S.No: 1	Exp. Name: ***Primitive data types***	Date: 2024-09-01

Aim:

Write a Java program that demonstrates the use of all primitive data types. Your program should prompt the user to input values for each primitive data type (byte, short, int, long, float, double, char, boolean) and then print out those values.

Source Code:

PrimitiveDataTypes.java

```
import java.io.*;
import java.util.Scanner;
class PrimitiveDataTypes
{
static Byte a;
static short b;
static int c;
static long d;
static float e;
static double f;
static char g;
static boolean h;
public static void main (String args[])
        {

Scanner obj=new Scanner(System.in);
System.out.print("Byte value: ");
a=obj.nextByte();
System.out.print("short value: ");
b=obj.nextShort();
System.out.print("int value: ");
c=obj.nextInt();
System.out.print("long value: ");
d=obj.nextLong();
System.out.print("float value: ");
e=obj.nextFloat();
System.out.print("double value: ");
f=obj.nextDouble();
System.out.print("char value: ");
g=obj.next().charAt(0);
System.out.print("boolean value (true/false): ");
h=obj.nextBoolean();
System.out.println("byteVar: "+a);
System.out.println("shortVar: "+b);
System.out.println("intVar: "+c);
System.out.println("longVar: "+d);
System.out.println("floatVar: "+e);
System.out.println("doubleVar: "+f);
System.out.println("charVar: "+g);
System.out.println("booleanVar: "+h);
        }
}
```


ID: 24K65A4702
2023-2027-CIC
Sasi Institute of Technology and Engineering (Autonomous)

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Byte value:
127
short value:
32767
int value:
2147483647
long value:
9223372036854775807
float value:
123.456
double value:
9876.54321
char value:
Z
boolean value (true/false):
true
byteVar: 127
shortVar: 32767
intVar: 2147483647
longVar: 9223372036854775807
floatVar: 123.456
doubleVar: 9876.54321
charVar: Z
booleanVar: true

ID: 24K65A4702

S.No: 2	Exp. Name: ***Roots of the quadratic equation***	Date: 2024-09-04

Aim:

Write a Java Program to find **Roots** of a Quadratic Equation.

Refer to the displayed sample test cases to strictly match the input and output layout.

Source Code:

q27331/QuadraticEquation.java

```
package q27331;

import java.util.Scanner;
public class QuadraticEquation {
        public static void main(String[] args){
                Scanner scanner = new Scanner(System.in);
                System.out.print("Coefficient a: ");
                double a = scanner.nextDouble();
                System.out.print("Coefficient b: ");
                double b = scanner.nextDouble();
                System.out.print("Coefficient c: ");
                double c = scanner.nextDouble();
                double determinant = b * b - 4 * a * c;
                if(determinant>0){
                        double root1 = (-b + Math.sqrt(determinant)) / (2 * a);
                        double root2 = (-b - Math.sqrt(determinant)) / (2 * a);
                        System.out.println("The roots are real and distinct");
                        System.out.println("Root1: " + root1 + " Root2: "+root2);
                }else if(determinant==0){
                        double root = -b / (2 * a);
                        System.out.println("The roots are real and equal");
                        System.out.println("Root: " + root);
                }else{
                        double realPart=-b/(2*a);
                        double imaginaryPart=Math.sqrt(-determinant)/(2*a);
                        System.out.println("The roots are imaginary");
                }
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Coefficient a:
1
Coefficient b:
6
Coefficient c:
9
The roots are real and equal

Root: -3.0

Test Case - 2
User Output
Coefficient a:
1
Coefficient b:
5
Coefficient c:
8
The roots are imaginary

Test Case - 3
User Output
Coefficient a:
2
Coefficient b:
6
Coefficient c:
1
The roots are real and distinct
Root1: -0.17712434446770464 Root2: -2.8228756555322954

Test Case - 4
User Output
Coefficient a:
2
Coefficient b:
6
Coefficient c:
4
The roots are real and distinct
Root1: -1.0 Root2: -2.0

S.No: 3	Exp. Name: ***Binary search Technique***	Date: 2024-09-04

Aim:

Write a program to check whether the given element is present or not in the array of elements using the binary search Technique

Input format:

- The first line of input contains an integer N representing the no. of elements of the array
- The second line input contains the array of N integers separated by space
- The last line of input contains the key element to be searched

Output format:

- If the element is found, print the index.
- If the element is not found, print **Not found**.

Sample Test Case:

Input:

7
1 2 3 4 3 5 6
3

Output:

2

Source Code:

```
q17128/CTJ17128.java
```

```
package q17128;
import java.io.*;
import java.util.Scanner;
class CTJ17128
	{
		public static void main(String args[])
		{
			int i,pos=0,mid,key,flag=0,l,h;
			int a[]=new int[30];
			Scanner obj=new Scanner(System.in);
			System.out.print("");
			int n=obj.nextInt();
			System.out.print("");
			for(i=0;i<n;i++)
				a[i]=obj.nextInt();
			System.out.print("");
			key=obj.nextInt();
			l=0;
			h=n-1;
			while(l<=h)
				{
					mid=(l+h)/2;
					if(key==a[mid])
					{
						flag=1;
						pos=mid;
						break;
					}
					else if(key>a[mid])
						l=mid+1;
					else
						h=mid-1;
				}
			if(flag==1)
				System.out.print(pos);
			else
				System.out.print("Not found");
		}
	}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
7
1 2 3 4 3 5 6
3
2

Test Case - 2

User Output
10
1 2 3 4 5 6 7 8 9 19
20
Not found

S.No: 4	Exp. Name: ***Sort an array using Bubble Sort Technique***	**Date: 2024-09-09**

Aim:

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

Input Format:

- The first line contains an integer n representing the no of elements of the array
- The second line contains n integers separated with space, representing the elements of the array

Output Format:

- The elements of the array after sorting

Source Code:

```
q32083/BubbleSort.java
```


ID: 24K65A4702
2023-2027-CIC
Sasi Institute of Technology and Engineering (Autonomous)

```
package q32083;
import java.util.Scanner;

public class BubbleSort {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        int n = scanner.nextInt();

        int[] arr = new int[n];
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextInt();
        }

        bubbleSort(arr);

        for (int num : arr) {
            System.out.print(num + " ");
        }

        scanner.close();
    }

    public static void bubbleSort(int[] arr) {
int n = arr.length;
        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - 1 - i; j++)
{
                if (arr[j] > arr[j + 1])
{
                    // Swap arr[j] and arr[j + 1]
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
//System.out.print(arr[i]+ " ");
}
}
}

    }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
5
12 35 69 48 51
12 35 48 51 69

Test Case - 2

User Output
3
5 -1 25
-1 5 25

S.No: 5	Exp. Name: ***Removing duplicate characters from a glitchy message by using StringBuffer constructor***	Date: 2024-09-04

Aim:

A group of friends sent a message to Jaya, but due to a technical glitch, the message contained duplicate characters. To overcome this kind of problem she decided to craft a Java program using a **StringBuffer** constructor that takes the glitchy message as input and returns a new string with all duplicate characters removed.

How can Jaya craft this program?

Input Format:

The input line reads a string representing the glitchy message.

Output Format:

The output is a string after removing the duplicate characters (Including special characters, space).

Note: Use println() to print the output statements.

Source Code:

q22135/Main.java

```
package q22135;
import java.util.HashSet;
import java.util.Scanner;
import java.util.Set;
public class Main{
        public static void main(String[] args){
                Scanner scan=new Scanner(System.in);
                String input=scan.nextLine();
                StringBuffer result=new StringBuffer();
                Set<Character>seen=new HashSet<>();
                for(char c:input.toCharArray()){
                        if(!seen.contains(c)){
                                seen.add(c);
                                        result.append(c);
                                }
                        }
                        System.out.println(result.toString());
                        scan.close();
                }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
abcdef
abcdef

Test Case - 2
User Output
a a b b c c
a bc

S.No: 6	Exp. Name: ***Constructor Overloading***	Date: 2024-09-04

Aim:

Write a Java program that demonstrates constructor overloading in a class named **Book**. The **Book** class should have the following attributes:

- title (String)
- author (String)
- pageCount (int)

Implement the following constructors in the **Book** class:

1. A default constructor that initializes the title and author to "Unknown Title" and "Unknown Author," respectively, and sets pageCount to 0.
2. A constructor that takes **title** and **author** as parameters and initializes the corresponding attributes. The **pageCount** should be set to 0.
3. A constructor that takes **title, author**, and **pageCount** as parameters and initializes the corresponding attributes.

