

1) Create a class called Invoice that a hardware store might use to represent an invoice for an item sold at the store. An Invoice should include four pieces of information as instance variables—a part number (type String), a part description (type String), a quantity of the item being purchased (type int) and a price per item (double). Your class should have a constructor that initializes the four instance variables. In addition, provide a method named `getInvoiceAmount` that calculates the invoice amount.

=> Program -

```
class Invoice {
    String partNumber;
    String partDescription;
    int quantity;
    double pricePerItem;
    public Invoice(String partNumber, String partDescription, int quantity, double pricePerItem)
    {
        this.partNumber = partNumber;
        this.partDescription = partDescription;
        if (quantity < 0) {
            this.quantity = 0;
        } else {
            this.quantity = quantity;
        }
        if (pricePerItem < 0) {
            this.pricePerItem = 0.0;
        } else {
            this.pricePerItem = pricePerItem;
        }
    }
    public double getInvoiceAmount() {
        return quantity * pricePerItem;
    }
}

public class MainInvoice {
    public static void main(String s[]) {
        Invoice invoice1 = new Invoice("1234", "Sofa", 4, 10050);
        System.out.println("Part Number: " + invoice1.partNumber);
        System.out.println("Part Description: " + invoice1.partDescription);
        System.out.println("Quantity: " + invoice1.quantity);
        System.out.println("Price per Item: " + invoice1.pricePerItem);
        System.out.println("Total Invoice Amount: " + invoice1.getInvoiceAmount());
    }
}
```

=> Output -

```

abstract class Bank {
    abstract int getBalance();
}
class BankA extends Bank {
    int getBalance() {
        return 100;
    }
}
class BankB extends Bank {
    int getBalance() {
        return 150;
    }
}
class BankC extends Bank {
    int getBalance() {
        return 200;
    }
}
public class Main {
    public static void main(String s[]) {
        Bank bankA = new BankA();
    }
}

```

```

java -cp /tmp/yOTtcIqfPd/Main
Balance in Bank A: $100
Balance in Bank B: $150
Balance in Bank C: $200

=== Code Execution Successful ===

```

2) Develop a Java application to generate Electricity bill. Create a class with the following members: Consumer no., consumer name, previous month reading, current month reading, and type of EB connection (i.e. domestic or commercial). Compute the bill amount using the user defined tariff.

=> Program -

```
import java.util.*;
```

```

class ElectricityBill {
    String consumerNo;
    String consumerName;
    double previousReading;
    double currentReading;
    String connectionType;
    public ElectricityBill(String consumerNo, String consumerName, double previousReading,
double currentReading, String connectionType) {
        this.consumerNo = consumerNo;
        this.consumerName = consumerName;
        this.previousReading = previousReading;
        this.currentReading = currentReading;
        this.connectionType = connectionType;
    }
    public double calculateBill() {
        if (currentReading < previousReading) {
            System.out.println("Error: Current reading cannot be less than previous reading.");
            return -1;
        }
        double unitsConsumed = currentReading - previousReading;
        double rate = connectionType.equalsIgnoreCase("domestic") ? 5 : 10;
        return unitsConsumed * rate;
    }
}

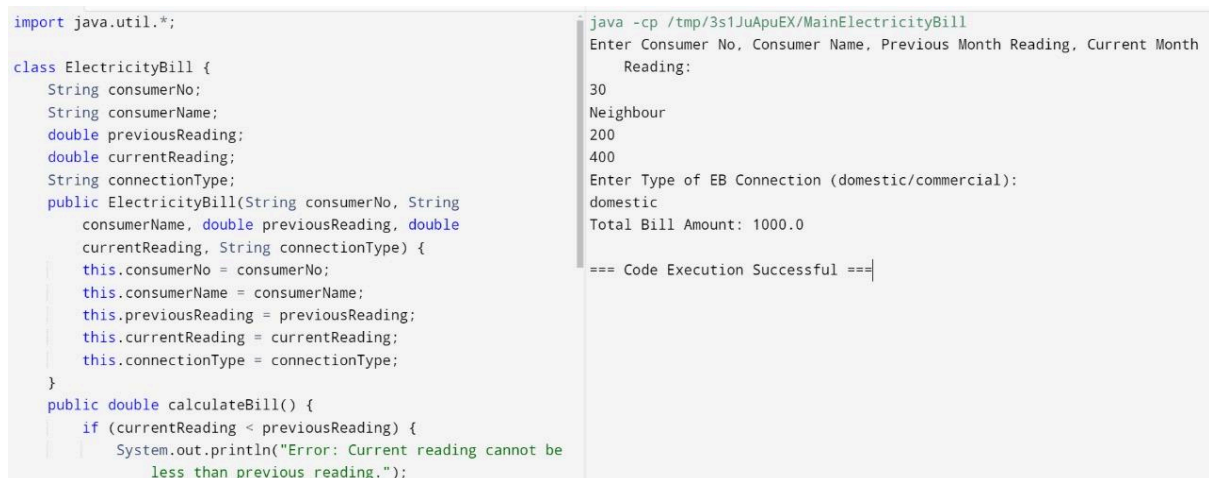
```

```

public class MainElectricityBill {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter Consumer No, Consumer Name, Previous Month Reading,
Current Month Reading:");
        String consumerNo = sc.nextLine();
        String consumerName = sc.nextLine();
        double previousReading = sc.nextDouble();
        double currentReading = sc.nextDouble();
        System.out.println("Enter Type of EB Connection (domestic/commercial):");
        String connectionType = sc.next();
        ElectricityBill bill = new ElectricityBill(consumerNo, consumerName, previousReading,
currentReading, connectionType);
        double billAmount = bill.calculateBill();
        if (billAmount >= 0) {
            System.out.println("Total Bill Amount: " + billAmount);
        }
    }
}

