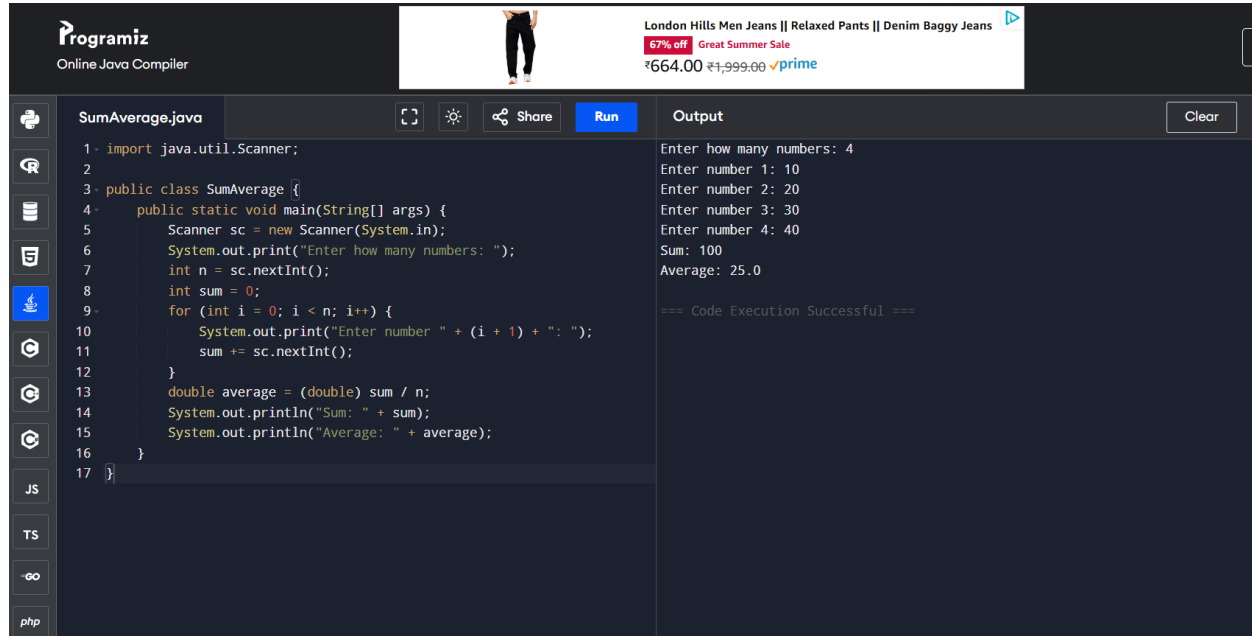


1. Write a program to find the average and sum of the N numbers using Command line Argument.



The screenshot shows the Programiz Online Java Compiler interface. The code editor contains a Java program named `SumAverage.java` that takes command-line arguments to calculate the sum and average of N numbers. The output window shows the program's execution with the following input and output:

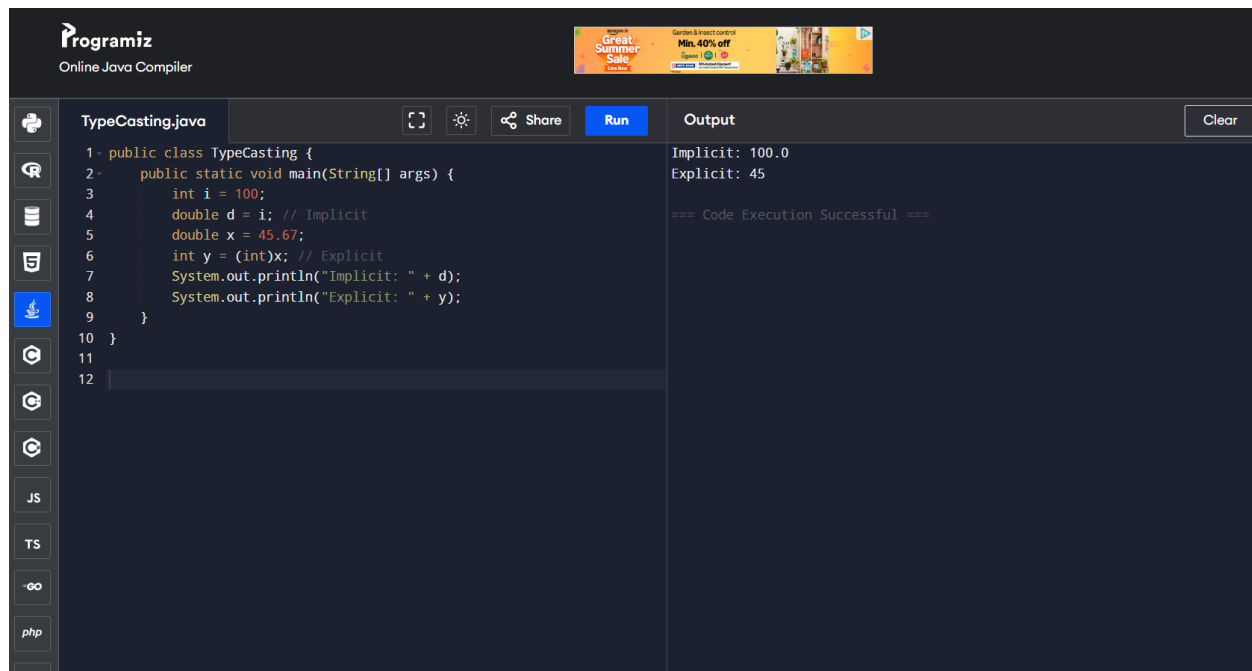
```
1- import java.util.Scanner;
2
3- public class SumAverage {
4-     public static void main(String[] args) {
5-         Scanner sc = new Scanner(System.in);
6-         System.out.print("Enter how many numbers: ");
7-         int n = sc.nextInt();
8-         int sum = 0;
9-         for (int i = 0; i < n; i++) {
10-             System.out.print("Enter number " + (i + 1) + ": ");
11-             sum += sc.nextInt();
12-         }
13-         double average = (double) sum / n;
14-         System.out.println("Sum: " + sum);
15-         System.out.println("Average: " + average);
16-     }
17- }
```

Output:

```
Enter how many numbers: 4
Enter number 1: 10
Enter number 2: 20
Enter number 3: 30
Enter number 4: 40
Sum: 100
Average: 25.0

=== Code Execution Successful ===
```

2. Write a program to demonstrate type casting.



The screenshot shows the Programiz Online Java Compiler interface. The code editor contains a Java program named `TypeCasting.java` that demonstrates implicit and explicit type casting. The output window shows the program's execution with the following input and output:

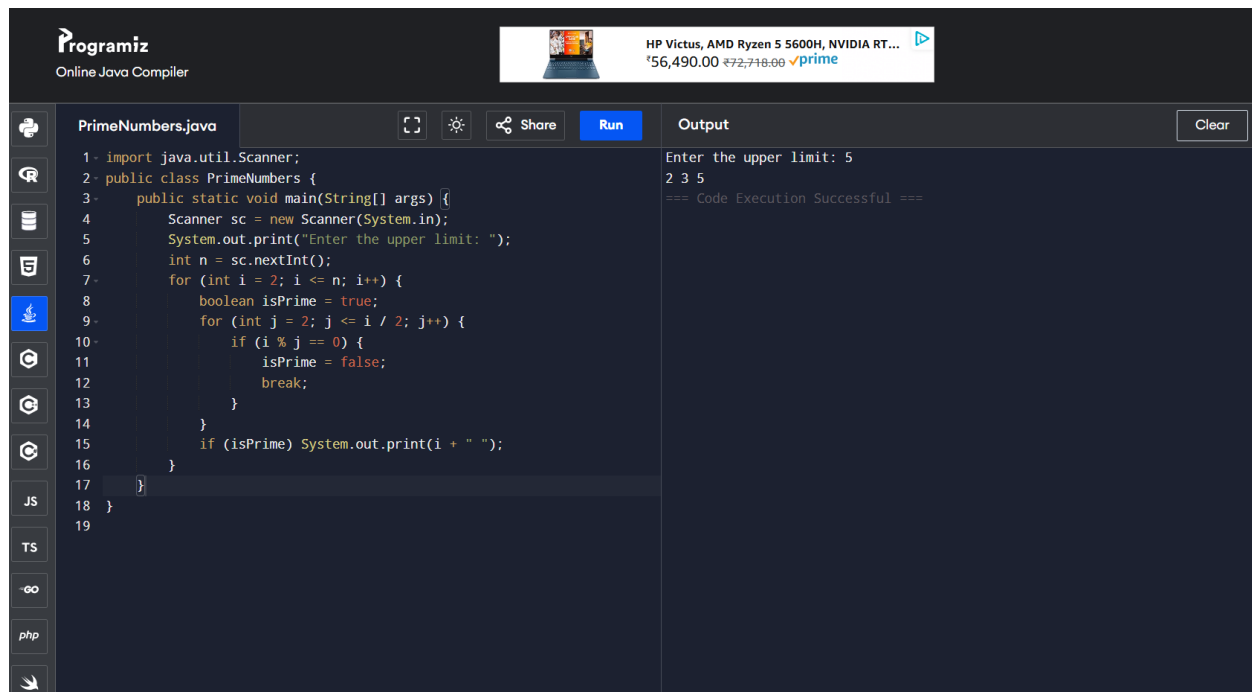
```
1- public class TypeCasting {
2-     public static void main(String[] args) {
3-         int i = 100;
4-         double d = i; // Implicit
5-         double x = 45.67;
6-         int y = (int)x; // Explicit
7-         System.out.println("Implicit: " + d);
8-         System.out.println("Explicit: " + y);
9-     }
10- }
11
12
```

