

S.No: 1

Exp. Name: **Write a Java program to Display Default values of all Primitive data types**

Date: 2023-08-29

Aim:

Write a **java** program to display the default values of all primitive data types.

Write a class `PrimitiveTypes` with `main(String[] args)` method.

Write code to produce the below output:

```
byte default value = 0
short default value = 0
int default value = 0
long default value = 0
boolean default value = false
double default value = 0.0
float default value = 0.0
```

Note: Please don't change the package name.

Source Code:

q10815/PrimitiveTypes.java

```
package q10815;
import java.io.*;
public class PrimitiveTypes{
public static void main(String args[])
{
    System.out.println("byte default value = 0");
    System.out.println("short default value = 0");
    System.out.println("int default value = 0");
    System.out.println("long default value = 0");
    System.out.println("boolean default value = false");
    System.out.println("double default value = 0.0");
    System.out.println("float default value = 0.0");
}
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
byte default value = 0
short default value = 0
int default value = 0
long default value = 0
boolean default value = false
```

```
double default value = 0.0  
float default value = 0.0
```

S.No: 2

Exp. Name: **Write a Java code to calculate the Roots of a Quadratic equation**

Date: 2023-09-04

Aim:

Write code to calculate **roots** of a **quadratic equation**.

Write a class `QuadraticRoots` with `main` method. The method receives three arguments, write code to parse them into `double` type.

For example:

if the values 2, 5, 3 are passed as arguments, then the output should be **First root is : -1.0** **Second root is : -1.5**

If the values 3, 2, 1 are passed then the output should be **Roots are imaginary**

Similarly, if the values 2, 4, 2 are passed then the output should be **Roots are equal and value is : -1.0**

Note: Make sure to use the `print()` and not the `println()` method.

Note: Please don't change the package name.

Source Code:

q10851/QuadraticRoots.java

```

package q10851;
class QuadraticRoots{
    double a,b,c;
    void getData(String c1,String c2,String c3)
    {
        a=Double.valueOf(c1);
        b=Double.valueOf(c2);
        c=Double.valueOf(c3);
    }
    void roots ()
    {
        double d;
        if(a==0)
        {
            double root;
            root=-c/b;
            System.out.println("linear equation "+root);
        }
        else
        {
            d=(b*b)-(4*a*c);
            if(d==0)
            {
                double root=-b/(2*a);
                System.out.println("Roots are equal and value is : "+root);
            }
            else if(d>0)
            {
                double r1,r2;
                r1=(-b+Math.sqrt(d))/(2*a);
                r2=(-b-Math.sqrt(d))/(2*a);
                System.out.println("First root is : "+r1+" Second root is : "+r2);
            }
            else
                System.out.println("Roots are imaginary");
        }
    }
    public static void main(String a[])
    {
        QuadraticRoots r=new QuadraticRoots();
        r.getData(a[0],a[1],a[2]);
        r.roots();
    }
}

```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

First root is : -0.6047152924789525 Second root is : -1.3952847075210475

Test Case - 2

User Output

Roots are equal and value is : -1.0

Test Case - 3

User Output

Roots are imaginary

S.No: 3

Exp. Name: **Write a Java program to print the speeds of qualifying bikers in a Race**

Date: 2023-09-07

Aim:

Five bikers compete in a race such that they drive at a constant speed which may or may not be the same as the other.

To qualify the race, the speed of a racer must be more than or equal to the average speed of all the 5 racers.

Take as input the speed of each racer and print back the speeds of qualifying racers.

Write a class `Race` with a method `main(String[] args)`. The main method receives five arguments. You can write code to parse them into `double` data type.

For example, if the values `54.55, 53.57, 54, 56.25, 57.30` are passed as arguments to the main() method, then the output should be `The speed of the racers >= average speed 55.134 : 56.25 57.3`.

Note: Make sure to use the `print()` method and not the `println()` method.