```

=> Output -



The screenshot shows a code editor with two panels. The left panel contains the source code for the `ElectricityBill` class. The right panel shows the output of running the program, which prompts the user for input and displays the calculated bill amount.

```

import java.util.*;

class ElectricityBill {
    String consumerNo;
    String consumerName;
    double previousReading;
    double currentReading;
    String connectionType;
    public ElectricityBill(String consumerNo, String
consumerName, double previousReading, double
currentReading, String connectionType) {
        this.consumerNo = consumerNo;
        this.consumerName = consumerName;
        this.previousReading = previousReading;
        this.currentReading = currentReading;
        this.connectionType = connectionType;
    }
    public double calculateBill() {
        if (currentReading < previousReading) {
            System.out.println("Error: Current reading cannot be
less than previous reading.");
        }
    }
}

```

```

java -cp /tmp/3s1JuApuEX/MainElectricityBill
Enter Consumer No, Consumer Name, Previous Month Reading, Current Month
Reading:
30
Neighbour
200
400
Enter Type of EB Connection (domestic/commercial):
domestic
Total Bill Amount: 1000.0

=== Code Execution Successful ===

```

3) Create class `SavingsAccount`. Use a static variable `annualInterestRate` to store the annual interest rate for all account holders. Each object of the class contains a private instance variable `savingsBalance` indicating the amount the saver currently has on deposit. Provide method `calculateMonthlyInterest` to calculate the monthly interest by multiplying the `savingsBalance` by `annualInterestRate` divided by 12 this interest should be added to `savingsBalance`. Provide a static method `modifyInterestRate` that sets the `annualInterestRate` to a new value.

=> Program -

```

class SavingsAccount {

```

```

private double savingsBalance;
private static double annualInterestRate;
public SavingsAccount(double initialBalance) {
    if (initialBalance > 0) {
        savingsBalance = initialBalance;
    } else {
        savingsBalance = 0;
    }
}
public void calculateMonthlyInterest() {
    double monthlyInterest = (savingsBalance * annualInterestRate) / 12;
    savingsBalance += monthlyInterest;
}
public static void modifyInterestRate(double newRate) {
    annualInterestRate = newRate;
}
public double getSavingsBalance() {
    return savingsBalance;
}
}

public class MainSavingsAccount {
    public static void main(String s[]) {
        SavingsAccount account1 = new SavingsAccount(1000);
        SavingsAccount.modifyInterestRate(0.04);
        account1.calculateMonthlyInterest();
        System.out.println("New Balance: " + account1.getSavingsBalance());
    }
}