Additionally, include a method named **displayBookDetails** that prints the details of the book, including the **title, author,** and **pageCount.**

Source Code:

`Book.java`

ID: 24K65A4702

```
import java.util.Scanner;

public class Book {




        private String title;
        private String author;
        private int pageCount;
        //default constructor
        public Book(){
                this.title = "Unknown Title";
                this.author = "Unknown Author";
                this.pageCount = 0;
        }
        //constructor with title and author
        public Book(String title, String author) {
                this.title = title;
                this.author = author;
                this.pageCount = pageCount;
        }
        //constructor with title,author,andpage Count
        public Book(String title, String author, int pageCount){
                this.title = title;
                this.author = author;
                this.pageCount = pageCount;
        }
        public void displayBookDetails(){
                System.out.println("Title: " + title);
                System.out.println("Author: " + author);
                System.out.println("Page Count: " + pageCount);
        }




    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Taking user inputs for Book 1
        System.out.println("Details for Book 1:");
        System.out.print("Title: ");
        String title1 = scanner.nextLine();
        System.out.print("Author: ");
        String author1 = scanner.nextLine();

        // Creating Book 1 using the constructor with title and author parameters
        Book book1 = new Book(title1, author1);

        // Taking user inputs for Book 2
        System.out.println("Details for Book 2:");
        System.out.print("Title: ");
        String title2 = scanner.nextLine();
        System.out.print("Author: ");
```

```
        int pageCount2 = scanner.nextInt();

        // Creating Book 2 using the constructor with all parameters
        Book book2 = new Book(title2, author2, pageCount2);

        // Displaying details of each book
        System.out.println("Book 1:");
        book1.displayBookDetails();

        System.out.println("Book 2:");
        book2.displayBookDetails();

        // Close the scanner
        scanner.close();
    }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Details for Book 1:
Title:
Introduction to Java Programming
Author:
John Smith
Details for Book 2:
Title:
Data Structures and Algorithms
Author:
Alice Johnson
Page Count:
350
Book 1:
Title: Introduction to Java Programming
Author: John Smith
Page Count: 0
Book 2:
Title: Data Structures and Algorithms
Author: Alice Johnson
Page Count: 350

Aim:

Write a Java program with a class name OverloadArea with overload methods area(float) and area(float, float) to find area of square and rectangle.

Write the main method within the class and assume that it will receive a total of 2 command line arguments of type float.

If the main() is provided with arguments : 1.34, 1.98 then the program should print the output as:
Area of square for side in meters 1.34 : 1.7956
Area of rectangle for length and breadth in meters 1.34, 1.98 : 2.6532001

Source Code:

OverloadArea.java

```
public class OverloadArea
{
public static float area(float side)
{
return (side*side);
}
public static float area (float len, float bre)
{
return (len*bre);
}
public static void main(String args[])
{
float f1=Float.parseFloat(args[0]);
float f2=Float.parseFloat(args[1]);
float f3=area (f1);
float f4=area (f1, f2);
System.out.println("Area of square for side in meters "+String.format("%.2f", f1) +":
"+String.format("%.4f",f3));
System.out.println("Area of rectangle for length and breadth in meters
"+String.format("%.2f", f1) +", "+String.format("%.2f", f2) +": "+String.format("%.7f",f4));
}
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Area of square for side in meters 1.34: 1.7956
Area of rectangle for length and breadth in meters 1.34, 1.98: 2.6532001

Test Case - 2
User Output
Area of square for side in meters 2.30: 5.2900
Area of rectangle for length and breadth in meters 2.30, 2.80: 6.4399996

S.No: 8	Exp. Name: ***Basics of Class and Object***	Date: 2024-09-01

Aim:

The Spy Academy is implementing a new system to manage information about their agents. As part of this system, they need to create a class to represent an agent. Imagine yourself as the chief developer, and your task is to design a basic class named **Agent**.

Task:

Design a class named Agent with the following attributes: name (String), age (int), codename (String), and isUndercover (boolean).

Implement a method named **introduce()** that prints a message introducing the agent, including their name, age, and codename.

Additional Instructions:

Create an instance of the **Agent** class named **topAgent** in the main method.

Utilize user input within the main method to set the attributes of topAgent (name, age, codename, and isUndercover).

Invoke the **introduce()** method on **topAgent** to introduce this top agent to the team.

Source Code:

```
q22852/SpySystem.java
```


ID: 24K65A4702
2023-2027-CIC
Sasi Institute of Technology and Engineering (Autonomous)

```
package q22852;
import java.util.Scanner;

// create class Agent and Attributes
class Agent {
    // Attributes
String name;
int age;
String codename;
boolean isUndercover;

    // create Method to introduce the agent

    void introduce(){
     System.out.println("Hello, I am Agent " +codename+ "."); // write your code here
        System.out.println("My real name is " +name+ "."); // write your code here
        System.out.println("I am " +age+ " years old."); // write your code here
    }
}

public class SpySystem {
    public static void main(String[] args) {
        // Create an instance of Agent
        Agent topAgent=new Agent();
        Scanner scanner = new Scanner(System.in);
        System.out.print("name: ");
        topAgent.name = scanner.nextLine();
        System.out.print("age: ");
        topAgent.age = scanner.nextInt();
        scanner.nextLine();
        System.out.print("codename: ");
        topAgent.codename = scanner.nextLine();
        System.out.print("undercover? (true/false): ");
        topAgent.isUndercover = scanner.nextBoolean();


        // Introducing the topAgent
        topAgent.introduce();
        scanner.close();
    }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
name:
Alice
age:
27
codename:
007