Source Code:

Race.java

```
class Race
{
    public static void main(String a[])
    {
        double[]arr=new double[5];
        double average,sum=0;
        for(int i=0;i<5;i++)
        {
            arr[i]=Double.valueOf(a[i]);
        }
        for(int i=0;i<5;i++)
        sum+=arr[i];
        average=sum/5;
        System.out.print("The speed of the racers >= average speed "+average+": ");
        for(int i=0;i<5;i++)
        {
            if(average<=arr[i])
            System.out.print(", "+arr[i]);
        }
    }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

The speed of the racers >= average speed 54.855999999999995: ,81.6,58.19,79.42

Test Case - 2

User Output

The speed of the racers >= average speed 78.0032: ,96.21,87.26,105.63

S.No: 4

Exp. Name: **Search for an element in a given list of elements using Binary Search.**

Date: 2023-09-19

Aim:

Write a program to search for an element in a given list of elements using **Binary Search** mechanism.

Source Code:

q36414/BinarySearch.java

```
package q36414;
import java.util.*;
class BinarySearch
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner (System.in);
        System.out.print("Enter the number of elements: ");
        int n=sc.nextInt();
        int a[] = new int[n];
        System.out.println("Enter the sorted elements:");
        for(int i=0;i<n;i++)
        {
            a[i]=sc.nextInt();
        }
        System.out.print("Enter the element to search for: ");
        int key=sc.nextInt();
        int low=0,high=n-1,mid;
        while(low<=high)
        {
            mid=(low+high)/2;
            if(key==a[mid])
            {
                System.out.println("Element "+key+" found at index "+mid);
                break;
            }
            else if(key>a[mid])
            {
                low=mid+1;
            }
            else
            {
                high=mid-1;
            }
            if(low>high)
            {
                System.out.println("Element "+key+" not found in the list.");
            }
        }
    }
}
```

Test Case - 1

User Output

Enter the number of elements:

5

Enter the sorted elements:

10 20 30 40 50

Enter the element to search for:

30

Element 30 found at index 2

Test Case - 2

User Output

Enter the number of elements:

8

Enter the sorted elements:

2 4 6 8 10 12 14 16

Enter the element to search for:

9

Element 9 not found in the list.

Aim:

Write a java program to sort the given list of elements using **Bubble Sort**.

Source Code:

q36415/BubbleSort.java

```
package q36415;
import java.util.*;
class BubbleSort
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        int temp,i,j;
        System.out.print("Enter no of elements: ");
        int n =sc.nextInt();
        System.out.println("Enter the elements: ");
        int a[] = new int[n];
        for(i=0;i<n;i++)
        {
            a[i] = sc.nextInt();
        }
        System.out.println("Sorted elements:");
        for(i=0;i<n;i++)
        {
            for(j=0;j<(n-i-1);j++)
            {
                if(a[j]>a[j+1])
                {
                    temp=a[j];
                    a[j]=a[j+1];
                    a[j+1]=temp;
                }
            }
            for(i=0;i<n;i++)
            {
                System.out.print(a[i]+" ");
            }
        }
    }
}
```

5 -1 25

Sorted elements:

-1 5 25

Test Case - 2

User Output

Enter no of elements:

6

Enter the elements:

15 54 87 69 32 151

Sorted elements:

15 32 54 69 87 151

S.No: 6

Exp. Name: ***Sort a list using Merge Sort Technique.***

Date: 2023-09-24

Aim:

Write a java program to sort the given list of elements using **Merge Sort**.

Source Code:

q36416/MergeSort.java

```

package q36416;
import java.util.*;
class MergeSort
{
    public static void main(String args[])
    {
        int n,i;
        Scanner s =new Scanner(System.in);
        System.out.print("Enter no of elements: ");
        n = s.nextInt();
        int a[] = new int[n];
        System.out.println("Enter the elements:");
        for(i=0;i<n;i++)
            a[i]=s.nextInt();
        Merge.SplitAndMerge(a,0,n-1,n);
        System.out.print("Sorted array: \n");
        for(i=0;i<n;i++)
            System.out.print(a[i]+" ");
    }
}
class MergeSortDemo
{
    public static void MergeSorting(int a[],int low,int mid,int high,int n)
    {
        int i,j,k;
        int b[]={};
        i=low;
        j=mid+1;
        k=low;
        while(i<=mid&&j<=high)
        {
            if(a[i]<=a[j])
            {
                b[k] = a[i];
                i++;
            }
            else
            {
                b[k] = a[j];
                j++;
            }
            k++;
        }
        if(i<=mid)
        {
            while(i<=mid)
            {
                b[k] =a[i];
                i++;
                k++;
            }
        }
        else
        {
            while(j<=high)
        }
    }
}

```

