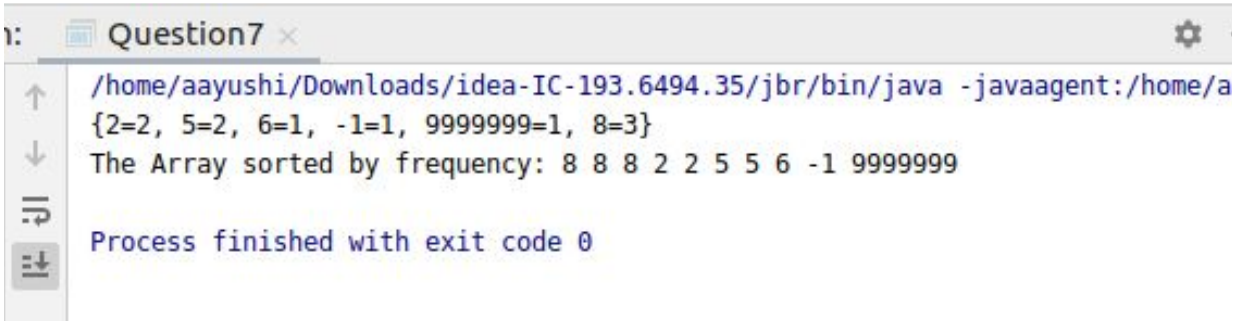
7. **Print the elements of an array in the decreasing frequency if 2 numbers have same frequency then print the one which came first.**

CODE

```
package aayushi;
import java.util.LinkedHashMap;
import java.util.Map;

public class Question7 {
  public static void main(String[] args) {
     int arr[] = { 2, 5, 2, 6, -1, 9999999, 5, 8, 8, 8 };
     Map<Integer, Integer> countMap = new LinkedHashMap<>();
     StringBuffer result = new StringBuffer();

     for (int i = 0; i < arr.length; i++)
     {
        if (countMap.containsKey(arr[i]))
        {
```

```
            countMap.put(arr[i], countMap.get(arr[i]) + 1);
        }
        else
        {
            countMap.put(arr[i], 1);
        }
    }
    System.out.println(countMap);

    countMap.entrySet().stream().sorted(Map.Entry.<Integer, Integer>
comparingByValue().reversed())
            .forEach(e -> {
                int key = e.getKey();
                int val = e.getValue();
                for (int i = 0; i < val; i++) {
                    result.append(key + " ");
                }
            });

    System.out.println("The Array sorted by frequency: "+result);
  }
}
```

OUTPUT



8. **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))**

CODE

```
package aayushi;
import java.util.Stack;
```

```
class myStack {

   Stack<Integer> stack;
   Integer minElement;

   public myStack() {
      stack = new Stack<Integer>();
   }

   public void peek() {

      if (stack.isEmpty()) {
         System.out.println("Stack is Empty ....");
      }
      Integer t = stack.peek();
      if (t < minElement) {
         System.out.println("Top most element is :" + minElement);
      } else
         System.out.println("Top most element is :" + t);
   }

   public void push(Integer i) {

      if (stack.isEmpty()) {
         minElement = i;
         stack.push(i);
         System.out.println("Element inserted :" + i);

         return;
      }

      if (i < minElement) {
         stack.push(2 * i - minElement);
         minElement = i;

      } else
         stack.push(i);

      System.out.println("Element inserted :" + i);
   }
```

```
  public void pop() {

    if (stack.isEmpty()) {
      System.out.println("Stack is empty, cannot pop an element");
      return;
    }
    Integer t = stack.pop();

    if (t < minElement) {

      System.out.println("element removed :" + minElement);

      minElement = 2 * minElement - t;

    } else
      System.out.println("element removed :" + t);
  }

  public void getMinElement() {

    if (stack.isEmpty()) {
      System.out.println("stack is empty..");
    } else
      System.out.println("min element is :" + minElement);
  }
}


public class Question8 {
  public static void main(String[] args) {
    myStack stack = new myStack();
    stack.getMinElement();
    stack.pop();
    stack.push(3);
    stack.push(4);
    stack.push(2);
    stack.push(5);
    stack.push(1);
    stack.push(8);
    stack.getMinElement();
    stack.peek();
    stack.pop();
    stack.getMinElement();
```

```
        stack.pop();
        stack.getMinElement();
    }
}
```

OUTPUT