

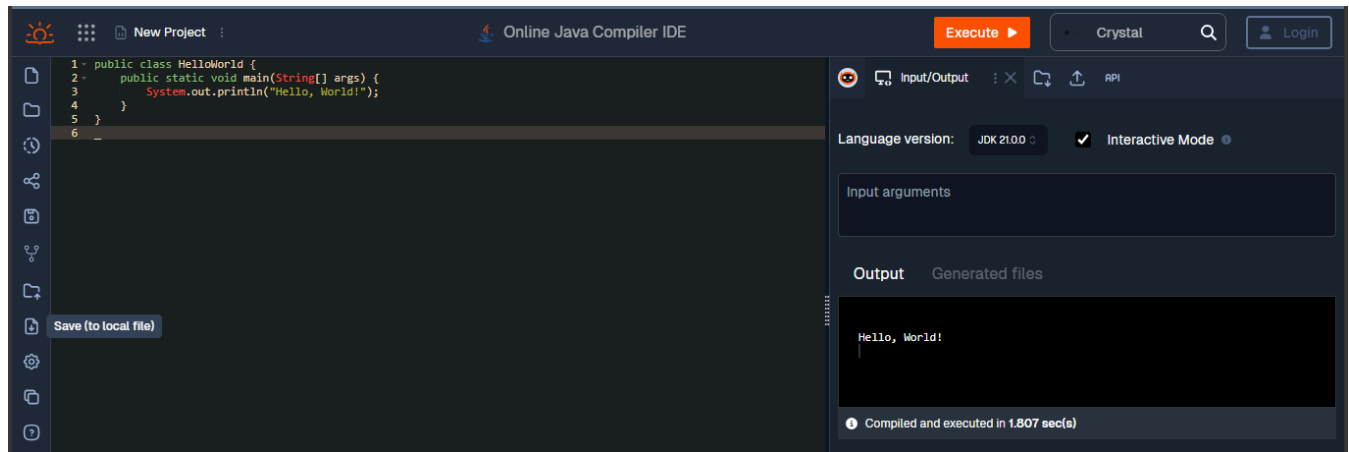
MANAV RACHNA INTERNATIONAL
INSTITUTE OF RESEARCH AND STUDIES



FACULTY OF COMPUTER APPLICATIONS
PRACTICAL FILE
OF
JAVA LAB (OBICA-DS-454)

Submitted By	
Student Name	DISHA THAKRAN
Roll No	23/SCA/BCA(DS & BDA)/012
Programme	Bachelor of Computer Applications
Semester	4th
Section/Group	E
Department	Computer Applications
Batch	2023-26
Submitted To	
Faculty Name	Ms. Priyanka Sharma

Q1. Write java program to print hello world



The screenshot shows the Online Java Compiler IDE interface. The code editor on the left contains the following Java code:

```
1- public class HelloWorld {  
2-     public static void main(String[] args) {  
3-         System.out.println("Hello, World!");  
4-     }  
5- }  
6- }
```

The right sidebar shows the 'Execute' button, 'Crystal' theme, and 'Interactive Mode' checked. The 'Output' tab displays the result: 'Hello, World!'. Below the output, it states 'Compiled and executed in 1.807 sec(s)'.

Q2. Java Program to take input from user and print the sum of two numbers



The screenshot shows the Online Java Compiler IDE interface. The code editor on the left contains the following Java code:

```
1 import java.util.Scanner;  
2  
3 public class SumOfTwoNumbers {  
4     public static void main(String[] args) {  
5         Scanner scanner = new Scanner(System.in);  
6         System.out.print("Enter the first number: ");  
7         int num1 = scanner.nextInt();  
8         System.out.print("Enter the second number: ");  
9         int num2 = scanner.nextInt();  
10  
11         int sum = num1 + num2;  
12  
13         System.out.println("The sum of " + num1 + " and " + num2 + " is: " + sum);  
14         scanner.close();  
15     }  
16 }  
17 }
```

The right sidebar shows the 'Execute' button, 'Moonscript' theme, and 'Interactive Mode' checked. The 'Output' tab displays the result: 'Enter the first number: 54', 'Enter the second number: 69', 'The sum of 54 and 69 is: 123'. Below the output, it states 'Compiled and executed in 11.295 sec(s)'.

Q3. Create a java program to check whether a number entered by user is even or odd



The screenshot shows the Online Java Compiler IDE interface. The code editor on the left contains the following Java code:

```
1 import java.util.Scanner;  
2  
3 public class EvenOddCheck {  
4     public static void main(String[] args) {  
5         Scanner scanner = new Scanner(System.in);  
6         System.out.print("Enter a number: ");  
7         int number = scanner.nextInt();  
8  
9         if (number % 2 == 0) {  
10             System.out.println(number + " is an even number.");  
11         } else {  
12             System.out.println(number + " is an odd number.");  
13         }  
14  
15         scanner.close();  
16     }  
17 }  
18 }
```

The right sidebar shows the 'Execute' button, 'Jelly' theme, and 'Interactive Mode' checked. The 'Output' tab displays the result: 'Enter a number: 88', '88 is an even number.'. Below the output, it states 'Compiled and executed in 4.812 sec(s)'.

Q4. Create a java program to print the average and sum of 5 numbers entered by user.

The screenshot shows an online Java compiler interface. The code editor on the left contains a Java program that prompts the user to enter five numbers, calculates their sum and average, and prints the results. The output panel on the right shows the program's execution with the following input and output:

```
1 import java.util.Scanner;
2
3 public class SumAndAverage {
4     public static void main(String[] args) {
5         Scanner scanner = new Scanner(System.in);
6         int sum = 0;
7         int count = 5;
8
9         System.out.println("Enter five numbers:");
10
11         for (int i = 0; i < count; i++) {
12             System.out.print("Number " + (i + 1) + ": ");
13             int number = scanner.nextInt();
14             sum += number;
15         }
16
17         double average = (double) sum / count;
18
19         System.out.println("The sum of the numbers is: " + sum);
20         System.out.println("The average of the numbers is: " + average);
21
22         scanner.close();
23     }
24 }
25
```

Output:

```
Enter five numbers:
Number 1: 4
Number 2: 5
Number 3: 6
Number 4: 9
Number 5: 3
The sum of the numbers is: 27
The average of the numbers is: 5.4
```

Compiled and executed in 63.372 sec(s)

Q5. Program to calculate the factorial of a number

The screenshot shows an online Java compiler interface. The code editor on the left contains a Java program that prompts the user to enter a number, calculates its factorial, and prints the result. The output panel on the right shows the program's execution with the following input and output:

```
1 import java.util.Scanner;
2
3 public class Factorial {
4     public static void main(String[] args) {
5         Scanner scanner = new Scanner(System.in);
6         System.out.print("Enter a number: ");
7         int number = scanner.nextInt();
8         long factorial = 1;
9
10        for (int i = 1; i <= number; i++) {
11            factorial *= i;
12        }
13
14        System.out.println("The factorial of " + number + " is: " + factorial);
15        scanner.close();
16    }
17 }
18
```

Output:

```
Enter a number: 55
The factorial of 55 is: 671148934688881664
```

Compiled and executed in 7.357 sec(s)

Q6. Program to print Fibonacci series up to n terms.

The screenshot shows an online Java compiler interface. The code editor on the left contains a Java program that prompts the user to enter the number of terms, generates the Fibonacci series, and prints the result. The output panel on the right shows the program's execution with the following input and output:

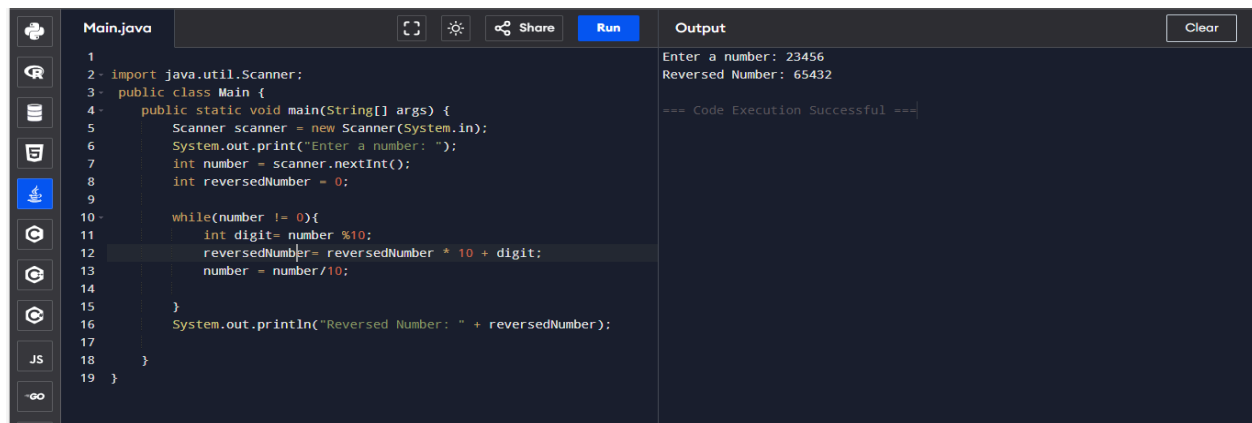
```
1 import java.util.Scanner;
2
3 public class FibonacciSeries {
4     public static void main(String[] args) {
5         Scanner scanner = new Scanner(System.in);
6         System.out.print("Enter the number of terms: ");
7         int n = scanner.nextInt();
8
9         int firstTerm = 0, secondTerm = 1;
10        System.out.print("Fibonacci Series: " + firstTerm + ", " + secondTerm);
11
12        for (int i = 3; i <= n; i++) {
13            int nextTerm = firstTerm + secondTerm;
14            System.out.print(", " + nextTerm);
15            firstTerm = secondTerm;
16            secondTerm = nextTerm;
17        }
18
19        scanner.close();
20    }
21 }
22
```

Output:

```
Enter the number of terms: 58
Fibonacci Series: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377,
```

Compiled and executed in 16.353 sec(s)