Output:

```
Implicit: 100.0
Explicit: 45

=== Code Execution Successful ===
```

3. Write a program to generate prime numbers between 1 & given number.

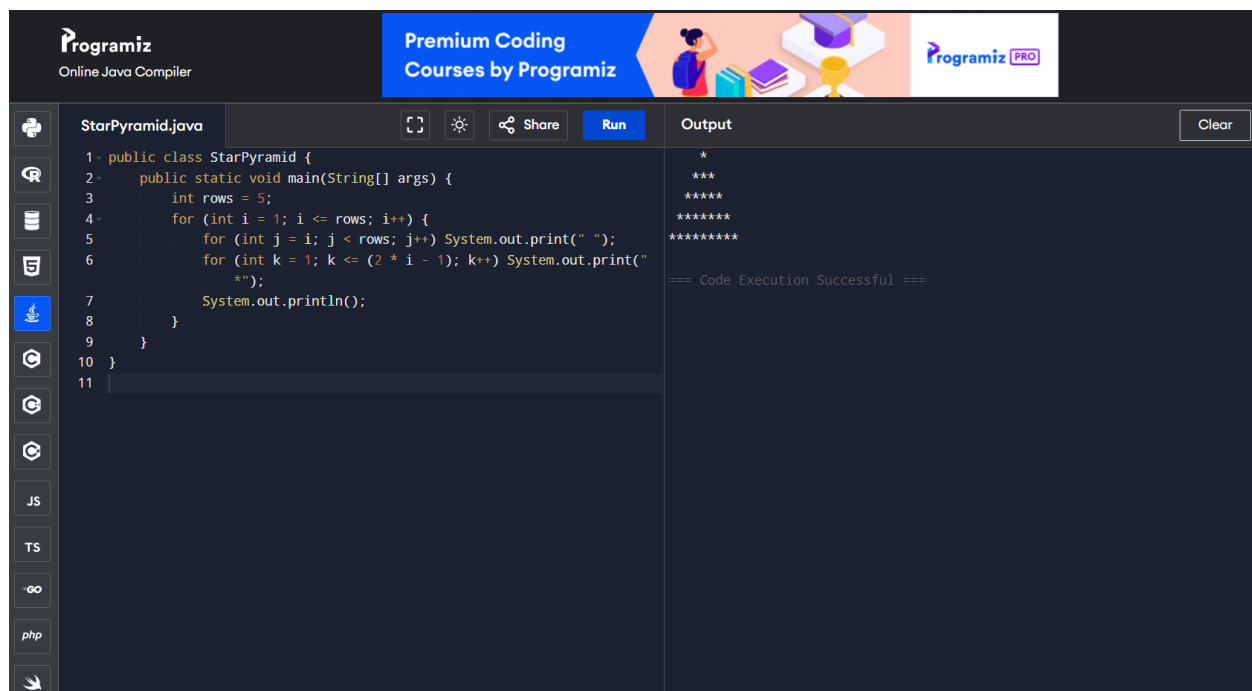


The screenshot shows the Programiz Online Java Compiler interface. The code editor contains a Java program named PrimeNumbers.java. The program prompts the user to enter an upper limit, which is 5. It then prints the prime numbers 2, 3, and 5. The output panel shows the execution results, confirming that the code executed successfully.

```
1- import java.util.Scanner;
2- public class PrimeNumbers {
3-     public static void main(String[] args) {
4-         Scanner sc = new Scanner(System.in);
5-         System.out.print("Enter the upper limit: ");
6-         int n = sc.nextInt();
7-         for (int i = 2; i <= n; i++) {
8-             boolean isPrime = true;
9-             for (int j = 2; j <= i / 2; j++) {
10-                 if (i % j == 0) {
11-                     isPrime = false;
12-                     break;
13-                 }
14-             }
15-             if (isPrime) System.out.print(i + " ");
16-         }
17-     }
18- }
```

Output: Enter the upper limit: 5
2 3 5
=== Code Execution Successful ===

4. Write a program to generate pyramid of stars using nested for loops.



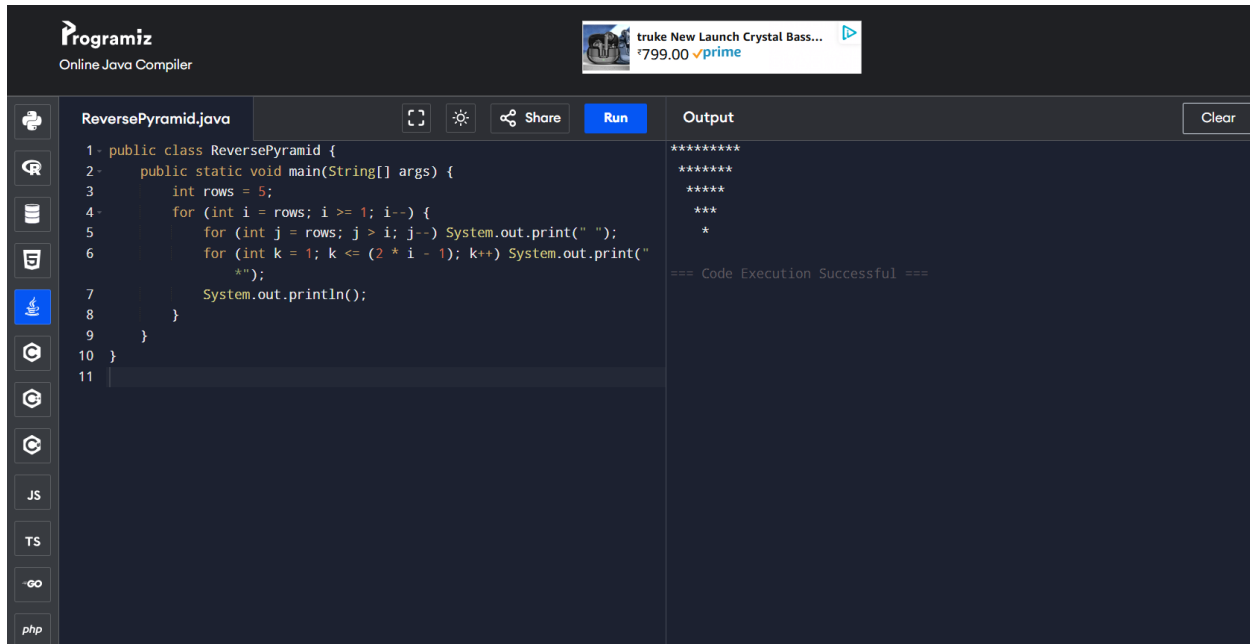
The screenshot shows the Programiz Online Java Compiler interface. The code editor contains a Java program named StarPyramid.java. The program generates a pyramid of stars with 5 rows. The output panel shows the execution results, confirming that the code executed successfully.

```
1- public class StarPyramid {
2-     public static void main(String[] args) {
3-         int rows = 5;
4-         for (int i = 1; i <= rows; i++) {
5-             for (int j = i; j <= rows; j++) System.out.print(" ");
6-             for (int k = 1; k <= (2 * i - 1); k++) System.out.print("
              *");
7-             System.out.println();
8-         }
9-     }
10- }
```

Output: *

=== Code Execution Successful ===

5. Write a program to reversed pyramid using for loops & decrement operator.



The screenshot shows the Programiz Online Java Compiler interface. The file name is "ReversePyramid.java". The code is as follows:

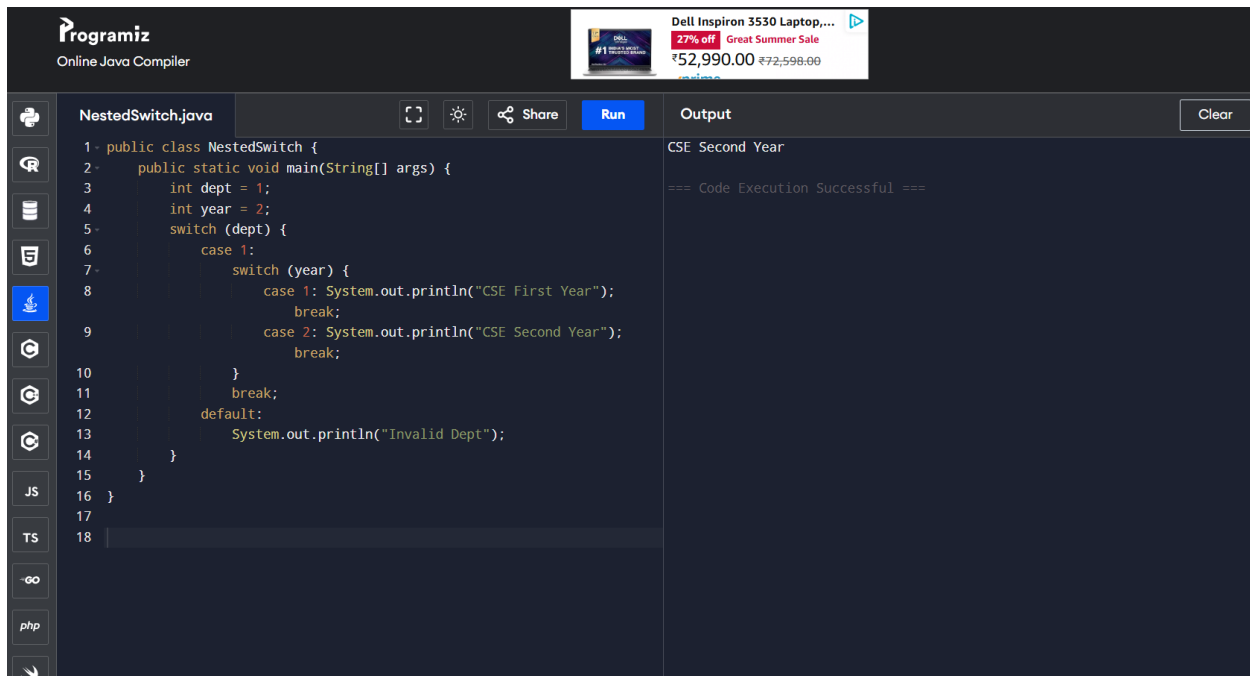
```
1- public class ReversePyramid {
2-     public static void main(String[] args) {
3-         int rows = 5;
4-         for (int i = rows; i >= 1; i--) {
5-             for (int j = rows; j > i; j--) System.out.print(" ");
6-             for (int k = 1; k <= (2 * i - 1); k++) System.out.print("
7-                 *");
8-             System.out.println();
9-         }
10-     }
11- }
```

The output shows a reversed pyramid of asterisks:

```
*****
*****
*****
***
*
```

Below the output, it says "=== Code Execution Successful ===".

6. Write a program for demonstrate Nested Switch .

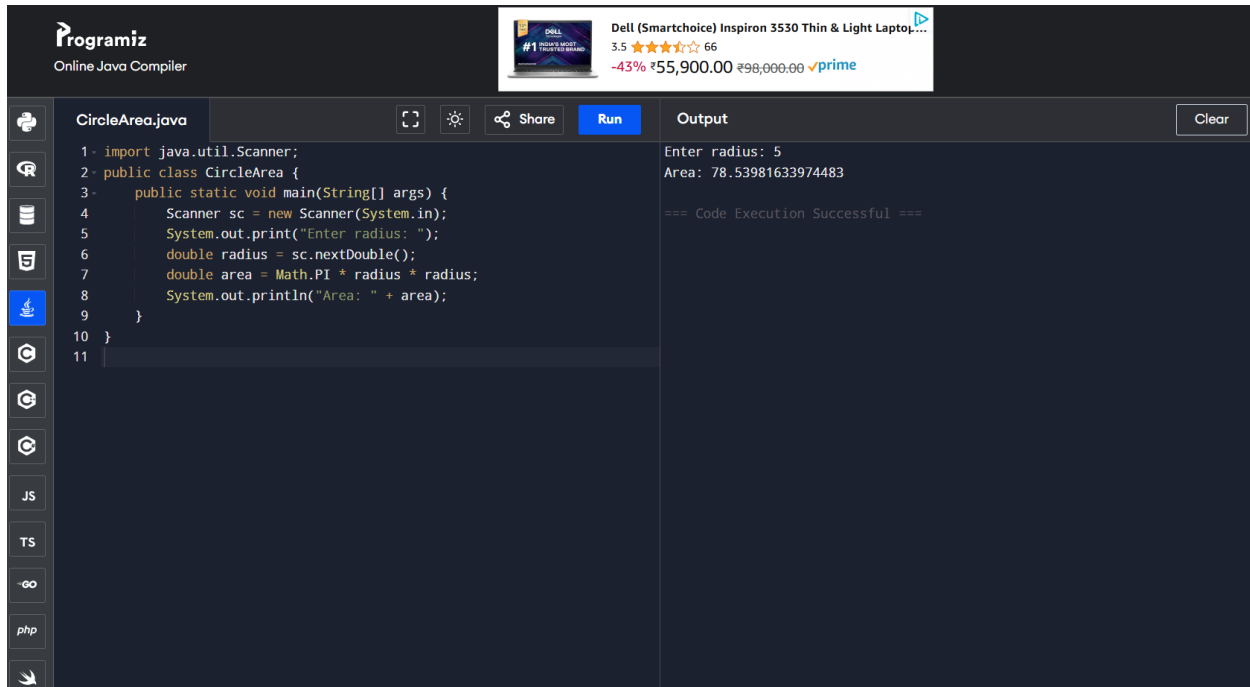


The screenshot shows the Programiz Online Java Compiler interface. The file name is "NestedSwitch.java". The code is as follows:

```
1- public class NestedSwitch {
2-     public static void main(String[] args) {
3-         int dept = 1;
4-         int year = 2;
5-         switch (dept) {
6-             case 1:
7-                 switch (year) {
8-                     case 1: System.out.println("CSE First Year");
9-                         break;
10-                     case 2: System.out.println("CSE Second Year");
11-                         break;
12-                 }
13-             default:
14-                 System.out.println("Invalid Dept");
15-         }
16-     }
17- }
```

The output shows "CSE Second Year". Below the output, it says "=== Code Execution Successful ===".