```

        j++;
        k++;
    }
}
for(k=low;k<=high;k++)
a[k] = b[k];
}
}
class Merge
{
    public static void SplitAndMerge(int a[],int low,int high,int n)
    {
        int mid;
        if(low<high)
        {
            mid=low+(high-low)/2;
            Merge.SplitAndMerge(a,low,mid,n);
            Merge.SplitAndMerge(a,mid+1,high,n);
            MergeSortDemo.MergeSorting(a,low,mid,high,n);
        }
    }
}

```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

Enter no of elements:

3

Enter the elements:

100 50 75

Sorted array:

50 75 100

Test Case - 2

User Output

Enter no of elements:

4

Enter the elements:

1 3 5 2

Sorted array:

1 2 3 5

S.No: 7

Exp. Name: **Write a Java program to Delete and Remove characters using StringBuffer class**

Date: 2023-09-23

Aim:

Write a class `JavaStringBufferDelete` with a **main** method to delete characters from a string using `StringBuffer` class.

Source Code:

JavaStringBufferDelete.java

```
import java.util.*;
import java.lang.StringBuffer;
public class JavaStringBufferDelete
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.print("string: ");
        String inputString = sc.nextLine();
        System.out.print("start index: ");
        int startIndex = sc.nextInt();
        System.out.print("end index: ");
        int endIndex = sc.nextInt();
        StringBuffer ste = new StringBuffer(inputString);
        ste.delete(startIndex,endIndex);
        System.out.print("Result after delete: "+ste.toString()+"\n");
    }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
string:
Hello world
start index:
0
end index:
6
Result after delete: world
```

S.No: 8

Exp. Name: **Write a Java program to implement a Class mechanism**

Date: 2023-09-23

Aim:

Write a Java program with a class name **Employee** which contains the data members **name** (String), **age** (int), **designation** (String), **salary** (double) and the methods **setData()**, **displayData()**.

The member function **setData()** is used to initialize the data members and **displayData()** is used to display the given employee data.

Write the **main()** method with in the class which will receive four arguments as **name**, **age**, **designation** and **salary**.

Create an object to the class **Employee** within the main(), call **setData()** with arguments and finally call the method **displayData()** to print the output.

If the input is given as command line arguments to the **main()** as "Saraswathi", "27", "Teacher", "37250" then the program should print the output as:

Name : Saraswathi
Age : 27
Designation : Teacher
Salary : 37250.0

Note: Please don't change the package name.

Source Code:

q11115/Employee.java

```

package q11115;
class Employee
{
    int age;
    double salary;
    String name,job;
    void setData(String a[])
    {
        name=a[0];
        age = Integer.parseInt(a[1]);
        job=a[2];
        salary = Double.parseDouble(a[3]);
    }
    void displayData()
    {
        System.out.println("Name : "+name+"\n"+ "Age : "+age);
        System.out.println("Designation : "+job+"\n"+ "Salary : "+salary);
    }
    public static void main(String args[])
    {
        Employee a = new Employee();
        a.setData(args);
        a.displayData();
    }
}

```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

Name : Ram
Age : 25
Designation : Team member
Salary : 25000.0

Test Case - 2

User Output

Name : Ravi
Age : 36
Designation : TeamLead
Salary : 35000.0

S.No: 9

Exp. Name: **Write a Java program to implement a Constructor**

Date: 2023-09-23

Aim:

Write a Java program with a class name **Staff**, contains the data members **id** (int), **name** (String) which are initialized with a **parameterized constructor** and the method **show()**.

The member function **show()** is used to display the given staff data.

Write the **main()** method with in the class which will receive two arguments as **id** and **name**.

Create an object to the class **Staff** with arguments **id** and **name** within the **main()**, and finally call the method **show()** to print the output.

If the input is given as command line arguments to the **main()** as "18", "Gayatri" then the program should print the output as:

```
Id : 18  
Name : Gayatri
```

Note: Please don't change the package name.

Source Code:

q11116/Staff.java

```
package q11116;  
class Staff  
{  
    int id;  
    String name;  
    Staff(String a[])  
    {  
        id = Integer.parseInt(a[0]);  
        name = a[1];  
    }  
    void show()  
    {  
        System.out.println("Id : "+id);  
        System.out.println("Name : "+name);  
    }  
    public static void main(String args[])  
    {  
        Staff b = new Staff(args);  
        b.show();  
    }  
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

Id : 18

Name : Gayatri

Test Case - 2

User Output

Id : 45

Name : Akbar

S.No: 10

Exp. Name: **Write a Java program to implement
Constructor overloading**

Date: 2023-09-23

Aim:

Write a class `Box` which contains the data members **width**, **height** and **depth** all of type **double**.

Write the implementation for the below **3 overloaded constructors** in the class `Box` :

- **Box()** - default constructor which initializes all the members with **-1**
- **Box(length)** - parameterized constructor with one argument and initialize all the members with the value in **length**

the members with the corresponding arguments

- **Box(width, height, depth)** - parameterized constructor with three arguments and initialize

Write a method `public double volume()` in the class `Box` to find out the **volume** of the given box.

Write the **main** method within the `Box` class and assume that it will receive either **zero** arguments, or **one** argument or **three** arguments.

For example, if the **main()** method is passed **zero** arguments then the program should print the output as:

Volume of Box() is : -1.0

Similarly, if the **main()** method is passed **one** argument : **2.34**, then the program should print the output as:

Volume of Box(2.34) is : 12.812903999999998

then the program should print the output as: Likewise, if the **main()** method is passed **three** arguments : **2.34, 3.45, 1.59**, then the program should print the output as:

Volume of Box(2.34, 3.45, 1.59) is : 12.836070000000001

Note: Please don't change the package name.

Source Code:

q11267/Box.java

```

package q11267;
class Box
{
    double width,height,depth;
    double volume()
    {
        return width*height*depth;
    }
    Box()
    {
        width=-1;
        height=-1;
        depth=-1;
        System.out.println("Volume of Box() is : "+volume()+"\n");
    }
    Box(String len)
    {
        width=height=depth=Double.parseDouble(len);
        System.out.println("Volume of Box("+width+") is : "+volume());
    }
    Box(String w,String h,String d)
    {
        width=Double.parseDouble(w);
        height=Double.parseDouble(h);
        depth=Double.parseDouble(d);
        System.out.println("Volume of Box("+width+", "+height+", "+depth") is :
"+volume());
    }
    public static void main(String a[])
    {
        int m=a.length;
        Box b;
        if(m==0)
            b=new Box();
        else if(m==1)
            b=new Box(a[0]);
        else if(m==3)
            b=new Box(a[0],a[1],a[2]);
    }
}

```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

Volume of Box() is : -1.0

Test Case - 2

User Output

Volume of Box(3.0) is : 27.0

Test Case - 3

User Output

Volume of Box(2.3, 3.5, 6.5) is : 52.32499999999996

S.No: 11

Exp. Name: **Write a Java program to implement Method overloading**

Date: 2023-09-25

Aim:

Write a Java program with a class name `Addition` with the methods `add(int, int)`, `add(int, float)`, `add(float, float)` and `add(float, double, double)` to add values of different argument types.

Write the `main(String[])` method within the class and assume that it will always receive a total of **6** command line arguments at least, such that the first **2** are **int**, next **2** are **float** and the last **2** are of type **double**.

If the `main()` is provided with arguments : **1, 2, 1.5f, 2.5f, 1.0, 2.0** then the program should print the output as:

```
Sum of 1 and 2 : 3
Sum of 1.5 and 2.5 : 4.0
Sum of 2 and 2.5 : 4.5
Sum of 1.5, 1.0 and 2.0 : 4.5
```

Note: Please don't change the package name.