Q7.Program to reverse a number



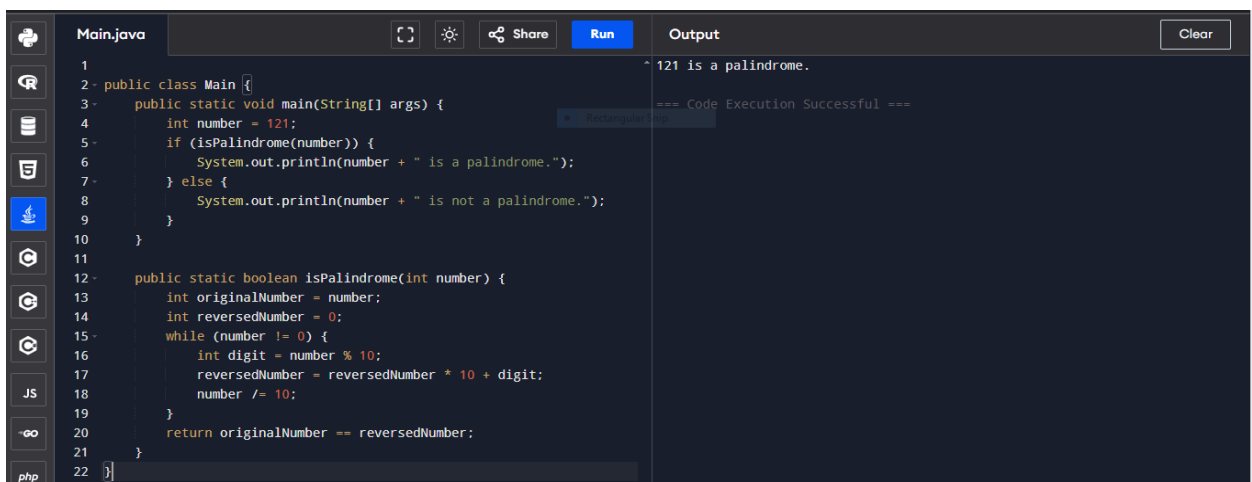
```
1
2- import java.util.Scanner;
3- public class Main {
4-     public static void main(String[] args) {
5-         Scanner scanner = new Scanner(System.in);
6-         System.out.print("Enter a number: ");
7-         int number = scanner.nextInt();
8-         int reversedNumber = 0;
9-
10-        while(number != 0){
11-            int digit= number %10;
12-            reversedNumber= reversedNumber * 10 + digit;
13-            number = number/10;
14-        }
15-        System.out.println("Reversed Number: " + reversedNumber);
16-    }
17- }
18
19 }
```

Output

```
Enter a number: 23456
Reversed Number: 65432

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

Q8.Program to check if a number is a palindrome



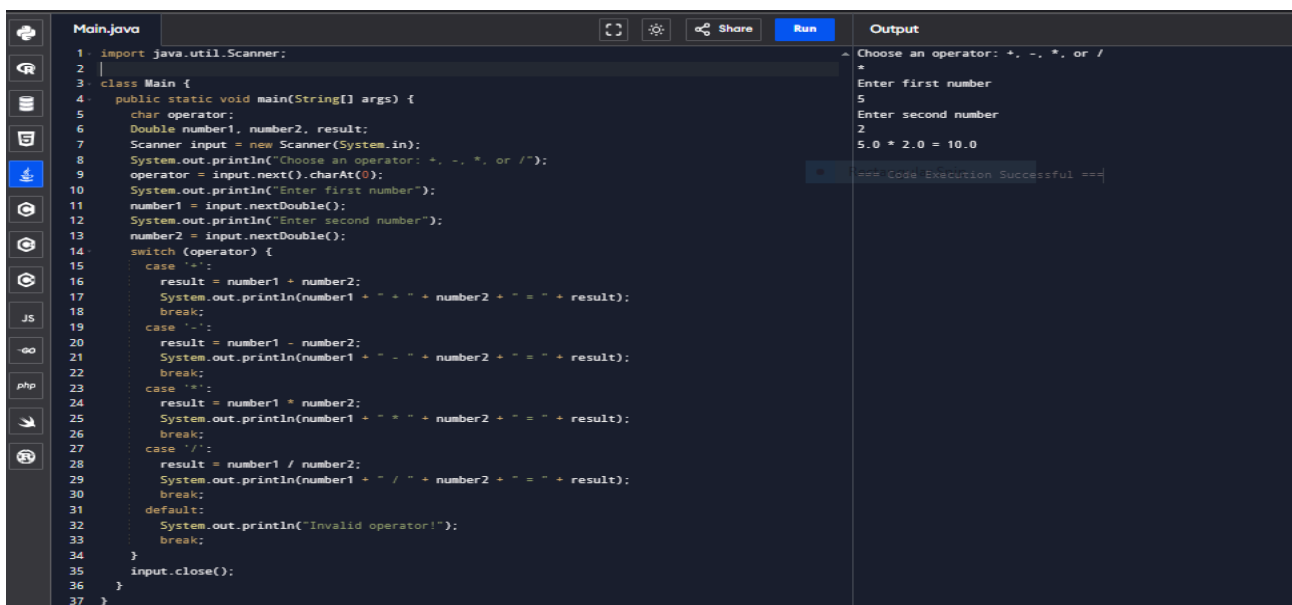
```
1
2- public class Main {
3-     public static void main(String[] args) {
4-         int number = 121;
5-         if (isPalindrome(number)) {
6-             System.out.println(number + " is a palindrome.");
7-         } else {
8-             System.out.println(number + " is not a palindrome.");
9-         }
10-    }
11-
12-    public static boolean isPalindrome(int number) {
13-        int originalNumber = number;
14-        int reversedNumber = 0;
15-        while (number != 0) {
16-            int digit = number % 10;
17-            reversedNumber = reversedNumber * 10 + digit;
18-            number /= 10;
19-        }
20-        return originalNumber == reversedNumber;
21-    }
22- }
```

Output

```
121 is a palindrome.

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

Q9.Program for a simple calculator



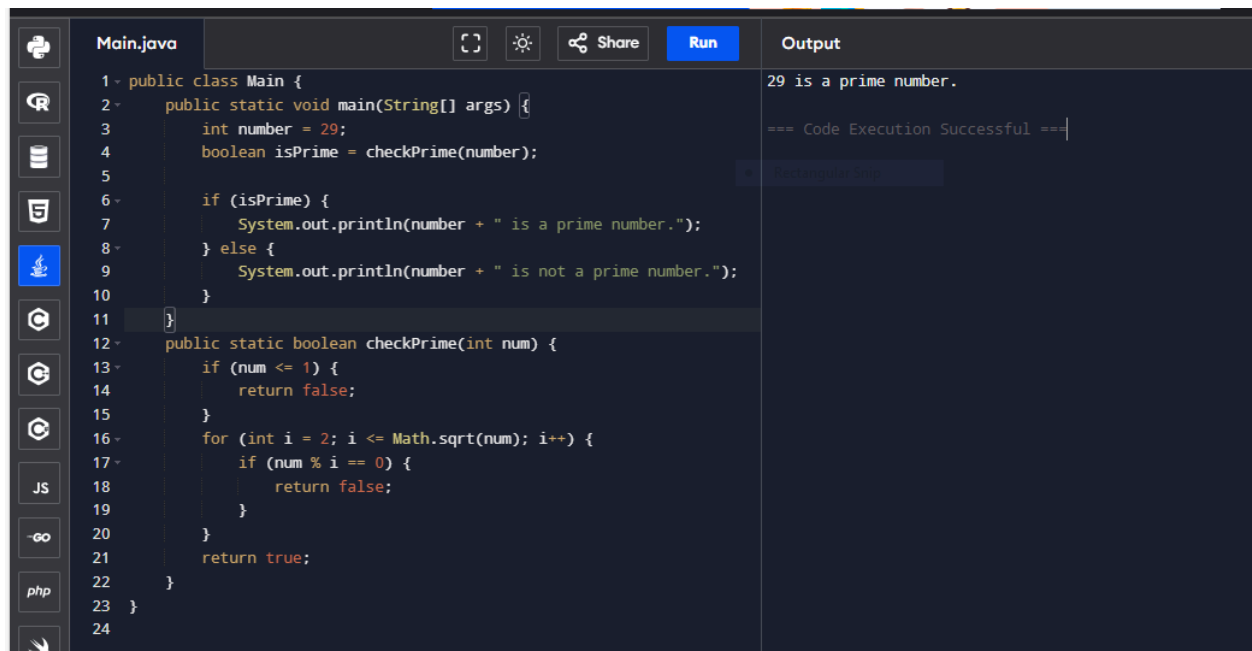
```
1- import java.util.Scanner;
2
3- class Main {
4-     public static void main(String[] args) {
5-         char operator;
6-         Double number1, number2, result;
7-         Scanner input = new Scanner(System.in);
8-         System.out.println("Choose an operator: +, -, *, or /");
9-         operator = input.next().charAt(0);
10-        System.out.println("Enter first number");
11-        number1 = input.nextDouble();
12-        System.out.println("Enter second number");
13-        number2 = input.nextDouble();
14-        switch (operator) {
15-            case '+':
16-                result = number1 + number2;
17-                System.out.println(number1 + " + " + number2 + " = " + result);
18-                break;
19-            case '-':
20-                result = number1 - number2;
21-                System.out.println(number1 + " - " + number2 + " = " + result);
22-                break;
23-            case '*':
24-                result = number1 * number2;
25-                System.out.println(number1 + " * " + number2 + " = " + result);
26-                break;
27-            case '/':
28-                result = number1 / number2;
29-                System.out.println(number1 + " / " + number2 + " = " + result);
30-                break;
31-            default:
32-                System.out.println("Invalid operator!");
33-                break;
34-        }
35-        input.close();
36-    }
37- }
```