7. Write a program to calculate area of a circle using radius .



The screenshot shows the Programiz Online Java Compiler interface. At the top, there's a banner for a Dell laptop. Below the banner, the file name "CircleArea.java" is displayed. The code editor contains the following Java code:

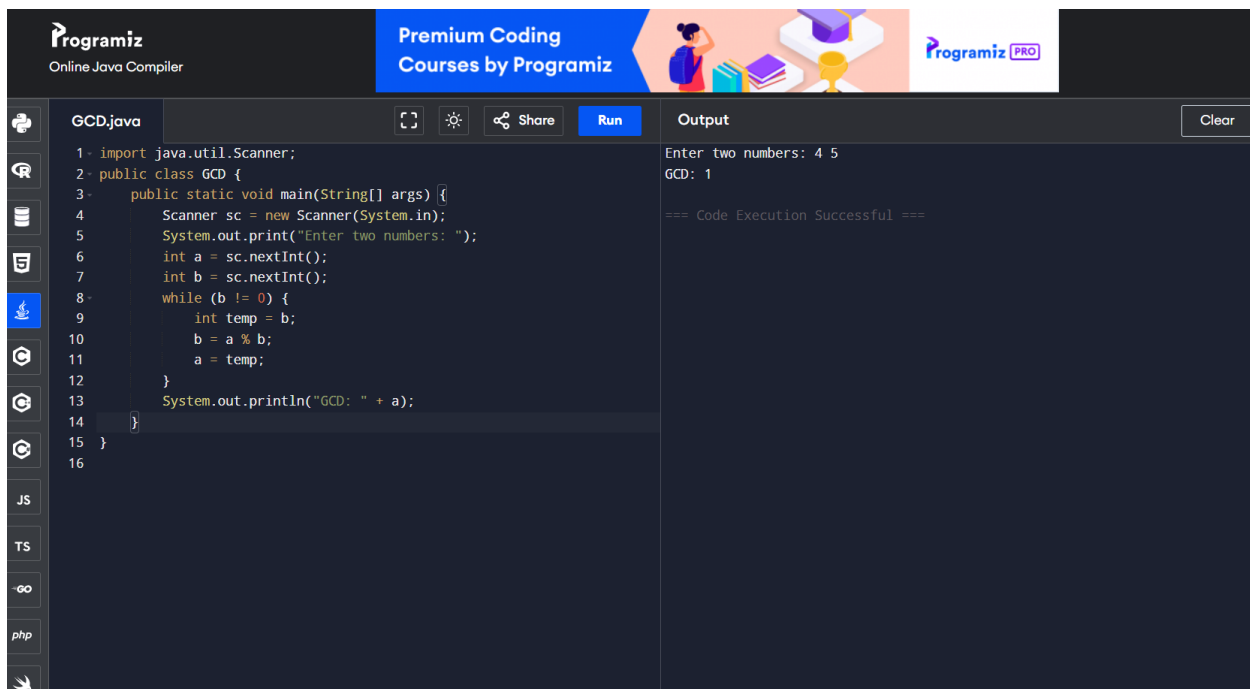
```
1- import java.util.Scanner;
2- public class CircleArea {
3-     public static void main(String[] args) {
4-         Scanner sc = new Scanner(System.in);
5-         System.out.print("Enter radius: ");
6-         double radius = sc.nextDouble();
7-         double area = Math.PI * radius * radius;
8-         System.out.println("Area: " + area);
9-     }
10- }
11-
```

The "Run" button is highlighted. The output panel on the right shows the execution results:

```
Enter radius: 5
Area: 78.53981633974483

=== Code Execution Successful ===
```

8. Write a program to find G.C.D of the number.



The screenshot shows the Programiz Online Java Compiler interface. At the top, there's a banner for "Premium Coding Courses by Programiz". Below the banner, the file name "GCD.java" is displayed. The code editor contains the following Java code:

```
1- import java.util.Scanner;
2- public class GCD {
3-     public static void main(String[] args) {
4-         Scanner sc = new Scanner(System.in);
5-         System.out.print("Enter two numbers: ");
6-         int a = sc.nextInt();
7-         int b = sc.nextInt();
8-         while (b != 0) {
9-             int temp = b;
10-            b = a % b;
11-            a = temp;
12-        }
13-         System.out.println("GCD: " + a);
14-     }
15- }
16-
```

The "Run" button is highlighted. The output panel on the right shows the execution results:

```
Enter two numbers: 4 5
GCD: 1

=== Code Execution Successful ===
```

9. Write a program to design a class account using the inheritance and static members which show all functions of a bank (Withdrawl, deposit)

```
1- import java.util.Scanner;
2-
3- class Bank {
4-     static String bankName = "State Bank of India";
5-
6-     void showBankName() {
7-         System.out.println("Bank: " + bankName);
8-     }
9- }
10-
11- class Account extends Bank {
12-     int accNumber;
13-     String holderName;
14-     double balance;
15-
16-     Account(int accNumber, String holderName, double balance) {
17-         this.accNumber = accNumber;
18-         this.holderName = holderName;
19-         this.balance = balance;
20-     }
21-
22-     void deposit(double amount) {
23-         balance += amount;
24-         System.out.println("₹" + amount + " deposited. Current balance: ₹" + balance);
25-     }
26-
27-     void withdraw(double amount) {
28-         if (amount <= balance) {
29-             balance -= amount;
30-             System.out.println("₹" + amount + " withdrawn. Current balance: ₹" + balance);
31-         } else {
32-             System.out.println("Insufficient balance!");
33-         }
34-     }
35-
36-     void display() {
```

Enter account number: 8527926166
Exception in thread "main" java.util.InputMismatchException: For input string: "8527926166"
at java.base/java.util.Scanner.nextInt(Scanner.java:2290)
at java.base/java.util.Scanner.nextInt(Scanner.java:2238)
at BankApp.main(BankApp.java:48)

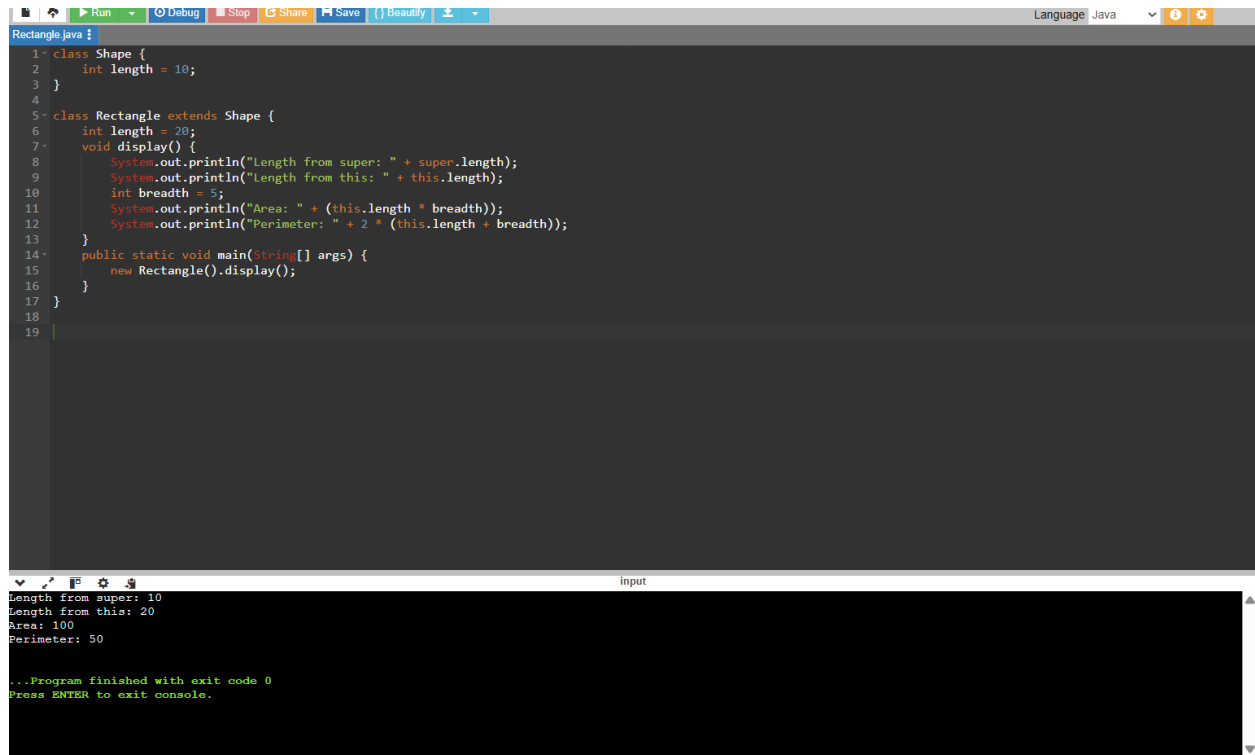
...Program finished with exit code 1
Press ENTER to exit console.

```
34-     }
35-
36-     void display() {
37-         System.out.println("Account Number: " + accNumber);
38-         System.out.println("Holder Name: " + holderName);
39-         System.out.println("Balance: ₹" + balance);
40-     }
41- }
42-
43- public class BankApp {
44-     public static void main(String args[]) {
45-         Scanner sc = new Scanner(System.in);
46-
47-         System.out.print("Enter account number: ");
48-         int accNo = sc.nextInt();
49-
50-         sc.nextLine();
51-         System.out.print("Enter account holder name: ");
52-         String name = sc.nextLine();
53-
54-         System.out.print("Enter initial balance: ");
55-         double bal = sc.nextDouble();
56-
57-         Account acc = new Account(accNo, name, bal);
58-         acc.showBankName();
59-         acc.display();
60-
61-         System.out.print("Enter amount to deposit: ");
62-         double dep = sc.nextDouble();
63-         acc.deposit(dep);
64-
65-         System.out.print("Enter amount to withdraw: ");
66-         double with = sc.nextDouble();
67-         acc.withdraw(with);
68-     }
69- }
```

Enter account number: 8527926166
Exception in thread "main" java.util.InputMismatchException: For input string: "8527926166"
at java.base/java.util.Scanner.nextInt(Scanner.java:2290)
at java.base/java.util.Scanner.nextInt(Scanner.java:2238)
at BankApp.main(BankApp.java:48)

...Program finished with exit code 1
Press ENTER to exit console.