```

=> Output -

The screenshot shows a code editor with two panes. The left pane displays the Java code for the `SavingsAccount` and `MainSavingsAccount` classes. The right pane shows the output of running the code, which is "New Balance: 1003.3333333333334" followed by a success message "=== Code Execution Successful ===".

```

savingsBalance += monthlyInterest;
}
public static void modifyInterestRate(double newRate) {
    annualInterestRate = newRate;
}
public double getSavingsBalance() {
    return savingsBalance;
}
}
public class MainSavingsAccount {
    public static void main(String[] args) {
        SavingsAccount account1 = new SavingsAccount(1000);
        SavingsAccount.modifyInterestRate(0.04);
        account1.calculateMonthlyInterest();
        System.out.println("New Balance: " + account1
            .getSavingsBalance());
    }
}

```

```

java -cp /tmp/20e7peACdK/MainSavingsAccount
New Balance: 1003.3333333333334
=== Code Execution Successful ===

```

4) Create a class called `Book` to represent a book. A `Book` should include four pieces of information as instance variables- a book name, an ISBN number, an author name and a publisher. Your class should have a constructor that initializes the four instance variables.

Provide a set method and get method for each instance variable. In addition, provide a method named `getBookInfo` that returns the description of the book as a `String`.

=> Program -

```
class Book {
    private String name;
    private String isbn;
    private String author;
    private String publisher;
    public Book(String name, String isbn, String author, String publisher) {
        this.name = name;
        this.isbn = isbn;
        this.author = author;
        this.publisher = publisher;
    }
    public String getBookInfo() {
        return "The name of the book is "+ name+". Its ISBN is " + isbn+". It is written by "
+author+" and published by "+publisher+".";
    }
}

public class MainBook {
    public static void main(String[] args) {
        Book book = new Book("VERY NICE", "17", "SEVENTEEN", "Pledis");
        System.out.println(book.getBookInfo());
    }
}
```

=> Output -



The screenshot displays a code editor with two panels. The left panel contains the Java source code for the `Book` and `MainBook` classes, which is identical to the code provided in the previous block. The right panel shows the output of running the program, which is the string returned by the `getBookInfo` method: "The name of the book is VERY NICE. Its ISBN is 17. It is written by SEVENTEEN and published by Pledis." Below the output, a status message reads "=== Code Execution Successful ===".

```
class Book {
    private String name;
    private String isbn;
    private String author;
    private String publisher;
    public Book(String name, String isbn, String author, String
publisher) {
        this.name = name;
        this.isbn = isbn;
        this.author = author;
        this.publisher = publisher;
    }
    public String getBookInfo() {
        return "The name of the book is "+ name+". Its ISBN is "
+ isbn+". It is written by " +author+" and published
by "+publisher+".";
    }
}

public class MainBook {
    public static void main(String[] args) {
        Book book = new Book("VERY NICE", "17", "SEVENTEEN",
"Pledis");
    }
}
```

```
java -cp /tmp/3kxG2Ueai4/MainBook
The name of the book is VERY NICE. Its ISBN is 17. It is written by
SEVENTEEN and published by Pledis.

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

5) Write a Java program to print all numbers below 100,000 that are both prime and Fibonacci number (some examples are 2, 3, 5, 13, etc.).

=> Program -

```
import java.util.*;
public class PrimeFibonacci {
    public static void main(String s[]) {
        for (int i = 1; i < 100000; i++) {
            if (isPrime(i) && isFibonacci(i)) {
                System.out.println(i);
            }
        }
    }
    private static boolean isPrime(int n) {
        if (n <= 1) return false;
        for (int i = 2; i * i <= n; i++) {
            if (n % i == 0) return false;
        }
        return true;
    }
    private static boolean isFibonacci(int n) {
        int a = 0, b = 1;
        while (b < n) {
            int temp = b;
            b += a;
            a = temp;
        }
        return b == n || n == 0;
    }
}
```

=> Output -