Output

```
Choose an operator: +, -, *, or /
*
Enter first number
5
Enter second number
2
5.0 * 2.0 = 10.0

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

Q10.Program to check if a number is prime



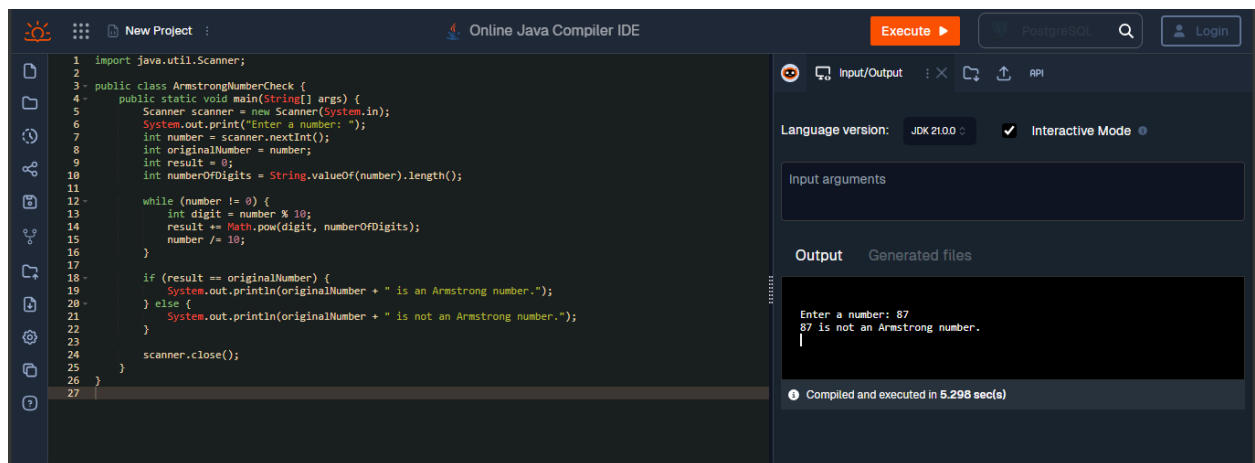
```
1 public class Main {
2     public static void main(String[] args) {
3         int number = 29;
4         boolean isPrime = checkPrime(number);
5
6         if (isPrime) {
7             System.out.println(number + " is a prime number.");
8         } else {
9             System.out.println(number + " is not a prime number.");
10        }
11    }
12    public static boolean checkPrime(int num) {
13        if (num <= 1) {
14            return false;
15        }
16        for (int i = 2; i <= Math.sqrt(num); i++) {
17            if (num % i == 0) {
18                return false;
19            }
20        }
21        return true;
22    }
23 }
24
```

Output

29 is a prime number.

=== Code Execution Successful ===

Q11. Program to check if a number is an Armstrong number



```
1 import java.util.Scanner;
2
3 public class ArmstrongNumberCheck {
4     public static void main(String[] args) {
5         Scanner scanner = new Scanner(System.in);
6         System.out.print("Enter a number: ");
7         int number = scanner.nextInt();
8         int originalNumber = number;
9         int result = 0;
10        int numberOfDigits = String.valueOf(number).length();
11
12        while (number != 0) {
13            int digit = number % 10;
14            result += Math.pow(digit, numberOfDigits);
15            number /= 10;
16        }
17
18        if (result == originalNumber) {
19            System.out.println(originalNumber + " is an Armstrong number.");
20        } else {
21            System.out.println(originalNumber + " is not an Armstrong number.");
22        }
23
24        scanner.close();
25    }
26 }
27
```

Language version: JDK 21.0.0 Interactive Mode

Input arguments

Output

Enter a number: 87
87 is not an Armstrong number.

Compiled and executed in 5.298 sec(s)

Q12. Find the Largest of Three Numbers using ternary operator



```
1 import java.util.Scanner;
2
3 public class LargestNumber {
4     public static void main(String[] args) {
5         Scanner scanner = new Scanner(System.in);
6         System.out.print("Enter the first number: ");
7         int num1 = scanner.nextInt();
8         System.out.print("Enter the second number: ");
9         int num2 = scanner.nextInt();
10
11        // Using the ternary operator to find the largest number
12        int largest = (num1 > num2) ? num1 : num2;
13
14        System.out.println("The largest number is: " + largest);
15        scanner.close();
16    }
17 }
18
```

Language version: JDK 21.0.0 Interactive Mode

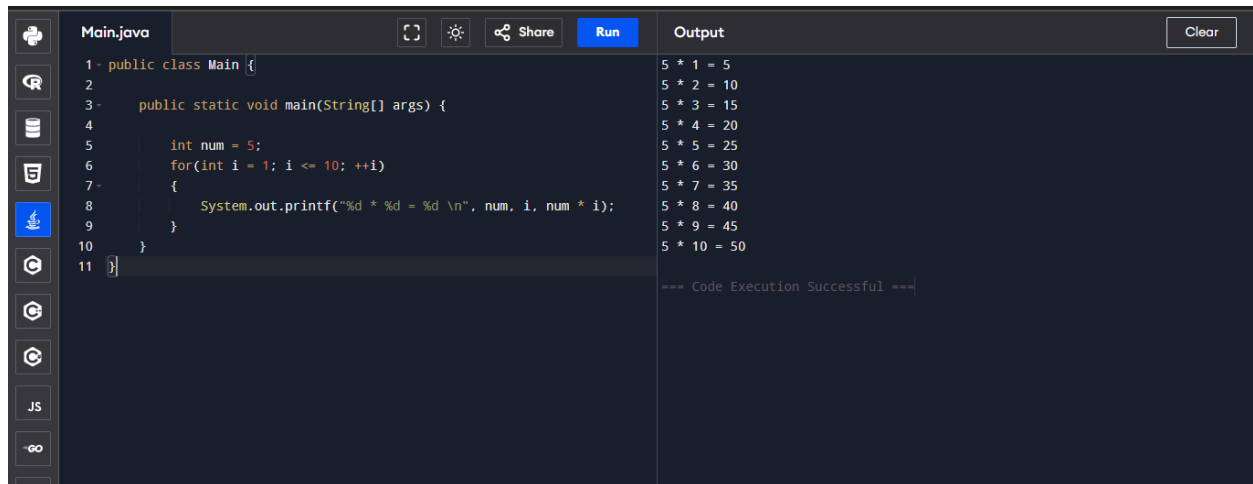
Input arguments

Output

Enter the first number: 54
Enter the second number: 84
The largest number is: 84

Compiled and executed in 12.316 sec(s)

Q13. Print Multiplication Table



The screenshot shows an online Java IDE with a dark theme. The editor displays a Java program named 'Main.java' that prints a multiplication table for the number 5, ranging from 1 to 10. The code is as follows:

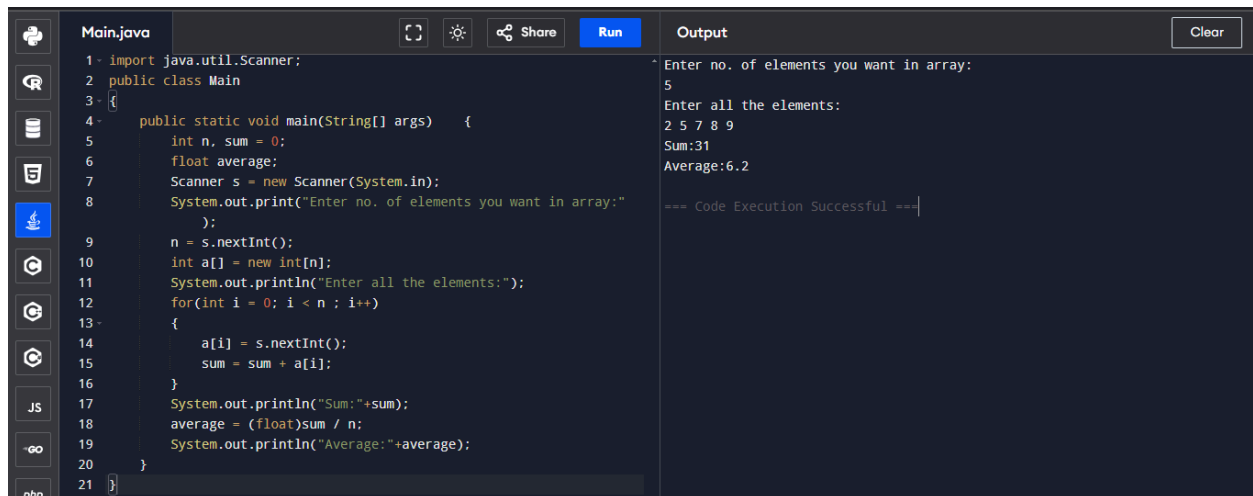
```
1- public class Main {
2-
3-     public static void main(String[] args) {
4-
5-         int num = 5;
6-         for(int i = 1; i <= 10; ++i)
7-         {
8-             System.out.printf("%d * %d = %d \n", num, i, num * i);
9-         }
10-    }
11-}
```

The output window on the right shows the execution results:

```
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50

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

Q14. Calculate Sum and Average of Array Elements



The screenshot shows an online Java IDE with a dark theme. The editor displays a Java program named 'Main.java' that calculates the sum and average of an array of integers. The code is as follows:

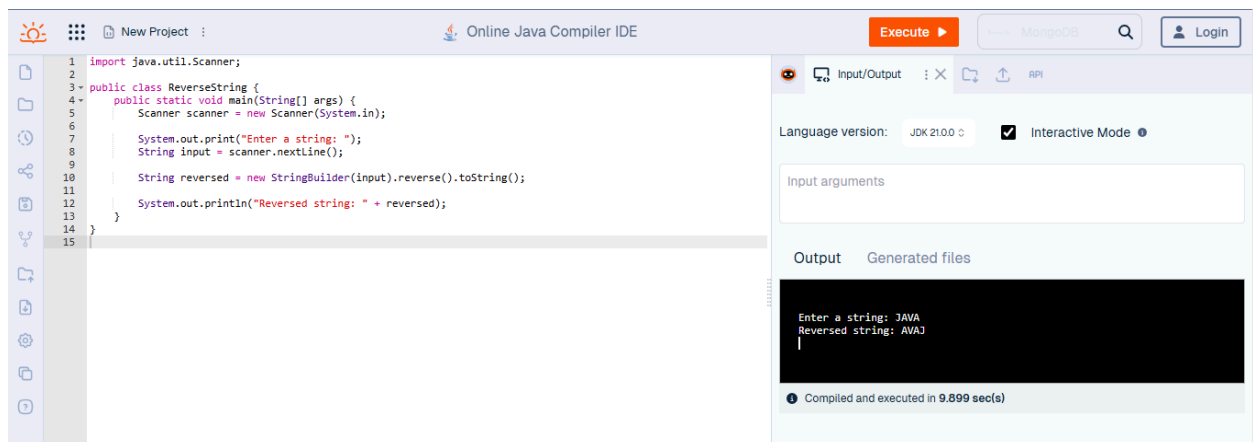
```
1- import java.util.Scanner;
2- public class Main
3- {
4-     public static void main(String[] args) {
5-         int n, sum = 0;
6-         float average;
7-         Scanner s = new Scanner(System.in);
8-         System.out.print("Enter no. of elements you want in array:");
9-         n = s.nextInt();
10-        int a[] = new int[n];
11-        System.out.println("Enter all the elements:");
12-        for(int i = 0; i < n; i++)
13-        {
14-            a[i] = s.nextInt();
15-            sum = sum + a[i];
16-        }
17-        System.out.println("Sum: "+sum);
18-        average = (float)sum / n;
19-        System.out.println("Average: "+average);
20-    }
21-}
```