10. Write a program to create a simple class to find out the area and perimeter of rectangle

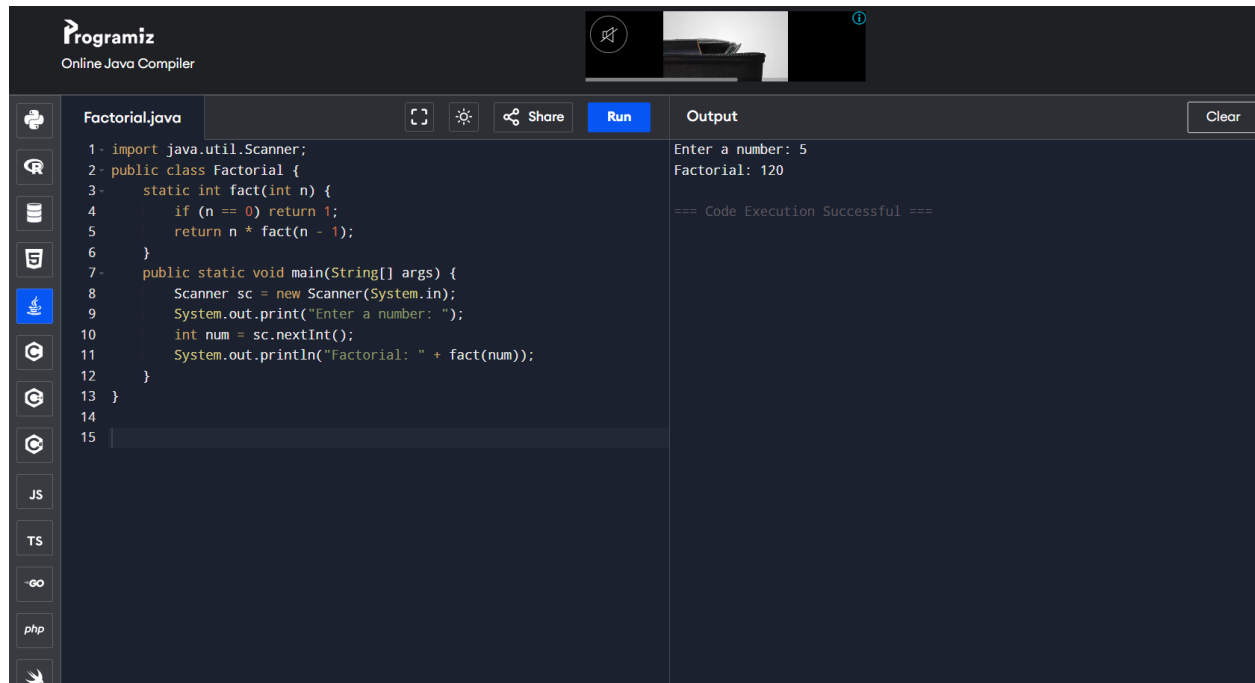


The screenshot shows an IDE window titled "Rectangle.java". The code defines a `Shape` class with an `int length = 10;` and a `Rectangle` class that extends `Shape`. The `Rectangle` class has an `int length = 20;`, an `int breadth = 5;`, and a `display()` method that prints the length from the superclass, the length from the subclass, the area, and the perimeter. A `main` method creates a `Rectangle` object and calls `display()`. The output window shows the following text:

```
Length from super: 10
Length from this: 20
Area: 100
Perimeter: 50

...Program finished with exit code 0
Press ENTER to exit console.
```

11. Write a program to find the factorial of a given number using recursion.



The screenshot shows the Programiz Online Java Compiler interface. The code in `Factorial.java` is as follows:

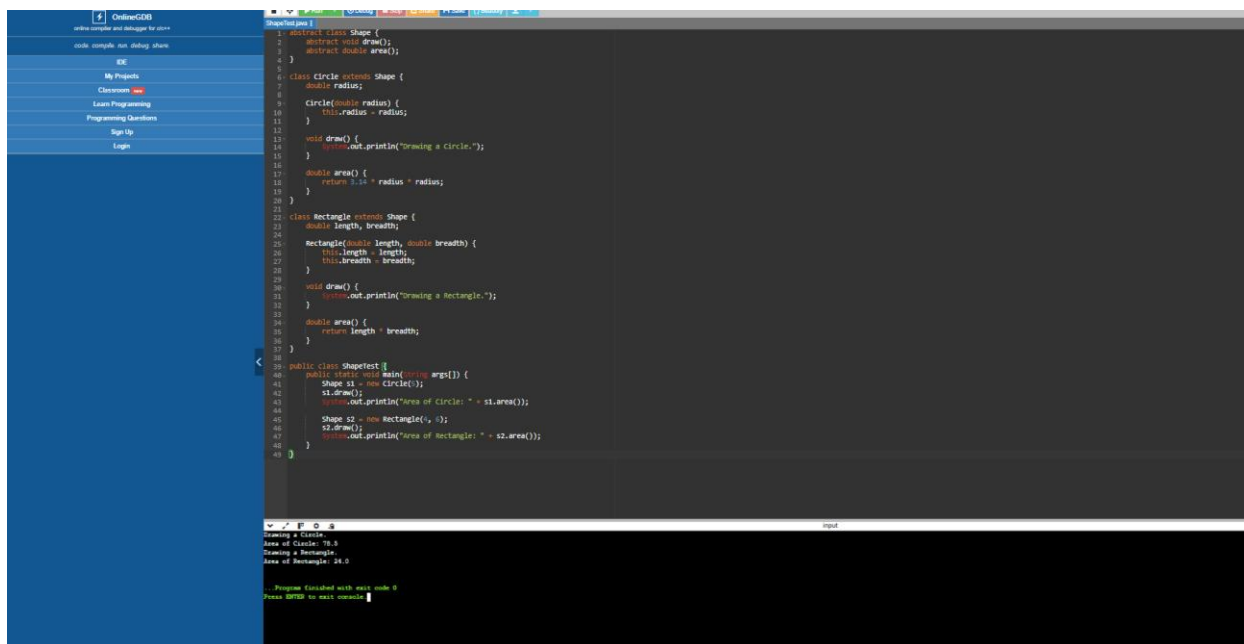
```
1- import java.util.Scanner;
2- public class Factorial {
3-     static int fact(int n) {
4-         if (n == 0) return 1;
5-         return n * fact(n - 1);
6-     }
7-     public static void main(String[] args) {
8-         Scanner sc = new Scanner(System.in);
9-         System.out.print("Enter a number: ");
10-        int num = sc.nextInt();
11-        System.out.println("Factorial: " + fact(num));
12-    }
13- }
14-
15-
```

The output window shows the following text:

```
Enter a number: 5
Factorial: 120

=== Code Execution Successful ===
```

12. Write a program to design a class using abstract methods and abstract classes.



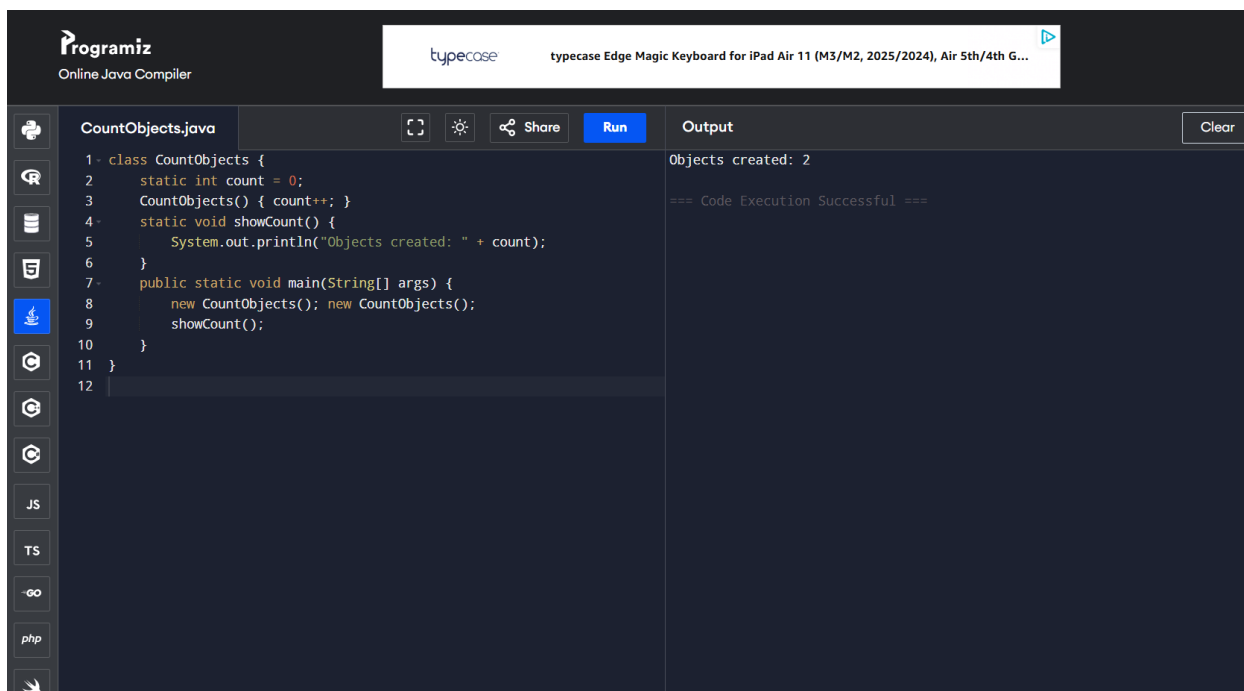
```
1 abstract class Shape {
2     abstract void draw();
3     abstract double area();
4 }
5
6 class Circle extends Shape {
7     double radius;
8     Circle(double radius) {
9         this.radius = radius;
10    }
11
12    void draw() {
13        System.out.println("Drawing a circle.");
14    }
15
16    double area() {
17        return 3.14 * radius * radius;
18    }
19 }
20
21 class Rectangle extends Shape {
22     double length, breadth;
23     Rectangle(double length, double breadth) {
24         this.length = length;
25         this.breadth = breadth;
26     }
27
28    void draw() {
29        System.out.println("Drawing a Rectangle.");
30    }
31
32    double area() {
33        return length * breadth;
34    }
35 }
36
37 public class ShapeTest {
38     public static void main(String args[]) {
39         Shape s1 = new Circle(3);
40         s1.draw();
41         System.out.println("Area of Circle: " + s1.area());
42
43         Shape s2 = new Rectangle(4, 3);
44         s2.draw();
45         System.out.println("Area of Rectangle: " + s2.area());
46     }
47 }
```