```
undercover? (true/false):
```

true

```
Hello, I am Agent 007.
My real name is Alice.
I am 27 years old.
```

S.No: 9	Exp. Name: ***Constructor overloading and default constructor***	Date: 2024-10-23

Aim:

Define a class named **Test** with two private integer variables, **i** and **j**. Implement the following:

- A default constructor that initializes **i** and **j** to 0.
- A parameterized constructor that takes an integer **i** and initializes it with the provided value, setting **j** to 0.
- A parameterized constructor that takes two integers, **i** and **j**, and initializes both instance variables accordingly.

Additionally, implement a method **display()** that prints the values of **i** and **j**.

In the **Main** class, use the provided **Scanner** code to:

- Create an object **obj1** using the default constructor and display its values.
- Accept an integer from the user, create **obj2** using the parameterized constructor with a single argument, and display its values.
- Accept two integers from the user, create **obj3** using the parameterized constructor with two arguments, and display its values.

Source Code:

q21947/Main.java

ID: 24K65A4702
2023-2027-CIC
Sasi Institute of Technology and Engineering (Autonomous)

```
package q21947;
import java.util.Scanner;
class Test {
    // Private instance variables
    private int i;
    private int j;
Test() {
    i=j-0;
}
    Test(int i) {
         this.i=i;
   }
    Test(int i,int j) {
         this.i=i;
         this.j=j;
    }
    void display() {
        System.out.println("Value of i: "+i);
        System.out.println("Value of j: "+j);
        }
}
public class Main {
    public static void main(String[] args) {
Scanner input=new Scanner(System.in);
                int i,j;
                Test obj1=new Test();
                System.out.println("Displaying values for obj1:");
                obj1.display();
                System.out.print("Enter a value for i: ");
                i=input.nextInt();
                Test obj2=new Test(i);
                System.out.println("Displaying values for obj2:");
                obj2.display();
                System.out.print("Enter a value for i: ");
                i=input.nextInt();
                System.out.print("Enter a value for j: ");
                j=input.nextInt();
                Test obj3=new Test(i,j);
                System.out.println("Displaying values for obj3:");
                obj3.display();

    }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Displaying values for obj1:
Value of i: 0
Value of j: 0
Enter a value for i:

10
Displaying values for obj2:
Value of i: 10
Value of j: 0
Enter a value for i:
40
Enter a value for j:
50
Displaying values for obj3:
Value of i: 40
Value of j: 50

Test Case - 2
User Output
Displaying values for obj1:
Value of i: 0
Value of j: 0
Enter a value for i:
10
Displaying values for obj2:
Value of i: 10
Value of j: 0
Enter a value for i:
20
Enter a value for j:
30
Displaying values for obj3:
Value of i: 20
Value of j: 30

S.No: 10	Exp. Name: ***Number and Square - Single Inheritance***	Date: 2024-10-22

Aim:

Create a class named **NumberIn** with the following attributes:

- An int variable **num**.
- A method **inputNum()** that takes user input for the number.

Create a class named SquareOut that inherits from **NumberIn** and has the following characteristics:

- A method **displaySquare()** that prints the square of the entered number to the console.

Note: The main class has been provided to you in the editor.

Source Code:

q23112/MainFunction.java

```
package q23112;
import java.util.Scanner;

        class NumberIn{
                int num;
                void inputNum(){
                Scanner scanner = new Scanner(System.in);
                System.out.print("Enter number: ");
                num = scanner.nextInt();

}
        }
        class SquareOut extends NumberIn{
                void displaySquare(){
        int square = num * num;
System.out.println(square);

}
        }
public class MainFunction {
    public static void main(String[] args) {
        SquareOut squareout = new SquareOut();
        squareout.inputNum();
        squareout.displaySquare();
    }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter number:
9
81

Test Case - 2
User Output
Enter number:
15
225

S.No: 11	Exp. Name: ***Multilevel Inheritance in Java***	Date: 2024-10-22

Aim:

Write a Java program to show multilevel inheritance to manage student information and calculate their marks. The program should include the following functionalities:

Student Class:

- Contains student ID and name.
- Methods to set and display student data.

Marks Class (inherits from Student):

- Contains marks for Java, C, and C++.
- Methods to set and display marks.

Result Class (inherits from Marks):

- Calculates the total and average marks.
- Method to display the total and average.

Input Format:

The program should prompt the user for the following inputs:

- Student ID (integer)
- Student Name (string)
- Java Marks (float)
- C Marks (float)
- C++ Marks (float)

Output Format:

The program should output the following:

- Student ID in the format **Id : <id>**
- Student Name in the format **Name : <name>**
- Java Marks in the format **Java marks : <java marks>**
- C Marks in the format **C marks : <c marks>**
- C++ Marks in the format **Cpp marks : <cpp marks>**
- Total Marks in the format **Total : <total>**
- Average Marks in the format **Avg : <avg>**

Source Code:

`MultilevelInheritance.java`