The output window on the right shows the execution results:

```
Enter no. of elements you want in array:
5
Enter all the elements:
2 5 7 8 9
Sum:31
Average:6.2

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

Q15. Reverse a String



The screenshot shows an online Java IDE with a light theme. The editor displays a Java program named 'ReverseString.java' that reverses a string. The code is as follows:

```
1- import java.util.Scanner;
2-
3- public class ReverseString {
4-     public static void main(String[] args) {
5-         Scanner scanner = new Scanner(System.in);
6-
7-         System.out.print("Enter a string: ");
8-         String input = scanner.nextLine();
9-
10-        String reversed = new StringBuilder(input).reverse().toString();
11-
12-        System.out.println("Reversed string: " + reversed);
13-    }
14-}
15-
```

The output window on the right shows the execution results:

```
Enter a string: JAVA
Reversed string: AVAJ
```

The IDE also shows the language version as 'JDK 21.0.0' and 'Interactive Mode' is checked. The output was compiled and executed in 9.899 seconds.

Q16. Find Factorial of a Number Using Recursion

The screenshot shows an online Java IDE with a dark theme. On the left is a sidebar with icons for file explorer, search, and other IDE features. The main editor displays a Java file named 'Main.java' with the following code:

```
1- public class Main {
2-
3-     public static void main(String[] args) {
4-         int num = 6;
5-         long factorial = multiplyNumbers(num);
6-         System.out.println("Factorial of " + num + " = " + factorial);
7-     }
8-     public static long multiplyNumbers(int num)
9-     {
10-         if (num >= 1)
11-             return num * multiplyNumbers(num - 1);
12-         else
13-             return 1;
14-     }
15- }
```

Below the code editor is a 'Run' button and a 'Share' icon. To the right of the code editor is an 'Output' panel with a 'Clear' button. The output panel displays the result of the program execution:

```
Factorial of 6 = 720
=== Code Execution Successful ===
```

Q17. Sort an Array in Ascending Order

The screenshot shows an online Java IDE with a dark theme. The main editor displays a Java file named 'Main.java' with the following code:

```
1- public class Main {
2-     public static void main(String[] args) {
3-         int [] arr = new int [] {5, 2, 8, 7, 1};
4-         int temp = 0;
5-         System.out.println("Elements of original array: ");
6-         for (int i = 0; i < arr.length; i++) {
7-             System.out.print(arr[i] + " ");
8-         }
9-         for (int i = 0; i < arr.length; i++) {
10-             for (int j = i+1; j < arr.length; j++) {
11-                 if(arr[i] > arr[j]) {
12-                     temp = arr[i];
13-                     arr[i] = arr[j];
14-                     arr[j] = temp;
15-                 }
16-             }
17-         }
18-         System.out.println();
19-         System.out.println("Elements of array sorted in ascending order: ");
20-         for (int i = 0; i < arr.length; i++) {
21-             System.out.print(arr[i] + " ");
22-         }
23-     }
24- }
```

The 'Output' panel on the right shows the execution results:

```
Elements of original array:
5 2 8 7 1
Elements of array sorted in ascending order:
1 2 5 7 8
=== Code Execution Successful ===
```

Q18. Check Palindrome for a String

The screenshot shows an online Java IDE with a dark theme. The main editor displays a Java file named 'PalindromeCheck.java' with the following code:

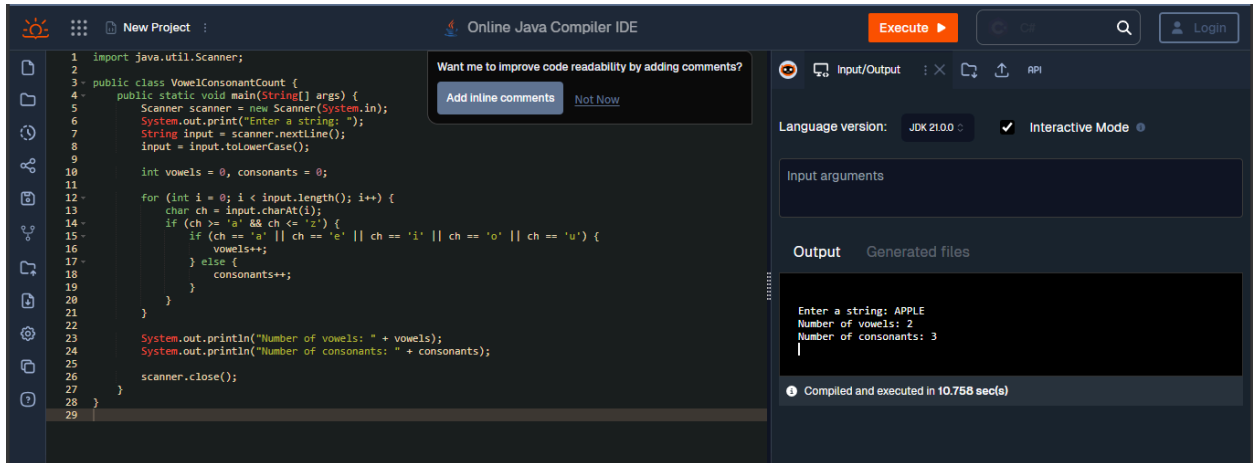
```
1- import java.util.Scanner;
2-
3- public class PalindromeCheck {
4-     public static void main(String[] args) {
5-         Scanner scanner = new Scanner(System.in);
6-         System.out.print("Enter a string: ");
7-         String input = scanner.nextLine();
8-         String reversedString = new StringBuilder(input).reverse().toString();
9-
10-         if (input.equalsIgnoreCase(reversedString)) {
11-             System.out.println(input + " is a palindrome.");
12-         } else {
13-             System.out.println(input + " is not a palindrome.");
14-         }
15-         scanner.close();
16-     }
17- }
```

The 'Output' panel on the right shows the execution results for the input 'aashi':

```
Enter a string: aashi
aashi is not a palindrome.
```

Below the output, it states: 'Compiled and executed in 7.267 sec(s)'.

Q19. Count Vowels and Consonants in a String



```
1 import java.util.Scanner;
2
3 public class VowelConsonantCount {
4     public static void main(String[] args) {
5         Scanner scanner = new Scanner(System.in);
6         System.out.print("Enter a string: ");
7         String input = scanner.nextLine();
8         input = input.toLowerCase();
9
10        int vowels = 0, consonants = 0;
11
12        for (int i = 0; i < input.length(); i++) {
13            char ch = input.charAt(i);
14            if (ch >= 'a' && ch <= 'z') {
15                if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {
16                    vowels++;
17                } else {
18                    consonants++;
19                }
20            }
21        }
22
23        System.out.println("Number of vowels: " + vowels);
24        System.out.println("Number of consonants: " + consonants);
25
26        scanner.close();
27    }
28 }
29
```

Want me to improve code readability by adding comments?
[Add inline comments](#) [Not Now](#)

Language version: JDK 21.0.0 ☒ Interactive Mode

Input arguments

Output Generated files

Enter a string: APPLE
Number of vowels: 2
Number of consonants: 3

Compiled and executed in 10.758 sec(s)

Q20.Implement a Simple Banking System



```
1 public class SimpleBankingSystem {
2     private String accountHolderName;
3     private double initialBalance;
4
5     public SimpleBankingSystem(String accountHolderName, double initialBalance) {
6         this.accountHolderName = accountHolderName;
7         this.balance = initialBalance;
8     }
9
10    private double balance;
11
12    public void deposit(double amount) {
13        if (amount > 0) {
14            balance += amount;
15            System.out.println("Deposited: " + amount);
16        } else {
17            System.out.println("Invalid deposit amount.");
18        }
19    }
20
21    public void withdraw(double amount) {
22        if (amount > 0 && amount <= balance) {
23            balance -= amount;
24            System.out.println("Withdrawn: " + amount);
25        } else {
26            System.out.println("Invalid withdrawal amount or insufficient balance.");
27        }
28    }
29
30    public void checkBalance() {
31        System.out.println("Current balance: " + balance);
32    }
33
34    public String getAccountHolderName() {
35        return accountHolderName;
36    }
37 }
38
39 public class SimpleBankingSystem {

```

Stop Execution

Language version: JDK 21.0.0 ☒ Interactive Mode

Input arguments

Output Generated files

Enter account holder name: AASHI
Enter initial balance: 2000

Banking System Menu:
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Choose an option: 2
Enter withdrawal amount: 500
Withdrawn: 500.0

Banking System Menu:

Q21.Write a program to demonstrate type casting.



```
1 public class TypeCastingDemo {
2     public static void main(String[] args) {
3         // Implicit (Widening) Type Casting
4         int intValue = 100;
5         double doubleValue = intValue; // int to double
6         System.out.println("Implicit Type Casting:");
7         System.out.println("int value: " + intValue);
8         System.out.println("double value: " + doubleValue);
9
10        // Explicit (Narrowing) Type Casting
11        double anotherDoubleValue = 9.78;
12        int anotherIntValue = (int) anotherDoubleValue; // double to int
13        System.out.println("\nExplicit Type Casting:");
14        System.out.println("double value: " + anotherDoubleValue);
15        System.out.println("int value: " + anotherIntValue);
16    }
17 }
18
```

Want me to debug your code?
[Debug code](#) [Not Now](#)

Language version: JDK 21.0.0 ☒ Interactive Mode

Input arguments

Output Generated files

Implicit Type Casting:
int value: 100
double value: 100.0

Explicit Type Casting:
double value: 9.78
int value: 9

Compiled and executed in 1.257 sec(s)

Q22.Write a program to generate prime numbers between 1 & given number

The screenshot shows an online Java compiler IDE with a dark theme. The code editor on the left contains a Java program to find prime numbers between 1 and 54. The program uses a Scanner to take input and a loop to check for primes. The output panel on the right shows the execution results, including the input number 54 and the list of prime numbers between 1 and 54. The IDE interface includes a top bar with 'New Project', 'Online Java Compiler IDE', 'Execute', 'COBOL', and 'Login' buttons. A sidebar on the left contains icons for file management and settings. A small pop-up window asks 'Want me to explain code logic?' with 'Explain code' and 'Not Now' buttons.