Output:

```
Drawing a circle.
Area of Circle: 78.5
Drawing a Rectangle.
Area of Rectangle: 24.0
```

Programme finished with exit code 0
Press ENTER to exit console

13. Write a program to count the number of objects created for a class using static member

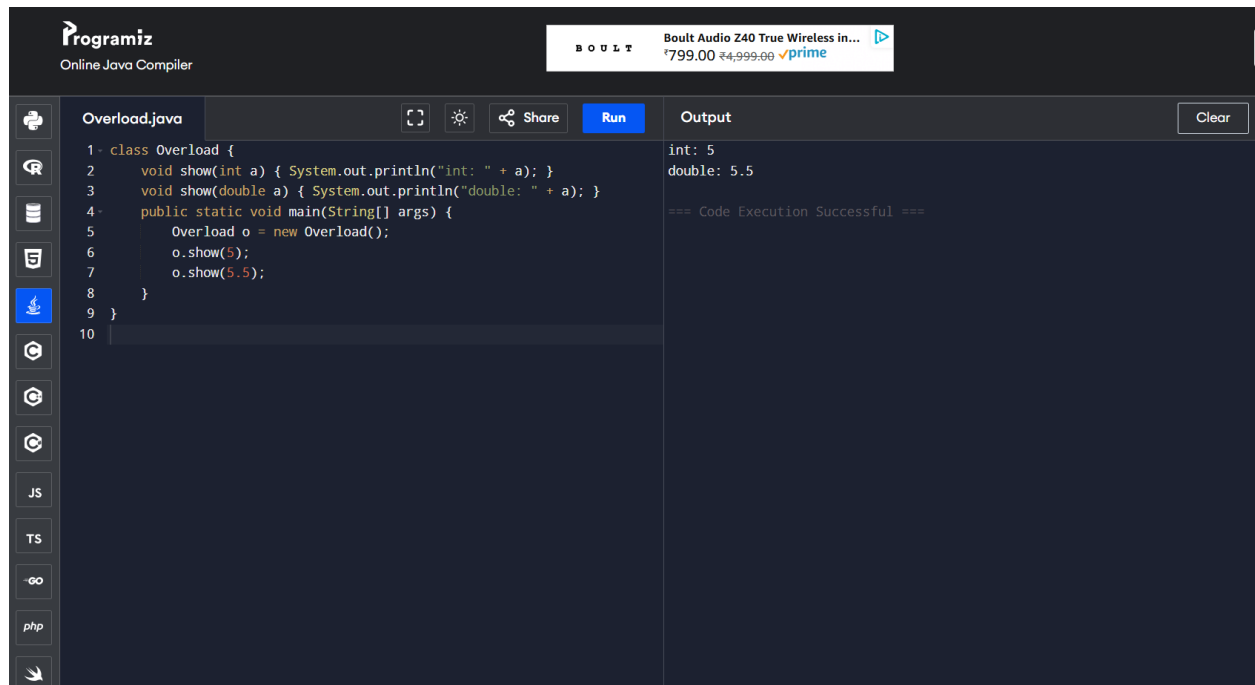


```
1 class CountObjects {
2     static int count = 0;
3     CountObjects() { count++; }
4     static void showCount() {
5         System.out.println("Objects created: " + count);
6     }
7     public static void main(String[] args) {
8         new CountObjects(); new CountObjects();
9         showCount();
10    }
11 }
12
```

Output:

```
Objects created: 2
=== Code Execution Successful ===
```

14. Write a program to demonstrate the use of function overloading.



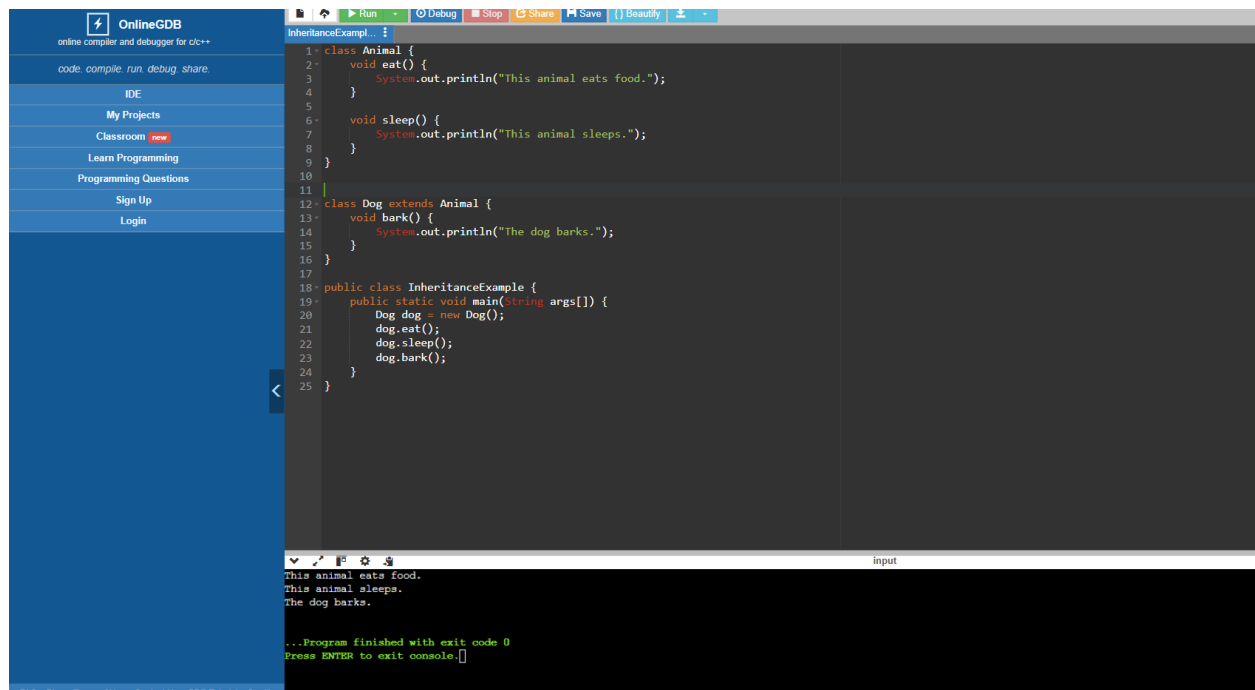
The screenshot shows the Programiz Online Java Compiler interface. The code editor contains a Java program named 'Overload.java' that demonstrates function overloading. The program defines a class 'Overload' with two methods: 'show(int a)' and 'show(double a)'. The 'main' method creates an instance of 'Overload' and calls both 'show' methods with different arguments (5 and 5.5). The output window shows the results of the program execution.

```
1- class Overload {
2-     void show(int a) { System.out.println("int: " + a); }
3-     void show(double a) { System.out.println("double: " + a); }
4-     public static void main(String[] args) {
5-         Overload o = new Overload();
6-         o.show(5);
7-         o.show(5.5);
8-     }
9- }
10
```

Output:

```
int: 5
double: 5.5
=== Code Execution Successful ===
```

15. Write a program to demonstrate the use of inheritance



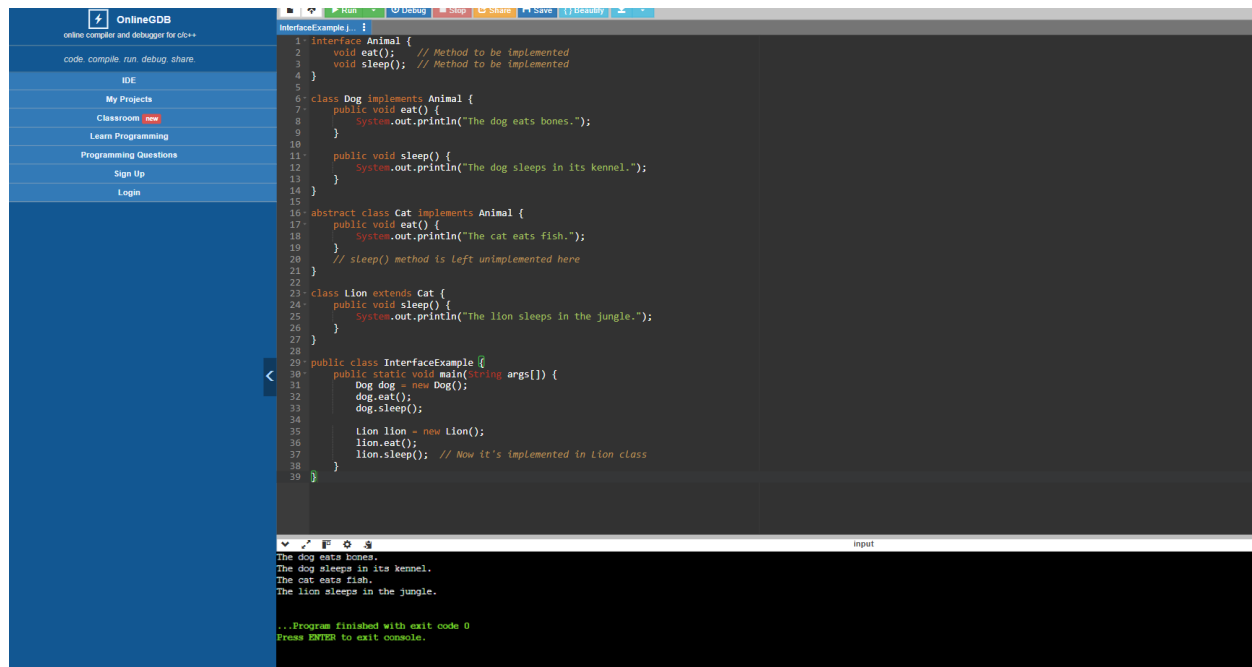
The screenshot shows the OnlineGDB online compiler and debugger interface. The code editor contains a Java program named 'InheritanceExample.java' that demonstrates inheritance. The program defines a base class 'Animal' with methods 'eat()' and 'sleep()'. A derived class 'Dog' extends 'Animal' and overrides the 'bark()' method. The 'InheritanceExample' class contains the 'main' method, which creates a 'Dog' object and calls its methods. The output window shows the results of the program execution.

```
1- class Animal {
2-     void eat() {
3-         System.out.println("This animal eats food.");
4-     }
5-     void sleep() {
6-         System.out.println("This animal sleeps.");
7-     }
8- }
9-
10-
11- class Dog extends Animal {
12-     void bark() {
13-         System.out.println("The dog barks.");
14-     }
15- }
16-
17-
18- public class InheritanceExample {
19-     public static void main(String args[]) {
20-         Dog dog = new Dog();
21-         dog.eat();
22-         dog.sleep();
23-         dog.bark();
24-     }
25- }
```

Output:

```
This animal eats food.
This animal sleeps.
The dog barks.
...Program finished with exit code 0
Press ENTER to exit console.
```