```
import java.util.*;
public class PrimeFibonacci {
    public static void main(String s[]) {
        for (int i = 1; i < 100000; i++) {
            if (isPrime(i) && isFibonacci(i)) {
                System.out.println(i);
            }
        }
    }
    private static boolean isPrime(int n) {
        if (n <= 1) return false;
        for (int i = 2; i * i <= n; i++) {
            if (n % i == 0) return false;
        }
        return true;
    }
    private static boolean isFibonacci(int n) {
        int a = 0, b = 1;
        while (b < n) {
            int temp = b;
            b += a;
            a = temp;
        }
        return b == n || n == 0;
    }
}
```

```
java -cp /tmp/7i0j6wUbpI/PrimeFibonacci
2
3
5
13
89
233
1597
28657

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

6) Write a java program to find whether given sentence is Pangram or not. If it is pangram, then print 1 else print -1. A pangram is a sentence containing every letter in the English Alphabet. Ex: The quick brown fox jumps over the lazy Dog.

=> Program -

```
import java.util.*;
```

```
public class PangramChecker {
    public static void main(String s[]) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter a sentence:");
        String sentence = sc.nextLine();
        if (isPangram(sentence)) {
            System.out.println(1);
        } else {
            System.out.println(-1);
        }
    }
    private static boolean isPangram(String str) {
        for (char c = 'a'; c <= 'z'; c++) {
            if (!str.toLowerCase().contains(String.valueOf(c))) {
                return false;
            }
        }
        return true;
    }
}
```

=> Output -

<pre>import java.util.*; public class PangramChecker { public static void main(String s[]) { Scanner sc = new Scanner(System.in); System.out.println("Enter a sentence:"); String sentence = sc.nextLine(); if (isPangram(sentence)) { System.out.println(1); } else { System.out.println(-1); } } private static boolean isPangram(String str) { for (char c = 'a'; c <= 'z'; c++) { if (!str.toLowerCase().contains(String.valueOf(c))) { return false; } } return true; } }</pre>	<pre>java -cp /tmp/zkn1NJlBor/PangramChecker Enter a sentence: The quick brown fox jumps over the lazy Dog. 1 === Code Execution Successful ===</pre>
--	---

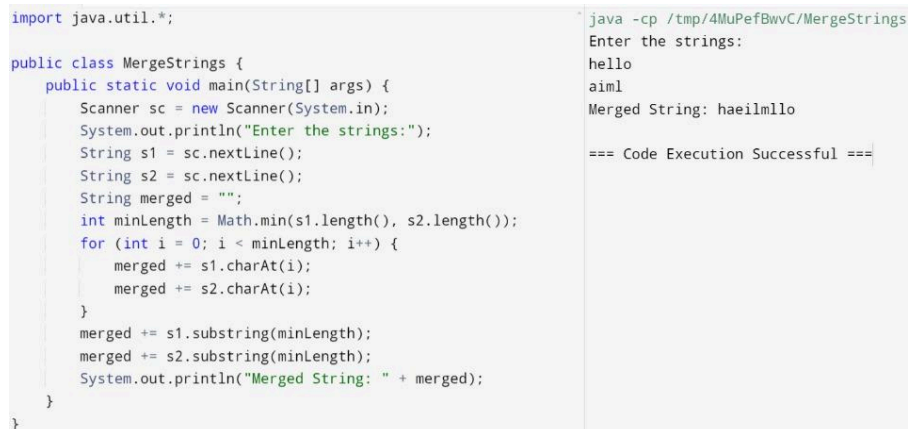
7) Given two strings s1 and s2, your task is to merge those strings to form a new merged string. A merge operation on two strings is described as follows: Append alternating characters from s1 and s2, respectively, to merged String. Once all of the characters in one of the strings have been merged, append the remaining characters in the other string to merged String.

=> Program -

```
import java.util.*;

public class MergeStrings {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the strings:");
        String s1 = sc.nextLine();
        String s2 = sc.nextLine();
        String merged = "";
        int minLength = Math.min(s1.length(), s2.length());
        for (int i = 0; i < minLength; i++) {
            merged += s1.charAt(i);
            merged += s2.charAt(i);
        }
        merged += s1.substring(minLength);
        merged += s2.substring(minLength);
        System.out.println("Merged String: " + merged);
    }
}
```