```
1 import java.util.Scanner;
2
3 public class PrintNumbers {
4     public static void main(String[] args) {
5         Scanner scanner = new Scanner(System.in);
6         System.out.print("Enter a number: ");
7         int n = scanner.nextInt();
8
9         System.out.println("Prime numbers between 1 and " + n + " are:");
10        for (int i = 2; i <= n; i++) {
11            if (isPrime(i)) {
12                System.out.print(i + " ");
13            }
14        }
15        scanner.close();
16    }
17
18    public static boolean isPrime(int num) {
19        if (num <= 1) {
20            return false;
21        }
22        for (int i = 2; i <= Math.sqrt(num); i++) {
23            if (num % i == 0) {
24                return false;
25            }
26        }
27        return true;
28    }
29 }
30 }
```

Language version: JDK 21.0.0 Interactive Mode

Input arguments

Output Generated files

Enter a number: 54
Prime numbers between 1 and 54 are:
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53

Compiled and executed in 8.776 sec(s)

Q23. Program to Demonstrate a Simple Class with Methods

The screenshot shows an online Java compiler IDE with a dark theme. The code editor on the left contains a Java program demonstrating a simple class with methods. The program defines a SimpleClass with methods for printing a greeting, adding two numbers, and multiplying two numbers. The main method creates an instance of SimpleClass and calls these methods. The output panel on the right shows the execution results, including the greeting message and the results of the arithmetic operations. The IDE interface includes a top bar with 'New Project', 'Online Java Compiler IDE', 'Execute', 'C', and 'Login' buttons. A sidebar on the left contains icons for file management and settings. A small pop-up window asks 'Want me to explain code logic?' with 'Explain code' and 'Not Now' buttons.

```
1 public class SimpleClass {
2     public void printGreeting() {
3         System.out.println("Hello, welcome to the simple class demonstration!");
4     }
5
6     public int addNumbers(int num1, int num2) {
7         return num1 + num2;
8     }
9
10    public int multiplyNumbers(int num1, int num2) {
11        return num1 * num2;
12    }
13
14    public static void main(String[] args) {
15        SimpleClass simpleClass = new SimpleClass();
16        simpleClass.printGreeting();
17
18        int sum = simpleClass.addNumbers(5, 10);
19        System.out.println("The sum of 5 and 10 is: " + sum);
20
21        int product = simpleClass.multiplyNumbers(5, 10);
22        System.out.println("The product of 5 and 10 is: " + product);
23    }
24 }
25 }
```

Language version: JDK 21.0.0 Interactive Mode

Input arguments

Output Generated files

Hello, welcome to the simple class demonstration!
The sum of 5 and 10 is: 15
The product of 5 and 10 is: 50

Compiled and executed in 1.337 sec(s)

Q24. Program for Class with Parameterized Constructor

The screenshot shows an online Java compiler IDE with a dark theme. The code editor on the left contains a Java program demonstrating a class with a parameterized constructor. The program defines a Student class with private attributes for name, age, and grade, and a parameterized constructor to initialize them. The main method creates an instance of Student and calls a displayDetails method. The output panel on the right shows the execution results, including the student's details. The IDE interface includes a top bar with 'New Project', 'Online Java Compiler IDE', 'Execute', 'C', and 'Login' buttons. A sidebar on the left contains icons for file management and settings. A small pop-up window asks 'Want me to explain code logic?' with 'Explain code' and 'Not Now' buttons.

```
1 public class Student {
2     private String name;
3     private int age;
4     private String grade;
5
6     // Parameterized constructor
7     public Student(String name, int age, String grade) {
8         this.name = name;
9         this.age = age;
10        this.grade = grade;
11    }
12
13    // Method to display student details
14    public void displayDetails() {
15        System.out.println("Name: " + name);
16        System.out.println("Age: " + age);
17        System.out.println("Grade: " + grade);
18    }
19
20    public static void main(String[] args) {
21        // Create an object of the Student class using the parameterized constructor
22        Student student = new Student("Alice", 20, "A");
23
24        // Call the displayDetails method
25        student.displayDetails();
26    }
27 }
28 }
```

Language version: JDK 21.0.0 Interactive Mode

Input arguments

Output Generated files

Name: Alice
Age: 20
Grade: A

Compiled and executed in 1.372 sec(s)

Q25. Program to Find the Area of a Rectangle Using Methods

The screenshot shows the Online Java Compiler IDE interface. The code editor on the left contains a Java program for calculating the area of a rectangle. The code is as follows:

```
1 import java.util.Scanner;
2
3 public class RectangleArea {
4     // Method to calculate the area of the rectangle
5     public double calculateArea(double length, double width) {
6         return length * width;
7     }
8
9     public static void main(String[] args) {
10         Scanner scanner = new Scanner(System.in);
11
12         // Create an object of the RectangleArea class
13         RectangleArea rectangle = new RectangleArea();
14
15         // Input length and width from the user
16         System.out.print("Enter the length of the rectangle: ");
17         double length = scanner.nextDouble();
18         System.out.print("Enter the width of the rectangle: ");
19         double width = scanner.nextDouble();
20
21         // Calculate the area using the calculateArea method
22         double area = rectangle.calculateArea(length, width);
23
24         // Print the result
25         System.out.println("The area of the rectangle is: " + area);
26
27         scanner.close();
28     }
29 }
30
```

The right sidebar shows the 'Input/Output' tab. The 'Language version' is set to 'JDK 21.0.0' and 'Interactive Mode' is checked. The 'Output' tab displays the following text:

```
Enter the length of the rectangle: 4
Enter the width of the rectangle: 8
The area of the rectangle is: 32.0
```

Below the output, it states 'Compiled and executed in 12.835 sec(s)'.

Q26. Program for Bank Account Class with Deposit and Withdraw Methods

The screenshot shows the Online Java Compiler IDE interface. The code editor on the left contains a Java program for a Bank Account class. The code is as follows:

```
1 public class BankAccount {
2     private double balance;
3
4     public BankAccount() {
5         this.balance = 0.0;
6     }
7
8     public BankAccount(double initialBalance) {
9         this.balance = initialBalance;
10    }
11
12    public void deposit(double amount) {
13        if (amount > 0) {
14            balance += amount;
15            System.out.println("Deposited: " + amount);
16        } else {
17            System.out.println("Deposit amount must be positive.");
18        }
19    }
20
21    public void withdraw(double amount) {
22        if (amount > 0 && amount <= balance) {
23            balance -= amount;
24            System.out.println("Withdraw: " + amount);
25        } else if (amount > balance) {
26            System.out.println("Insufficient balance.");
27        } else {
28            System.out.println("Withdrawal amount must be positive.");
29        }
30    }
31
32    public double getBalance() {
33        return balance;
34    }
35 }
```

The right sidebar shows the 'Input/Output' tab. The 'Language version' is set to 'JDK 21.0.0' and 'Interactive Mode' is checked. The 'Output' tab displays the following text:

```
Initial Balance: 1000.0
Deposited: 500.0
Current Balance: 1500.0
Withdraw: 200.0
Current Balance: 1300.0
Insufficient balance.
Current Balance: 1300.0
```

Below the output, it states 'Compiled and executed in 1.88 sec(s)'.

Q27. Program to Demonstrate Static Methods

The screenshot shows the Online Java Compiler IDE interface. The code editor on the left contains a Java program demonstrating static methods. The code is as follows:

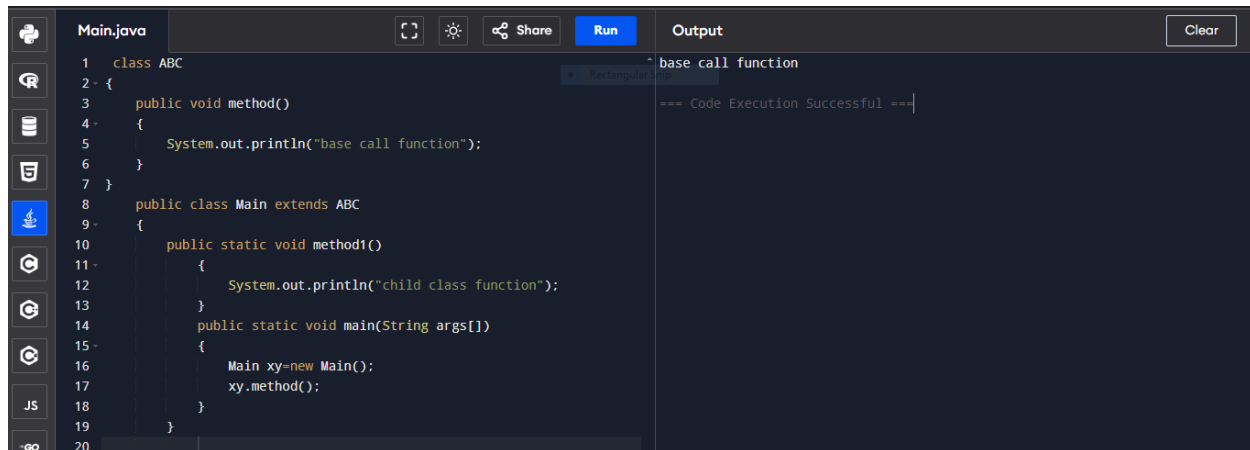
```
1 public class Main {
2     static void staticmethod() {
3         System.out.println("static function called");
4     }
5
6     public static void main(String arg[]) {
7         Main.staticmethod();
8     }
9 }
10
```

The right sidebar shows the 'Input/Output' tab. The 'Language version' is set to 'JDK 21.0.0' and 'Interactive Mode' is checked. The 'Output' tab displays the following text:

```
static function called
```

Below the output, it states 'Compiled and executed in 2.575 sec(s)'.

Q28. Program to Demonstrate Method Overriding

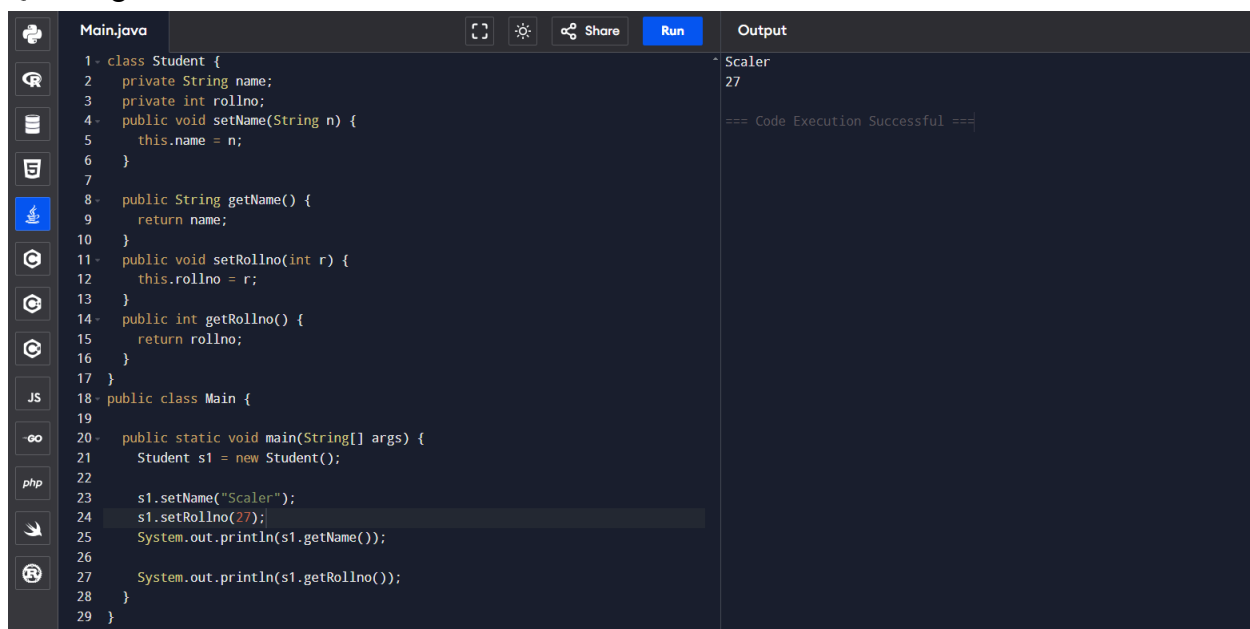


```
1 class ABC
2 {
3     public void method()
4     {
5         System.out.println("base call function");
6     }
7 }
8 public class Main extends ABC
9 {
10     public static void method1()
11     {
12         System.out.println("child class function");
13     }
14     public static void main(String args[])
15     {
16         Main xy=new Main();
17         xy.method();
18     }
19 }
20
```

base call function

=== Code Execution Successful ===

Q29.Program to Demonstrate Getters and Setters



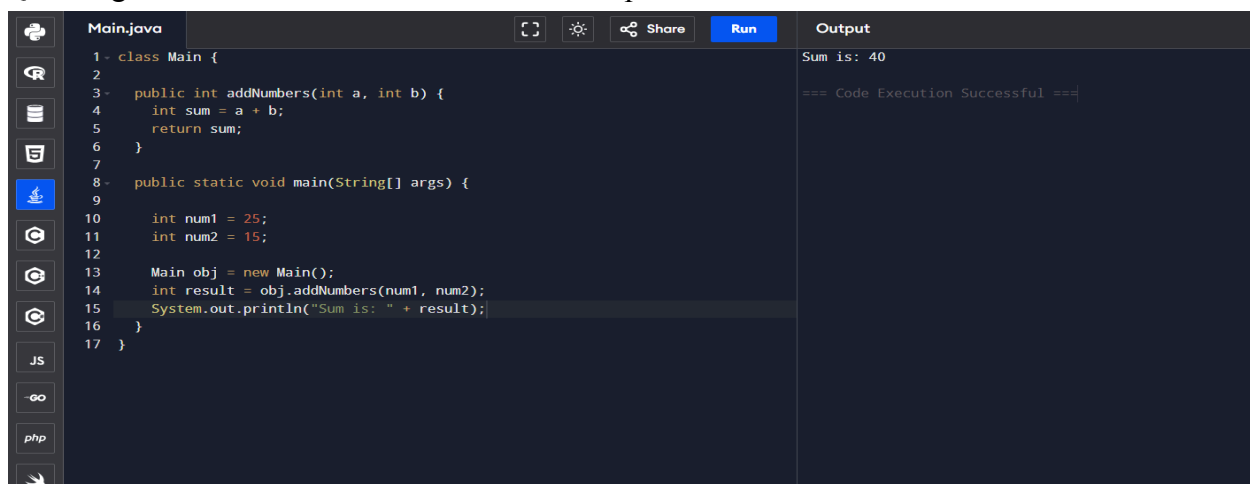
```
1 class Student {
2     private String name;
3     private int rollno;
4     public void setName(String n) {
5         this.name = n;
6     }
7
8     public String getName() {
9         return name;
10    }
11    public void setRollno(int r) {
12        this.rollno = r;
13    }
14    public int getRollno() {
15        return rollno;
16    }
17 }
18 public class Main {
19
20     public static void main(String[] args) {
21         Student s1 = new Student();
22
23         s1.setName("Scaler");
24         s1.setRollno(27);
25         System.out.println(s1.getName());
26
27         System.out.println(s1.getRollno());
28     }
29 }
30
```

Scaler

27

=== Code Execution Successful ===

Q30.Program to Demonstrate a Class with Multiple Methods



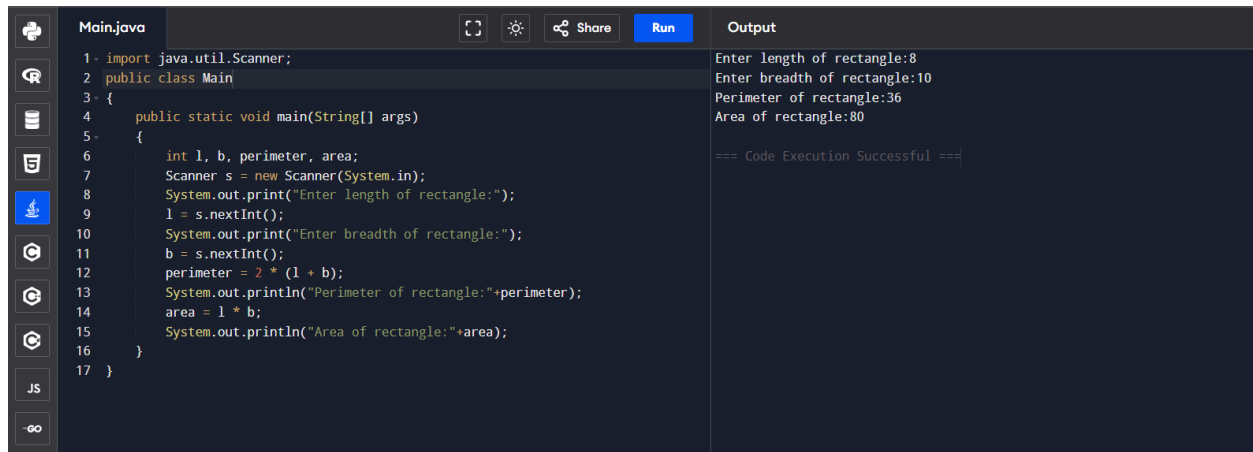
```
1 class Main {
2
3     public int addNumbers(int a, int b) {
4         int sum = a + b;
5         return sum;
6     }
7
8     public static void main(String[] args) {
9
10        int num1 = 25;
11        int num2 = 15;
12
13        Main obj = new Main();
14        int result = obj.addNumbers(num1, num2);
15        System.out.println("Sum is: " + result);
16    }
17 }
```

Sum is: 40

=== Code Execution Successful ===

Q31.Write a program to create a simple class to find out the area and perimeter of rectangle

using super and this keyword.



The screenshot shows a Java IDE with a file named 'Main.java'. The code imports 'java.util.Scanner' and defines a 'Main' class with a 'main' method. Inside the 'main' method, it prompts the user to enter the length and breadth of a rectangle, reads the input, and calculates the perimeter and area. The output window shows the user input and the calculated values.

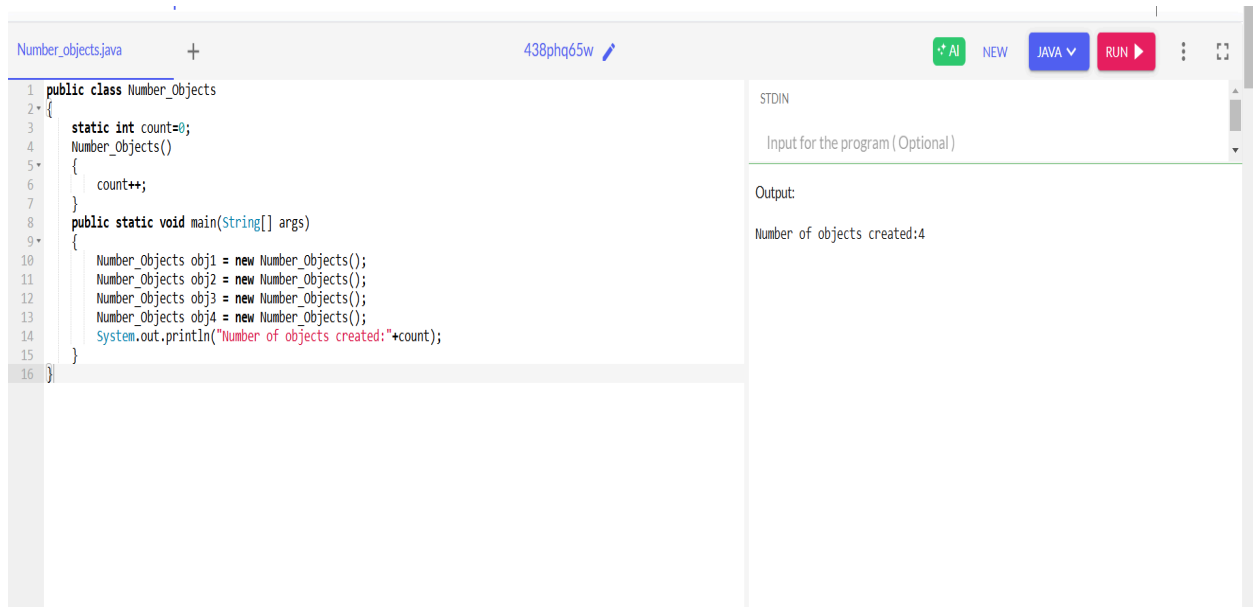
```
1 import java.util.Scanner;
2 public class Main
3 {
4     public static void main(String[] args)
5     {
6         int l, b, perimeter, area;
7         Scanner s = new Scanner(System.in);
8         System.out.print("Enter length of rectangle:");
9         l = s.nextInt();
10        System.out.print("Enter breadth of rectangle:");
11        b = s.nextInt();
12        perimeter = 2 * (l + b);
13        System.out.println("Perimeter of rectangle:"+perimeter);
14        area = l * b;
15        System.out.println("Area of rectangle:"+area);
16    }
17 }
```

Output