Source Code:

```
q11266/Addition.java
```

```

package q11266;
class Addition
{
    void add(int a,int b)
    {
        System.out.println("Sum of " +a+ " and " +b+ " : "+(a+b));
    }
    void add(float c,float d)
    {
        System.out.println("Sum of " +c+ " and " +d+ " : "+(c+d));
    }
    void add(int b,float d)
    {
        System.out.println("Sum of "+b+ " and " +d+ " : "+(b+d));
    }
    void add(float c,double e,double f)
    {
        System.out.println("Sum of " +c+ ", "+e+ " and " +f+ " : "+(c+e+f));
    }
    public static void main(String args[])
    {
        Addition g=new Addition();
        int a,b;
        float c,d;
        double e,f;
        a=Integer.parseInt(args[0]);
        b=Integer.parseInt(args[1]);
        c=Float.parseFloat(args[2]);
        d=Float.parseFloat(args[3]);
        e=Double.parseDouble(args[4]);
        f=Double.parseDouble(args[5]);
        g.add(a,b);
        g.add(c,d);
        g.add(b,d);
        g.add(c,e,f);
    }
}

```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Sum of 2 and 1 : 3
Sum of 5.0 and 3.6 : 8.6
Sum of 1 and 3.6 : 4.6
Sum of 5.0, 9.2 and 5.26 : 19.46

S.No: 12

Exp. Name: **Write a Java program to implement Single Inheritance**

Date: 2023-09-25

Aim:

Write a Java program to illustrate the **single inheritance** concept.

Create a class **Marks**

- contains the data members **id** of **int** data type, **javaMarks**, **cMarks** and **cppMarks** of **float** data type
- write a method **setMarks()** to initialize the data members
- write a method **displayMarks()** which will display the given data

Create another class **Result** which is derived from the class **Marks**

- contains the data members **total** and **avg** of **float** data type
- write a method **compute()** to find total and average of the given marks
- write a method **showResult()** which will display the total and avg marks

Write a class **SingleInheritanceDemo** with **main()** method it receives four arguments as **id**, **javaMarks**, **cMarks** and **cppMarks**.

Create object only to the class **Result** to access the methods.

If the input is given as command line arguments to the **main()** as "101", "45.50", "67.75", "72.25" then the program should print the output as:

```
Id : 101
Java marks : 45.5
C marks : 67.75
Cpp marks : 72.25
Total : 185.5
Avg : 61.833332
```

Note: While computing the total marks, add the marks in the following order only **javaMarks**, **cMarks** and **cppMarks**

Source Code:

```
q11263/SingleInheritanceDemo.java
```

```

package q11263;
class Marks
{
    int id;
    float jm,cm,cppm;
    void setMarks(String args[])
    {
        id=Integer.valueOf(args[0]);
        jm=Float.valueOf(args[1]);
        cm=Float.valueOf(args[2]);
        cppm=Float.valueOf(args[3]);
    }
    void displayMarks()
    {
        System.out.println("Id : "+id);
        System.out.println("Java marks : "+jm);
        System.out.println("C marks : "+cm);
        System.out.println("Cpp marks : "+cppm);
    }
}
class Result extends Marks
{
    float t,a;
    void compute(String args[])
    {
        super.setMarks(args);
        t=jm+cm+cppm;
        a=t/3;
    }
    void showResult()
    {
        super.displayMarks();
        System.out.println("Total : "+t);
        System.out.println("Avg : "+a);
    }
}
class SingleInheritanceDemo
{
    public static void main(String args[])
    {
        Result r = new Result();
        r.compute(args);
        r.showResult();
    }
}

```

Execution Results - All test cases have succeeded!

Test Case - 1	
User Output	
Id : 102	
Java marks : 35.6	
C marks : 45.0	

Cpp marks : 65.5

Total : 146.1

Avg : 48.7

Test Case - 2

User Output

Id : 101

Java marks : 45.5

C marks : 67.75

Cpp marks : 72.25

Total : 185.5

Avg : 61.833332

Test Case - 3

User Output

Id : 103

Java marks : 50.5

C marks : 46.8

Cpp marks : 52.65

Total : 149.95001

Avg : 49.983337