=> Output -

The screenshot shows a code editor with two panes. The left pane contains the Java source code for the MergeStrings class, which takes two strings as input and merges them character by character up to the minimum length, then appends the remaining characters. The right pane shows the output of running the program: 'Enter the strings:', 'hello', 'aiml', and 'Merged String: haeilmlllo'. Below the output, it says '=== Code Execution Successful ==='.

```
import java.util.*;

public class MergeStrings {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the strings:");
        String s1 = sc.nextLine();
        String s2 = sc.nextLine();
        String merged = "";
        int minLength = Math.min(s1.length(), s2.length());
        for (int i = 0; i < minLength; i++) {
            merged += s1.charAt(i);
            merged += s2.charAt(i);
        }
        merged += s1.substring(minLength);
        merged += s2.substring(minLength);
        System.out.println("Merged String: " + merged);
    }
}
```

```
java -cp /tmp/4MuPefBwvC/MergeStrings
Enter the strings:
hello
aiml
Merged String: haeilmlllo

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

8) Write a Java program to create an abstract class named Shape that contains two integers and an empty method named print Area (). Provide three classes named Rectangle, Triangle, and Circle such that each one of the classes extends the class Shape. Each one of the classes contains only the method print Area () that prints the area of the given shape.

=> Program -

```
abstract class Shape {
    int length;
    int width;
    abstract void printArea();
}
```



```

}
class Rectangle extends Shape {
    Rectangle(int length, int width) {
        this.length = length;
        this.width = width;
    }
    void printArea() {
        System.out.println("Area of Rectangle: " + (length * width));
    }
}
class Triangle extends Shape {
    Triangle(int base, int height) {
        this.length = base;
        this.width = height;
    }
    void printArea() {
        System.out.println("Area of Triangle: " + (0.5 * length * width));
    }
}
class Circle extends Shape {
    Circle(int radius) {
        this.length = radius;
    }
    void printArea() {
        System.out.println("Area of Circle: " + (Math.PI * length * length));
    }
}
public class shapes{
    public static void main(String[] args) {
        Shape rectangle = new Rectangle(5, 8);
        rectangle.printArea();
        Shape triangle = new Triangle(2, 7);
        triangle.printArea();
        Shape circle = new Circle(4);
        circle.printArea();
    }
}

```

=> Output -

```

Circle(int radius) {
    this.length = radius;
}
void printArea() {
    System.out.println("Area of Circle: " + (Math.PI *
        length * length));
}
}
public class shapes{
    public static void main(String[] args) {
        Shape rectangle = new Rectangle(5, 8);
        rectangle.printArea();
        Shape triangle = new Triangle(2, 7);
        triangle.printArea();
        Shape circle = new Circle(4);
        circle.printArea();
    }
}
}

java -cp /tmp/XkLPKyo1XU/shapes
Area of Rectangle: 40
Area of Triangle: 7.0
Area of Circle: 50.26548245743669

=== Code Execution Successful ===

```

9) Create an abstract class 'Bank' with an abstract method 'getBalance'. \$100, \$150 and \$200 are deposited in banks A, B and C respectively. 'BankA', 'BankB' and 'BankC' are subclasses of class 'Bank', each having a method named 'getBalance'.

=> Program -

```

abstract class Bank {
    abstract int getBalance();
}
class BankA extends Bank {
    int getBalance() {
        return 100;
    }
}
class BankB extends Bank {
    int getBalance() {
        return 150;
    }
}
class BankC extends Bank {
    int getBalance() {
        return 200;
    }
}
public class Main {
    public static void main(String s[]) {
        Bank bankA = new BankA();
        System.out.println("Balance in Bank A: $" + bankA.getBalance());
        Bank bankB = new BankB();
        System.out.println("Balance in Bank B: $" + bankB.getBalance());
        Bank bankC = new BankC();
        System.out.println("Balance in Bank C: $" + bankC.getBalance());
    }
}

```

=> Output -

```

abstract class Bank {
    abstract int getBalance();
}
class BankA extends Bank {
    int getBalance() {
        return 100;
    }
}
class BankB extends Bank {
    int getBalance() {
        return 150;
    }
}
class BankC extends Bank {
    int getBalance() {
        return 200;
    }
}
public class Main {
    public static void main(String s[]) {
        Bank bankA = new BankA();
    }
}

```