```
Enter length of rectangle:8
Enter breadth of rectangle:10
Perimeter of rectangle:36
Area of rectangle:80

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

Q32. Write a program to count the number of objects created for a class using static member function



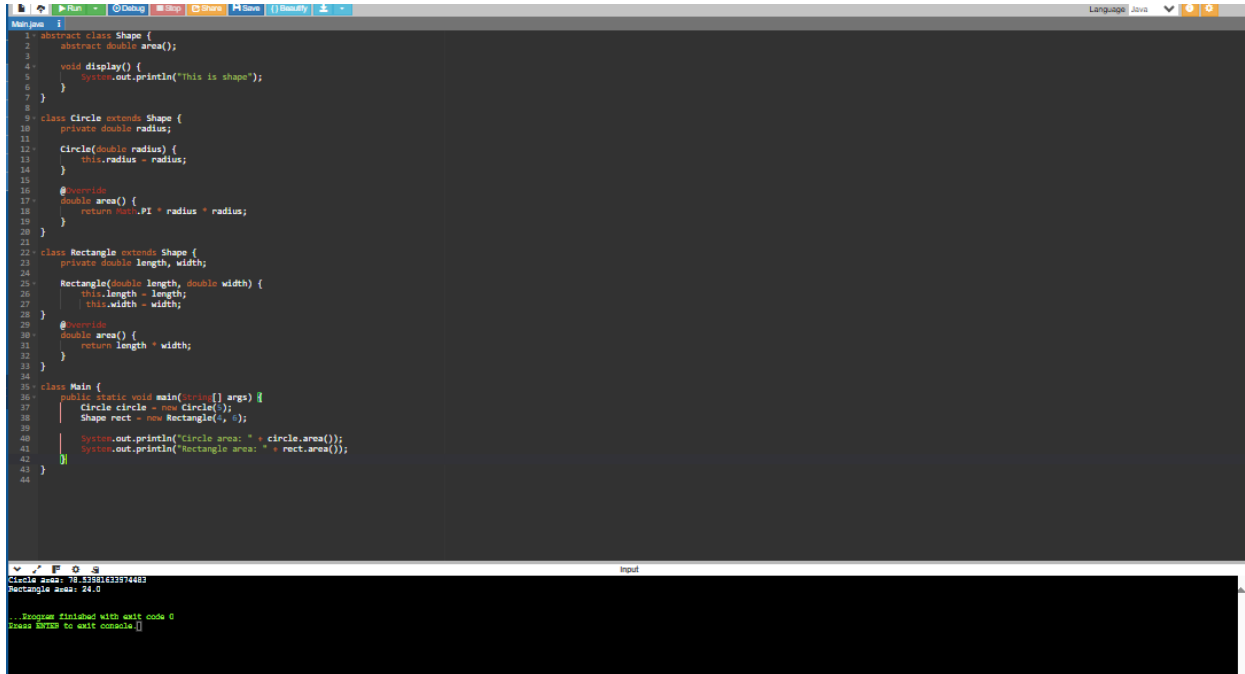
The screenshot shows a Java IDE with a file named 'Number_objects.java'. The code defines a 'Number_Objects' class with a static 'count' variable and a 'main' method. The 'main' method creates four objects of the 'Number_Objects' class and prints the total count. The output window shows the result.

```
1 public class Number_Objects
2 {
3     static int count=0;
4     Number_Objects()
5     {
6         count++;
7     }
8     public static void main(String[] args)
9     {
10        Number_Objects obj1 = new Number_Objects();
11        Number_Objects obj2 = new Number_Objects();
12        Number_Objects obj3 = new Number_Objects();
13        Number_Objects obj4 = new Number_Objects();
14        System.out.println("Number of objects created:"+count);
15    }
16 }
```

Output:

```
Number of objects created:4
```

Q33. Write a program to design a class using abstract methods and abstract classes.

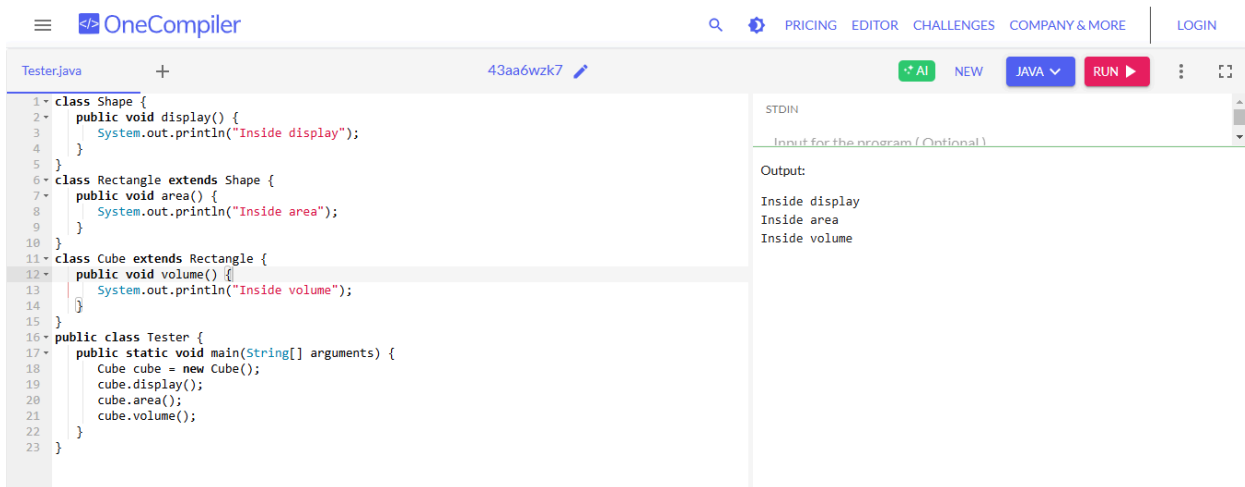


```
1 abstract class Shape {
2     abstract double area();
3
4     void display() {
5         System.out.println("this is shape");
6     }
7 }
8
9 class Circle extends Shape {
10     private double radius;
11
12     Circle(double radius) {
13         this.radius = radius;
14     }
15
16     @Override
17     double area() {
18         return Math.PI * radius * radius;
19     }
20 }
21
22 class Rectangle extends Shape {
23     private double length, width;
24
25     Rectangle(double length, double width) {
26         this.length = length;
27         this.width = width;
28     }
29
30     @Override
31     double area() {
32         return length * width;
33     }
34 }
35
36 class Main {
37     public static void main(String[] args) {
38         Circle circle = new Circle(5);
39         Shape rect = new Rectangle(4, 6);
40         System.out.println("Circle area: " + circle.area());
41         System.out.println("Rectangle area: " + rect.area());
42     }
43 }
44
```

Circle area: 78.53981633974483
Rectangle area: 24.0

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

Q34. Write a program to demonstrate the use of multilevel inheritance



```
1 class Shape {
2     public void display() {
3         System.out.println("Inside display");
4     }
5 }
6 class Rectangle extends Shape {
7     public void area() {
8         System.out.println("Inside area");
9     }
10 }
11 class Cube extends Rectangle {
12     public void volume() {
13         System.out.println("Inside volume");
14     }
15 }
16 public class Tester {
17     public static void main(String[] arguments) {
18         Cube cube = new Cube();
19         cube.display();
20         cube.area();
21         cube.volume();
22     }
23 }
```

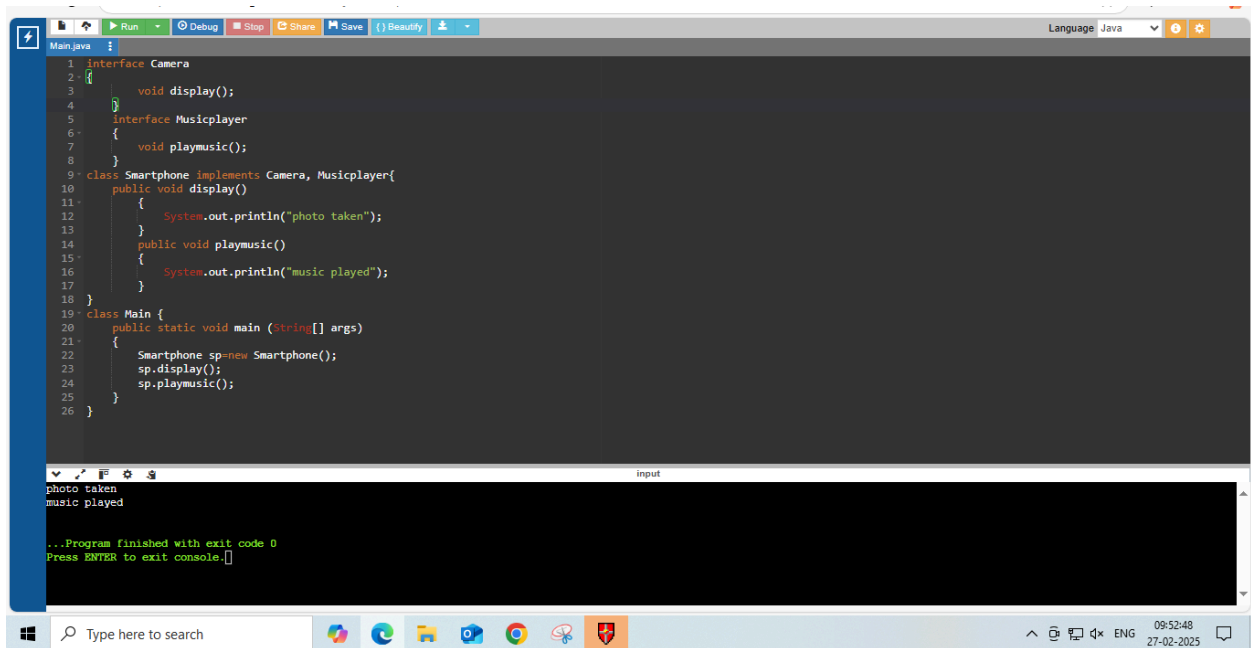
STDIN

Input for the program (Optional)