16. Write a program that show the partial implementation of Interface

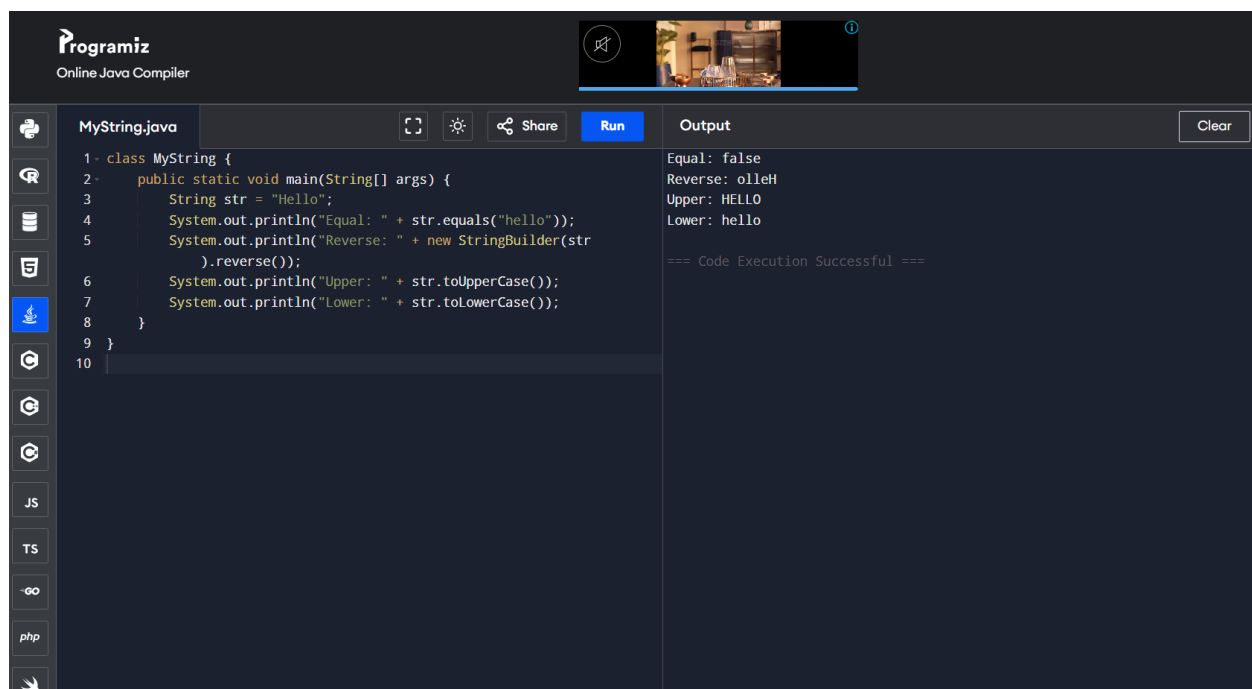


The screenshot shows the OnlineGDB IDE interface. On the left is a sidebar with navigation links: OnlineGDB, code, compile, run, debug, share, IDE, My Projects, Classroom, Learn Programming, Programming Questions, Sign Up, and Login. The main editor displays a Java program titled 'InterfaceExample.java'. The code defines an 'Animal' interface with 'eat()' and 'sleep()' methods. It then shows 'Dog' and 'Cat' classes implementing these methods, and a 'Lion' class extending 'Cat'. A 'main' method in 'InterfaceExample' class creates instances of 'Dog' and 'Lion' and calls their 'eat()' and 'sleep()' methods. The output window at the bottom shows the execution results: 'The dog eats bones.', 'The dog sleeps in its kennel.', 'The cat eats fish.', 'The lion sleeps in the jungle.', and a message indicating the program finished with exit code 0.

```
1: interface Animal {
2:     void eat(); // Method to be implemented
3:     void sleep(); // Method to be implemented
4: }
5:
6: class Dog implements Animal {
7:     public void eat() {
8:         System.out.println("The dog eats bones.");
9:     }
10:
11:     public void sleep() {
12:         System.out.println("The dog sleeps in its kennel.");
13:     }
14: }
15:
16: abstract class Cat implements Animal {
17:     public void eat() {
18:         System.out.println("The cat eats fish.");
19:     }
20:     // sleep() method is left unimplemented here
21: }
22:
23: class Lion extends Cat {
24:     public void sleep() {
25:         System.out.println("The lion sleeps in the jungle.");
26:     }
27: }
28:
29: public class InterfaceExample {
30:     public static void main(String args[]) {
31:         Dog dog = new Dog();
32:         dog.eat();
33:         dog.sleep();
34:
35:         Lion lion = new Lion();
36:         lion.eat();
37:         lion.sleep(); // Now it's implemented in Lion class
38:     }
39: }
```

The dog eats bones.
The dog sleeps in its kennel.
The cat eats fish.
The lion sleeps in the jungle.
...Program finished with exit code 0
Press ENTER to exit console.

17. Write a program to design a string class that perform string method (Equal, Reverse the string, change case).

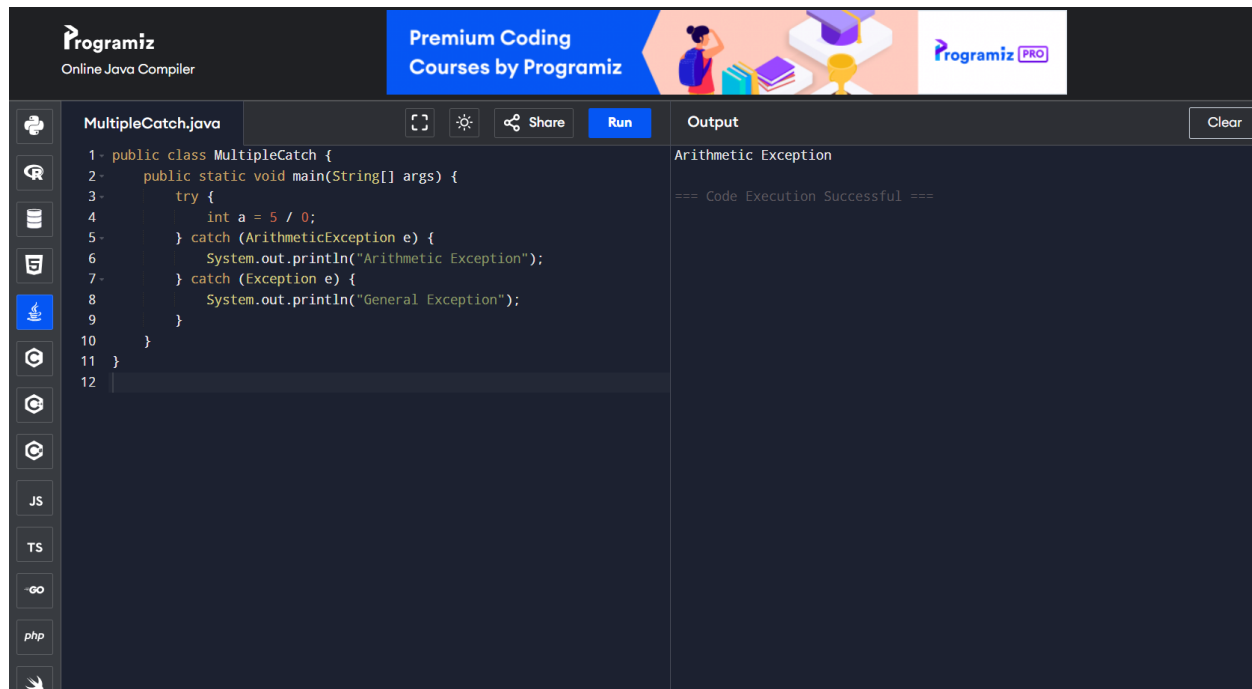


The screenshot shows the Programiz Online Java Compiler interface. The top bar includes the Programiz logo, 'Online Java Compiler', a microphone icon, a video feed, and a 'Run' button. The main editor shows a file named 'MyString.java' with the following code: a 'MyString' class with a 'main' method that uses 'String' methods: 'equals()', 'reverse()', 'toUpperCase()', and 'toLowerCase()'. The output window on the right shows the results: 'Equal: false', 'Reverse: olleH', 'Upper: HELLO', 'Lower: hello', and a success message '=== Code Execution Successful ==='. A 'Clear' button is located next to the output window.

```
1: class MyString {
2:     public static void main(String[] args) {
3:         String str = "Hello";
4:         System.out.println("Equal: " + str.equals("hello"));
5:         System.out.println("Reverse: " + new StringBuilder(str)
6:             .reverse());
7:         System.out.println("Upper: " + str.toUpperCase());
8:         System.out.println("Lower: " + str.toLowerCase());
9:     }
10: }
```

Equal: false
Reverse: olleH
Upper: HELLO
Lower: hello
=== Code Execution Successful ===

18. Write a program to handle the exception using try and multiple catch block.



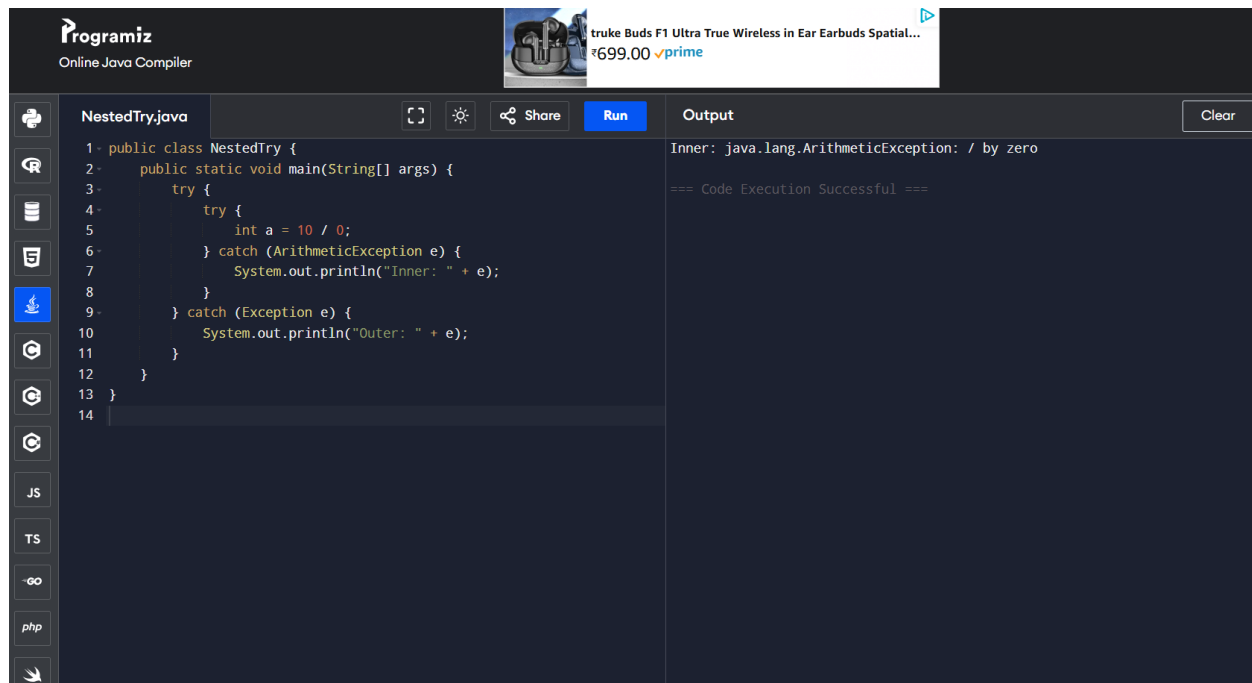
The screenshot shows the Programiz Online Java Compiler interface. The editor contains a Java file named `MultipleCatch.java` with the following code:

```
1- public class MultipleCatch {
2-     public static void main(String[] args) {
3-         try {
4-             int a = 5 / 0;
5-         } catch (ArithmeticException e) {
6-             System.out.println("Arithmetic Exception");
7-         } catch (Exception e) {
8-             System.out.println("General Exception");
9-         }
10-    }
11- }
12-
```