```

java -cp /tmp/y0TtcIqfPd/Main
Balance in Bank A: $100
Balance in Bank B: $150
Balance in Bank C: $200

=== Code Execution Successful ===

```

10) Develop a java application to inherit currency converter (Dollar to INR, EURO to INR, Yen to INR and vice versa), distance converter (meter to KM, miles to KM and vice versa), time converter (hours to minutes, seconds and vice versa) from a base class Converter.

=> Program -

```

class Converter {
    public double dollarsToINR(double dollars) {
        return dollars * 83.72;
    }
    public double euroToINR(double euros) {
        return euros * 93.53;
    }
    public double yenToINR(double yen) {
        return yen * 0.59;
    }
    public double INRtodollars(double INR) {
        return INR * 0.012;
    }
    public double INRtoeuro(double INR) {
        return INR * 0.011;
    }
    public double INRtoyen(double INR) {
        return INR * 1.70;
    }
    public double mtokm(double m) {
        return m * 0.001;
    }
    public double mitokm(double mi) {
        return mi * 1.60934;
    }
    public double kmtom(double km) {
        return km * 1000;
    }
}

```

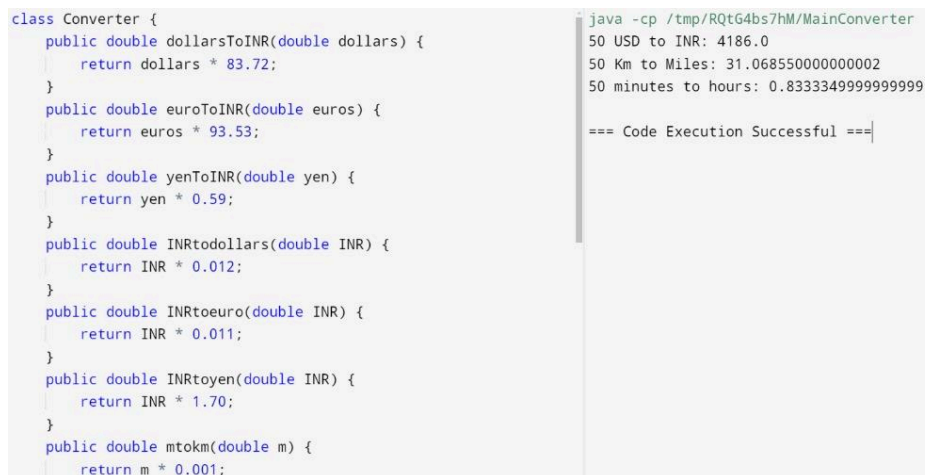
```

    }
    public double kmtomi(double km) {
        return km * 0.621371;
    }
    public double hrstomin(double hrs) {
        return hrs * 60;
    }
    public double hrstosec(double hrs) {
        return hrs * 3600;
    }
    public double mintohrs(double min) {
        return min * 0.0166667;
    }
    public double sectohrs(double sec) {
        return sec * 0.000277778;
    }
}

public class MainConverter {
    public static void main(String s[]) {
        Converter converter = new Converter();
        System.out.println("50 USD to INR: "+converter.dollarsToINR(50));
        System.out.println("50 Km to Miles: " + converter.kmtomi(50));
        System.out.println("50 minutes to hours: " + converter.mintohrs(50));
    }
}

```

=> Output -



```

class Converter {
    public double dollarsToINR(double dollars) {
        return dollars * 83.72;
    }
    public double euroToINR(double euros) {
        return euros * 93.53;
    }
    public double yenToINR(double yen) {
        return yen * 0.59;
    }
    public double INRtodollars(double INR) {
        return INR * 0.012;
    }
    public double INRtoeuro(double INR) {
        return INR * 0.011;
    }
    public double INRtoyen(double INR) {
        return INR * 1.70;
    }
    public double mtkm(double m) {
        return m * 0.001;
    }
}

java -cp /tmp/RQtG4bs7hM/MainConverter
50 USD to INR: 4186.0
50 Km to Miles: 31.068550000000002
50 minutes to hours: 0.8333349999999999