```
import java.util.Scanner;
public class MultilevelInheritance{
        public static void main(String args[]) {
            Scanner s=new Scanner(System.in);
            Scanner sc=new Scanner(System.in);
                Result r = new Result();
                System.out.print("Id: ");
                int id=s.nextInt();
                System.out.print("Name: ");
                String name = sc.nextLine();
                r.setData(id, name);
                System.out.print("Java marks: ");
                float jv=s.nextFloat();
                System.out.print("C marks: ");
                float c=s.nextFloat();
                System.out.print("CPP marks: ");
                float cpp=s.nextFloat();
        r.displayData();
                r.setMarks(jv,c,cpp);
                r.displayMarks();
                r.compute();
                r.showResult();
        }
}
class Student{
        int id;
        String name;
        public void setData(int id,String name){
                this.id=id;
                this.name=name;
        }
        public void displayData(){
                System.out.println("Id : "+id+"\n"+"Name : "+name);
        }
}

class Marks extends Student{
        float jv,c,cpp;
        public void setMarks(float jv,float c,float cpp){
                this.jv=jv;
                this.c=c;
                this.cpp=cpp;
        }
        public void displayMarks(){
                System.out.println("Java marks : "+jv+"\n"+"C marks : "+c+"\n"+"Cpp marks :
"+cpp);
        }
}
class Result extends Marks{
        float tot,avg;
        public void compute(){
                tot=jv+c+cpp;
                avg=(tot)/3;
}
        public void showResult(){
```

```
}
// Write your code here
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Id:
1001
Name:
Yamuna
Java marks:
75.5
C marks:
96.5
CPP marks:
87
Id : 1001
Name : Yamuna
Java marks : 75.5
C marks : 96.5
Cpp marks : 87.0
Total : 259.0
Avg : 86.333336

Test Case - 2
User Output
Id:
1000
Name:
Ganga
Java marks:
77
C marks:
85.5
CPP marks:
96.5
Id : 1000
Name : Ganga
Java marks : 77.0
C marks : 85.5
Cpp marks : 96.5
Total : 259.0
Avg : 86.333336

S.No: 12	Exp. Name: ***Areas of Shapes Using Abstract Class***	Date: 2024-10-22

Aim:

Create an abstract class called **Shape.**

- Declare two abstract methods of type double: **calculateArea()** and **calculatePerimeter()**. These methods will be implemented by subclasses.
- Implement a concrete method named **displayDetails()** that displays information about the shape, including its area and perimeter.

Next, create a subclass called **Circle** that extends the Shape class:

- Implement the **calculateArea()** method in this subclass. It should calculate and return the area of the circle.
- Implement the **calculatePerimeter()** method in this subclass. It should calculate and return the perimeter of the circle.

Note: The main method is already provided

Source Code:

Circle.java

```
import java.lang.Math;
import java.util.Scanner;
abstract class Shape {
        abstract double calculateArea();
        abstract double calculatePerimeter();
        public void displayDetails(){
                System.out.println("Shape details:");
                System.out.println("Area: "+calculateArea());
                System.out.println("Perimeter: "+calculatePerimeter());
        }
}
class Circle extends Shape {
    double radius;
    public Circle(double radius) {
        this.radius = radius;
    }
    @Override
    double calculateArea() {
        return(Math.PI*radius*radius);
    }
    @Override
    double calculatePerimeter() {

                return(2*Math.PI*radius);

    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the radius of the circle: ");
        double radius = scanner.nextDouble();
        Circle circle = new Circle(radius);
        circle.displayDetails();
        scanner.close();
    }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the radius of the circle:
3
Shape details:
Area: 28.274333882308138
Perimeter: 18.84955592153876

Test Case - 2
User Output
Enter the radius of the circle:
4.2
Shape details:
Area: 55.41769440932395
Perimeter: 26.389378290154262