The output window on the right shows the result of the execution:

```
Arithmetic Exception
=== Code Execution Successful ===
```

19. Write a program that implement the Nested Try Statements.



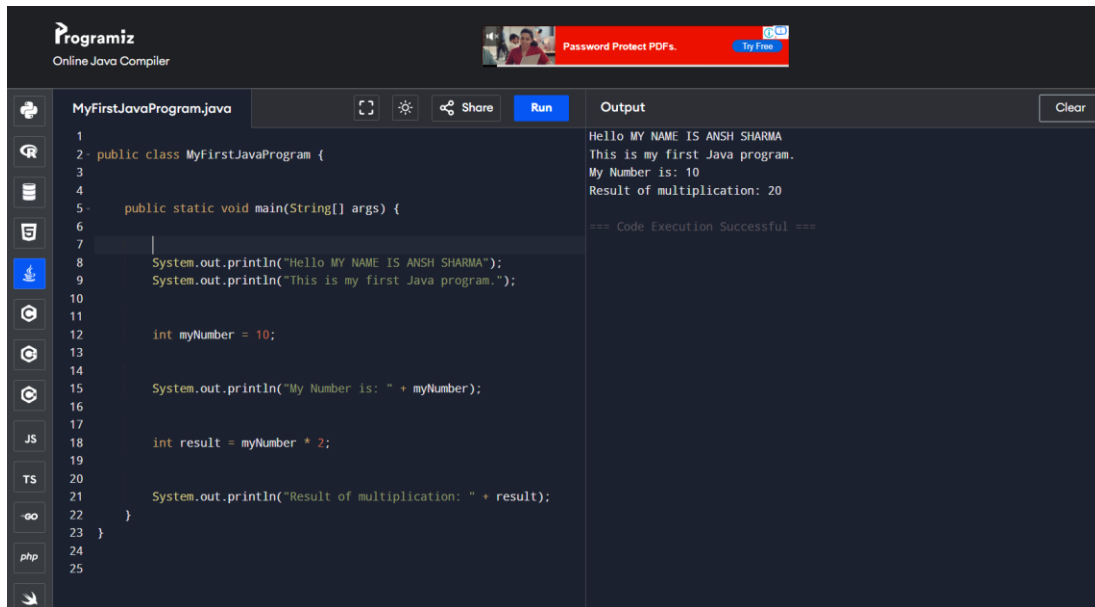
The screenshot shows the Programiz Online Java Compiler interface. The editor contains a Java file named `NestedTry.java` with the following code:

```
1- public class NestedTry {
2-     public static void main(String[] args) {
3-         try {
4-             try {
5-                 int a = 10 / 0;
6-             } catch (ArithmeticException e) {
7-                 System.out.println("Inner: " + e);
8-             }
9-         } catch (Exception e) {
10-             System.out.println("Outer: " + e);
11-         }
12-    }
13- }
14-
```

The output window on the right shows the result of the execution:

```
Inner: java.lang.ArithmeticException: / by zero
=== Code Execution Successful ===
```

20. Write a program to create a package that access the member of External class as well as



The screenshot shows the Programiz Online Java Compiler interface. The code editor on the left contains a Java program named `MyFirstJavaProgram.java`. The program defines a public class `MyFirstJavaProgram` with a `main` method. Inside the `main` method, it prints "Hello MY NAME IS ANSH SHARMA", "This is my first Java program.", "My Number is: 10", and "Result of multiplication: 20". The output window on the right shows the execution results, which match the printed statements in the code. The output also includes a success message: "=== Code Execution Successful ===".

```
1  
2: public class MyFirstJavaProgram {  
3  
4  
5:     public static void main(String[] args) {  
6  
7  
8         System.out.println("Hello MY NAME IS ANSH SHARMA");  
9         System.out.println("This is my first Java program.");  
10  
11  
12         int myNumber = 10;  
13  
14         System.out.println("My Number is: " + myNumber);  
15  
16  
17         int result = myNumber * 2;  
18  
19  
20  
21         System.out.println("Result of multiplication: " + result);  
22     }  
23 }  
24  
25
```

Output

```
Hello MY NAME IS ANSH SHARMA  
This is my first Java program.  
My Number is: 10  
Result of multiplication: 20  
  
=== Code Execution Successful ===
```

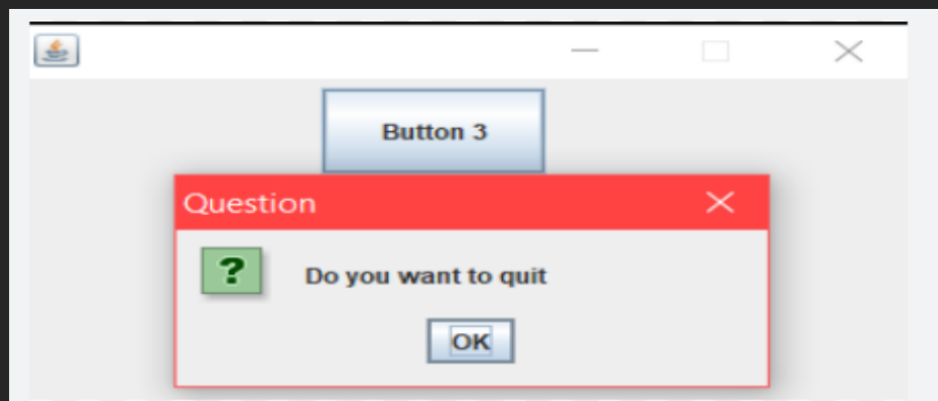
21. Write a program that import the user define package and access the Member variable of classes that contained by package

```
import javax.swing.*; import java.awt.event.*;

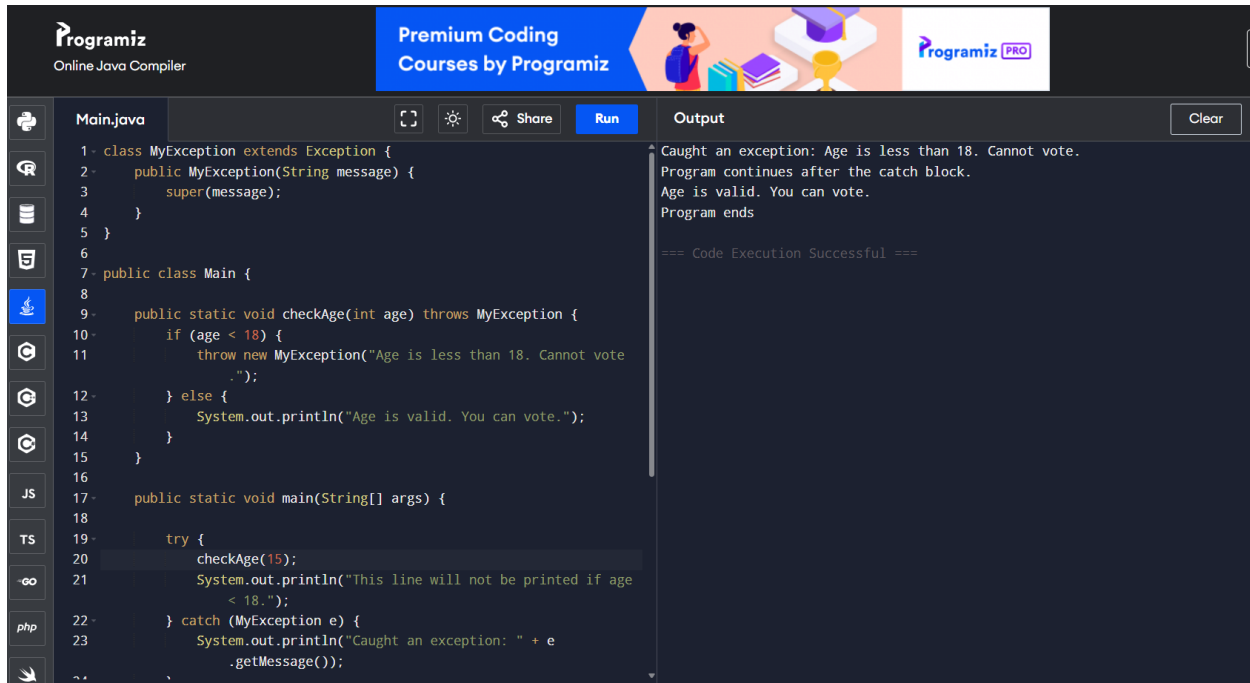
public class DialogBoxExample { public static void main(String[] args) { // Create frame
JFrame frame = new JFrame("Dialog Demo"); frame.setSize(300, 200);
frame.setLayout(null);

    // Create button
    JButton button = new JButton("Button 3");
    button.setBounds(100, 60, 100, 30);
    // Add action listener
    button.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            JOptionPane.showMessageDialog(
                frame,
                "Do you want to quit",
                "Question",
                JOptionPane.QUESTION_MESSAGE
            );
        }
    });
    // Add button and frame settings
    frame.add(button);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setVisible(true);
}}
```

Result/Output:



22. Write a program to handle the user define exception using throw keyword.



Programiz
Online Java Compiler

Premium Coding Courses by Programiz

Programiz PRO

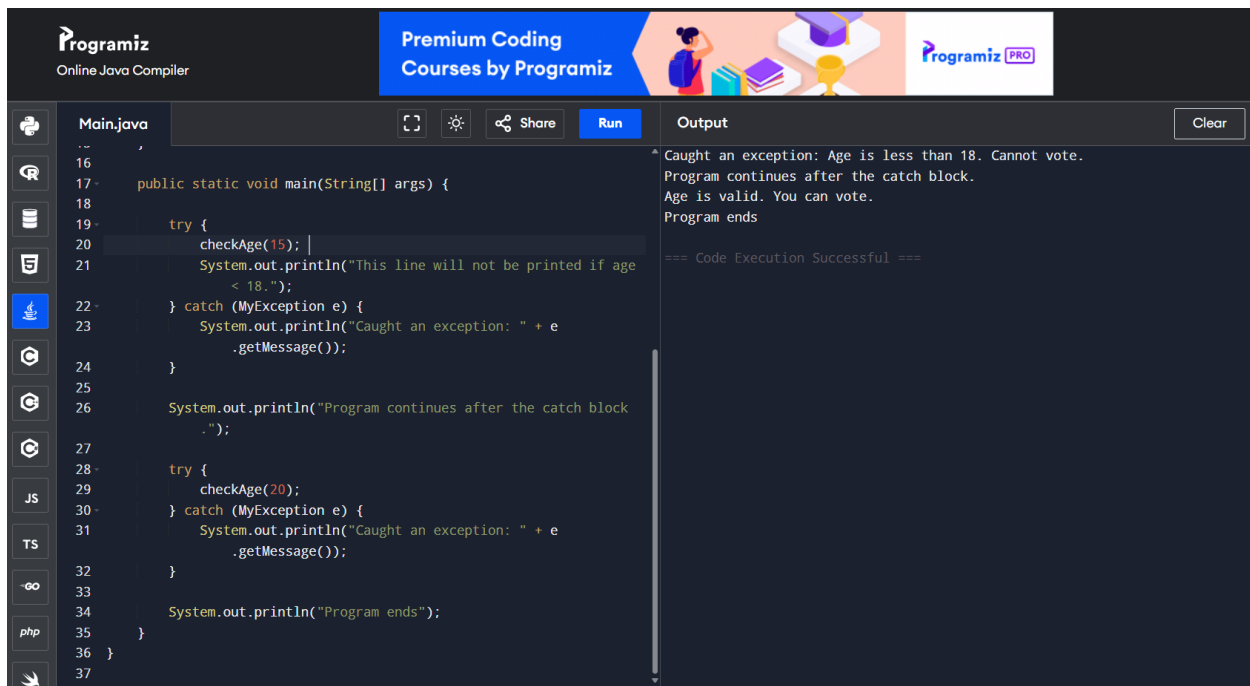
Main.java

```
1 class MyException extends Exception {
2     public MyException(String message) {
3         super(message);
4     }
5 }
6
7 public class Main {
8
9     public static void checkAge(int age) throws MyException {
10        if (age < 18) {
11            throw new MyException("Age is less than 18. Cannot vote.");
12        } else {
13            System.out.println("Age is valid. You can vote.");
14        }
15    }
16
17    public static void main(String[] args) {
18
19        try {
20            checkAge(15);
21            System.out.println("This line will not be printed if age < 18.");
22        } catch (MyException e) {
23            System.out.println("Caught an exception: " + e.getMessage());
24        }
25    }
26 }
```

Output

```
Caught an exception: Age is less than 18. Cannot vote.
Program continues after the catch block.
Age is valid. You can vote.
Program ends

=== Code Execution Successful ===
```



Programiz
Online Java Compiler

Premium Coding Courses by Programiz

Programiz PRO

Main.java

```
16
17 public static void main(String[] args) {
18
19     try {
20         checkAge(15);
21         System.out.println("This line will not be printed if age < 18.");
22     } catch (MyException e) {
23         System.out.println("Caught an exception: " + e.getMessage());
24     }
25
26     System.out.println("Program continues after the catch block.");
27
28     try {
29         checkAge(20);
30     } catch (MyException e) {
31         System.out.println("Caught an exception: " + e.getMessage());
32     }
33
34     System.out.println("Program ends");
35 }
36
37 }
```

Output

```
Caught an exception: Age is less than 18. Cannot vote.
Program continues after the catch block.
Age is valid. You can vote.
Program ends

=== Code Execution Successful ===
```

23. Write a program to create a class component that show control and event handling on that control .

Experiment No: 23

```
import java.awt.*; import java.awt.event.*; import javax.swing.*;

public class CombinedLayoutExample extends JFrame implements ActionListener
{ CardLayout cardLayout; JPanel cardPanel;

public CombinedLayoutExample() {
    setTitle("CardLayout + GridLayout Example");
    setSize(400, 300);
    setDefaultCloseOperation(EXIT_ON_CLOSE);
    setLayout(new BorderLayout());
    // Top buttons to switch cards
    JPanel buttonPanel = new JPanel();
    JButton btnAlphabet = new JButton("Alphabets");
    JButton btnNumbers = new JButton("Numbers");
    buttonPanel.add(btnAlphabet);
    buttonPanel.add(btnNumbers);
    // Card panel with CardLayout
    cardLayout = new CardLayout();
    cardPanel = new JPanel(cardLayout);
    // Alphabet card using GridLayout (3x1)
    JPanel alphabetPanel = new JPanel(new GridLayout(1, 3, 10,
10));
    alphabetPanel.add(new JButton("A"));
    alphabetPanel.add(new JButton("B"));
    alphabetPanel.add(new JButton("C"));
    // Number card using GridLayout (3x3)
    JPanel numberPanel = new JPanel(new GridLayout(3, 3, 10, 10));
    for (int i = 1; i <= 9; i++) {
        numberPanel.add(new JButton(String.valueOf(i)));
    }
    // Add both cards
    cardPanel.add(alphabetPanel, "Alphabets");
    cardPanel.add(numberPanel, "Numbers");
    // Add action listeners
    btnAlphabet.addActionListener(this);
    btnNumbers.addActionListener(this);
    // Add panels to frame
    add(buttonPanel, BorderLayout.NORTH);
    add(cardPanel, BorderLayout.CENTER);
    setVisible(true);
}

public void actionPerformed(ActionEvent e) {
    cardLayout.show(cardPanel, e.getActionCommand());
}

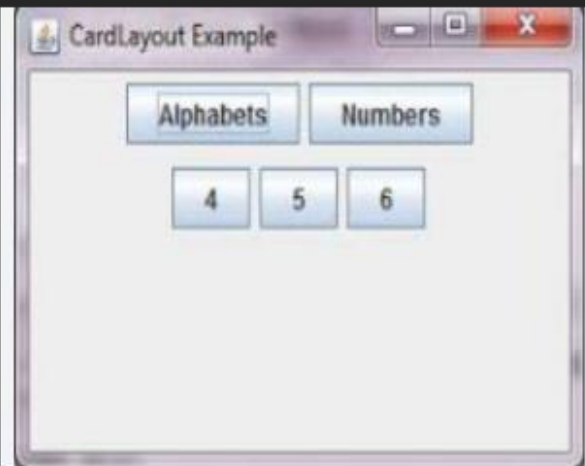
public static void main(String[] args) {
```

Result/Output:

Applet Viewer: GridLayoutDemo

Applet

1	2	3
4	5	6
7	8	9



Experiment No: 24

```
import javax.swing.*; import javax.swing.table.DefaultTableModel; import java.awt.*;
import java.awt.event.*;

public class StudentRecordForm extends JFrame implements ActionListener { // Form
components JTextField rollNoField, nameField, classField, sectionField; JTextArea
addressArea; JTable table; DefaultTableModel model; JButton insertBtn, viewBtn,
clearBtn, exitBtn;

public StudentRecordForm() {
    setTitle("School Record Management System");
    setSize(700, 500);
    setLayout(null);
    setDefaultCloseOperation(EXIT_ON_CLOSE);
    // Title label
    JLabel title = new JLabel("School Record Management System",
JLabel.CENTER);
    title.setFont(new Font("Serif", Font.BOLD | Font.ITALIC, 20));
    title.setBounds(150, 10, 400, 30);
    add(title);
    // Labels and text fields
    JLabel rollLabel = new JLabel("Roll No:");
    rollLabel.setBounds(50, 60, 100, 25);
    add(rollLabel);
    rollNoField = new JTextField();
    rollNoField.setBounds(150, 60, 100, 25);
    add(rollNoField);
    JLabel classLabel = new JLabel("Class:");
    classLabel.setBounds(400, 60, 50, 25);
    add(classLabel);
    classField = new JTextField();
    classField.setBounds(460, 60, 50, 25);
    add(classField);
    JLabel nameLabel = new JLabel("Name:");
    nameLabel.setBounds(50, 100, 100, 25);
    add(nameLabel);
    nameField = new JTextField();
    nameField.setBounds(150, 100, 200, 25);
    add(nameField);
    JLabel sectionLabel = new JLabel("Section:");
    sectionLabel.setBounds(400, 100, 60, 25);
    add(sectionLabel);
    sectionField = new JTextField();
    sectionField.setBounds(460, 100, 50, 25);
    add(sectionField);
    JLabel addressLabel = new JLabel("Address:");
    addressLabel.setBounds(50, 140, 100, 25);
```



```

        add(addressLabel);
        addressArea = new JTextArea();
        JScrollPane scrollPane = new JScrollPane(addressArea);
        scrollPane.setBounds(150, 140, 400, 60);
        add(scrollPane);
        // Table
        model = new DefaultTableModel(new String[]{"Roll No.", "Name",
"Class", "Section", "Address"}, 0);
        table = new JTable(model);
        JScrollPane tablePane = new JScrollPane(table);
        tablePane.setBounds(50, 220, 600, 100);
        add(tablePane);
        // Buttons
        insertBtn = new JButton("Insert");
        insertBtn.setBounds(50, 340, 100, 30);
        insertBtn.addActionListener(this);
        add(insertBtn);
        viewBtn = new JButton("View data");
        viewBtn.setBounds(180, 340, 100, 30);
        viewBtn.addActionListener(this);
        add(viewBtn);
        clearBtn = new JButton("Clear");
        clearBtn.setBounds(310, 340, 100, 30);
        clearBtn.addActionListener(this);
        add(clearBtn);
        exitBtn = new JButton("Exit");
        exitBtn.setBounds(440, 340, 100, 30);
        exitBtn.addActionListener(this);
        add(exitBtn);
        setVisible(true);
    }
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == insertBtn) {
            String roll = rollNoField.getText();
            String name = nameField.getText();
            String cls = classField.getText();
            String section = sectionField.getText();
            String address = addressArea.getText();
            if (!roll.isEmpty() && !name.isEmpty()) {
                model.addRow(new Object[]{roll, name, cls, section,
address});
                JOptionPane.showMessageDialog(this, "Record Added
Successfully", "Message", JOptionPane.INFORMATION_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(this, "Please fill in
Roll No and Name", "Warning", JOptionPane.WARNING_MESSAGE);
            }
        } else if (e.getSource() == clearBtn) {
            rollNoField.setText("");

```

```

        nameField.setText("");
        classField.setText("");
        sectionField.setText("");
        addressArea.setText("");
    } else if (e.getSource() == exitBtn) {
        System.exit(0);
    } else if (e.getSource() == viewBtn) {
        // View logic already handled by JTable model
        JOptionPane.showMessageDialog(this, "Table updated with
current records.", "Info", JOptionPane.INFORMATION_MESSAGE);
    }
    public static void main(String[] args) {
        new StudentRecordForm();
    }
}

```

Result/Output:

The screenshot displays the 'School Record Management System' window. It features input fields for 'Roll No.' (1), 'Class' (5), 'Name' (Suga), 'Section' (A), and 'Address' (Seoul, Korea). Below these fields is a table with columns 'Roll No.', 'Name', and 'Address'. At the bottom are buttons for 'Insert', 'View data', 'Clear', and 'Exit'. A 'Message' dialog box is overlaid on the table, displaying an information icon and the text 'Record Added Successfully' with an 'OK' button.