=== Code Execution Successful ===

```

11) Write a Java program that works as a simple calculator. Use a switch statement to accept 2 numbers and an operator for the +, -, *, % operations. Perform arithmetic operation display the result. Handle any possible exceptions like divided by zero for division operation.

=> Program -

```

import java.util.*;

public class IntegerDivision {
    public static void main(String s[]) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the two integers:");
        try {
            int num1 = Integer.parseInt(sc.nextLine());
            int num2 = Integer.parseInt(sc.nextLine());
            int result = num1 / num2;
            System.out.println("Result: " + result);
        } catch (NumberFormatException e) {
            System.out.println("Error: Please enter valid integers.");
        } catch (ArithmeticException e) {
            System.out.println("Error: Division by zero.");
        }
    }
}

```

=> Output -



The screenshot shows a code editor with two panels. The left panel contains the source code for a Java class named SimpleCalculator. The code imports java.util.Scanner, defines a main method that prompts the user for two numbers and an operator, and uses a switch statement to perform addition, subtraction, multiplication, and division. The right panel shows the output of the program when executed. The user entered 5 and 6 for the numbers, and * for the operator. The program calculated the result as 30 and displayed '=== Code Execution Successful ==='.

```

import java.util.Scanner;

public class SimpleCalculator {
    public static void main(String s[]) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the numbers :");
        int num1 = sc.nextInt();
        int num2 = sc.nextInt();
        System.out.println("Enter operator (+, -, *, /):");
        char operator = sc.next().charAt(0);
        switch (operator) {
            case '+':
                System.out.println("result = " + (num1 + num2));
                break;
            case '-':
                System.out.println("result = " + (num1 - num2));
                break;
            case '*':
                System.out.println("result = " + (num1 * num2));
                break;
            case '/':

```

```

java -cp /tmp/sLLz1MlWHH/SimpleCalculator
Enter the numbers :
5
6
Enter operator (+, -, *, /):
*
result = 30
=== Code Execution Successful ===

```

12) Write a Java program that perform integer divisions. The user enters two numbers, Num1 and Num2. The division of Num1 and Num 2 is displayed in the Result. If Num1 or Num2 were not an integer, the program would throw a Number Format Exception. If Num2 were Zero, the program would throw an Arithmetic Exception.

=> Program-

```

import java.util.*;

public class IntegerDivision {
    public static void main(String s[]) {
        Scanner sc = new Scanner(System.in);

```

```

System.out.println("Enter the two integers:");
try {
    int num1 = Integer.parseInt(sc.nextLine());
    int num2 = Integer.parseInt(sc.nextLine());
    int result = num1 / num2;
    System.out.println("Result: " + result);
} catch (NumberFormatException e) {
    System.out.println("Error: Please enter valid integers.");
} catch (ArithmeticException e) {
    System.out.println("Error: Division by zero.");
}
}
}

```

=> Output -

<pre> import java.util.*; public class IntegerDivision { public static void main(String s[]) { Scanner sc = new Scanner(System.in); System.out.println("Enter the two integers:"); try { int num1 = Integer.parseInt(sc.nextLine()); int num2 = Integer.parseInt(sc.nextLine()); int result = num1 / num2; System.out.println("Result: " + result); } catch (NumberFormatException e) { System.out.println("Error: Please enter valid integers."); } catch (ArithmeticException e) { System.out.println("Error: Division by zero."); } } } </pre>	<pre> java -cp /tmp/v9SfKL9iUi/IntegerDivision Enter the two integers: 5 0 ERROR! Error: Division by zero. === Code Execution Successful === </pre>
--	--

<pre> import java.util.*; public class IntegerDivision { public static void main(String s[]) { Scanner sc = new Scanner(System.in); System.out.println("Enter the two integers:"); try { int num1 = Integer.parseInt(sc.nextLine()); int num2 = Integer.parseInt(sc.nextLine()); int result = num1 / num2; System.out.println("Result: " + result); } catch (NumberFormatException e) { System.out.println("Error: Please enter valid integers."); } catch (ArithmeticException e) { System.out.println("Error: Division by zero."); } } } </pre>	<pre> java -cp /tmp/Q8BXVORSSw/IntegerDivision Enter the two integers: h ERROR! Error: Please enter valid integers. === Code Execution Successful === </pre>
--	---