Output:

Inside display
Inside area
Inside volume

Q35. Write a program to demonstrate the use of multiple inheritance

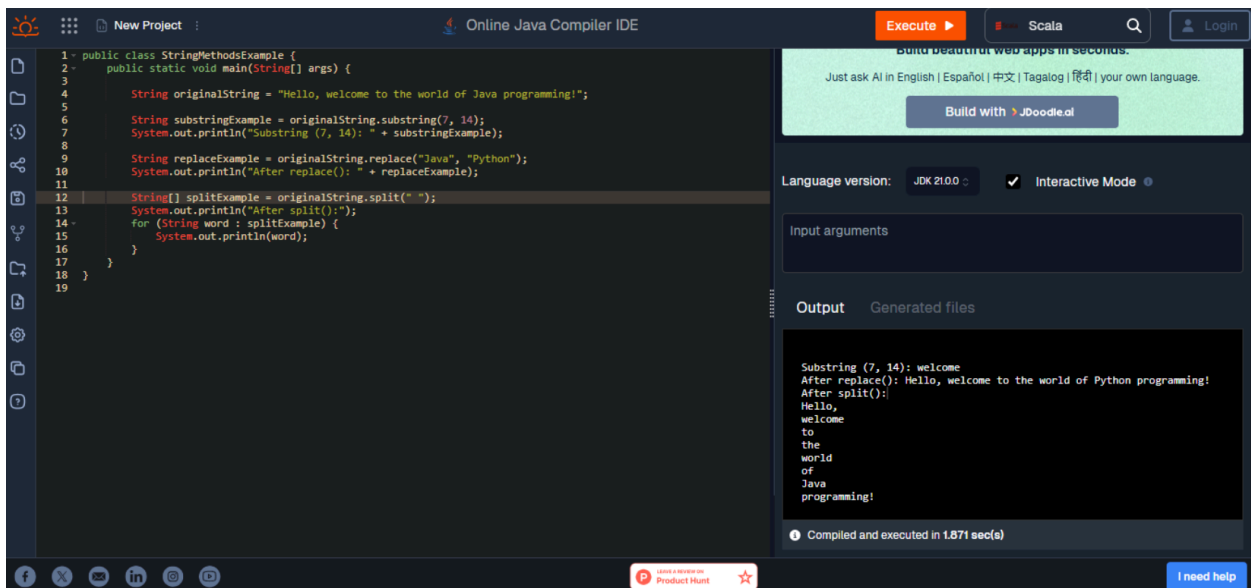


```
1 interface Camera
2 {
3     void display();
4 }
5 interface Musicplayer
6 {
7     void playmusic();
8 }
9 class Smartphone implements Camera, Musicplayer{
10     public void display()
11     {
12         System.out.println("photo taken");
13     }
14     public void playmusic()
15     {
16         System.out.println("music played");
17     }
18 }
19 class Main {
20     public static void main (String[] args)
21     {
22         Smartphone sp=new Smartphone();
23         sp.display();
24         sp.playmusic();
25     }
26 }
```

photo taken
music played

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

Q36. Write a Java program demonstrating String methods like substring(), replace(), and split().

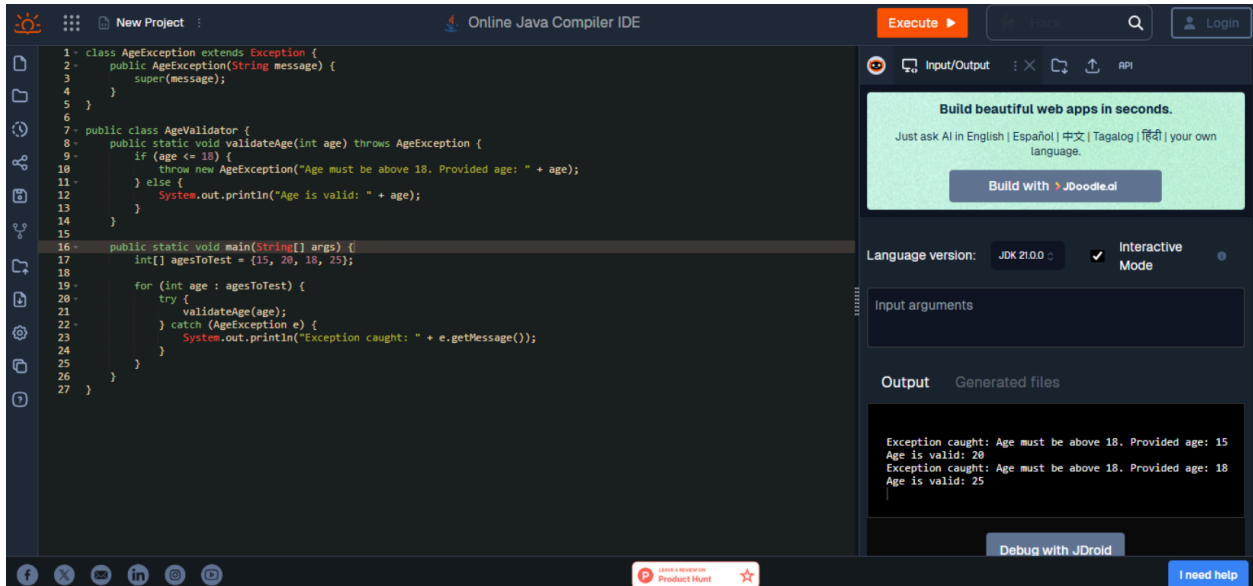


```
1 public class StringMethodsExample {
2     public static void main(String[] args) {
3
4         String originalString = "Hello, welcome to the world of Java programming!";
5
6         String substringExample = originalString.substring(7, 14);
7         System.out.println("Substring (7, 14): " + substringExample);
8
9         String replaceExample = originalString.replace("Java", "Python");
10        System.out.println("After replace(): " + replaceExample);
11
12        String[] splitExample = originalString.split(" ");
13        System.out.println("After split():");
14        for (String word : splitExample) {
15            System.out.println(word);
16        }
17    }
18 }
19 }
```

Substring (7, 14): welcome
After replace(): Hello, welcome to the world of Python programming!
After split():
Hello,
welcome
to
the
world
of
Java
programming!

Compiled and executed in 1.871 sec(s)

Q37. Create a custom exception `AgeException` that checks if a person's age is valid (above 18). in java

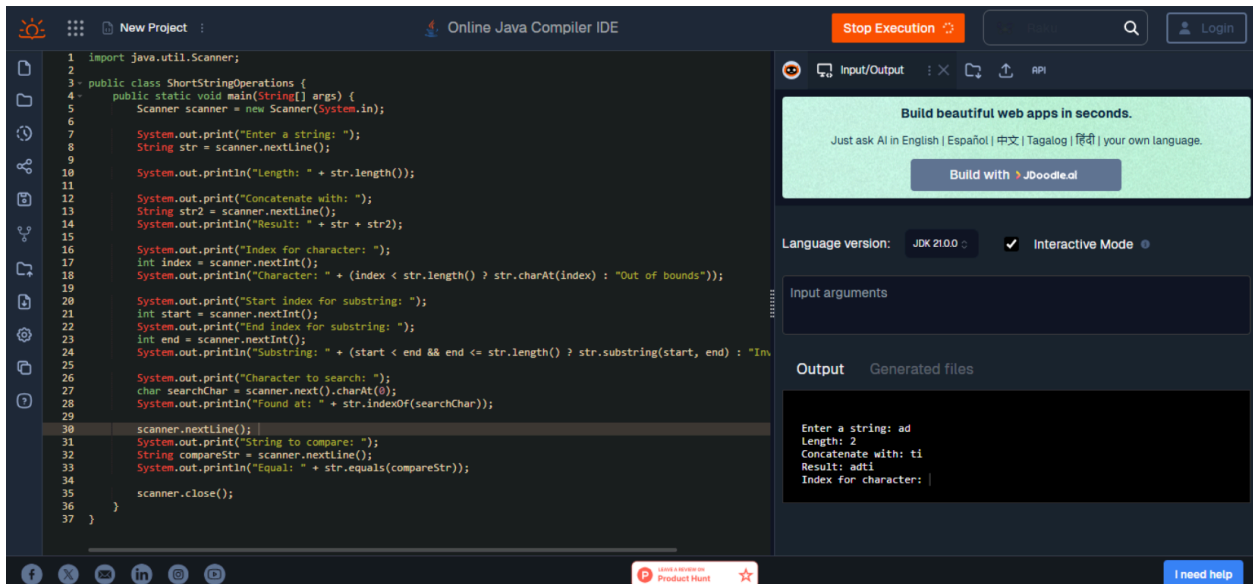


```
1- class AgeException extends Exception {
2-     public AgeException(String message) {
3-         super(message);
4-     }
5- }
6-
7- public class AgeValidator {
8-     public static void validateAge(int age) throws AgeException {
9-         if (age <= 18) {
10-             throw new AgeException("Age must be above 18. Provided age: " + age);
11-         } else {
12-             System.out.println("Age is valid: " + age);
13-         }
14-     }
15-
16-     public static void main(String[] args) {
17-         int[] agesToTest = {15, 20, 18, 25};
18-
19-         for (int age : agesToTest) {
20-             try {
21-                 validateAge(age);
22-             } catch (AgeException e) {
23-                 System.out.println("Exception caught: " + e.getMessage());
24-             }
25-         }
26-     }
27- }
```

Output:

```
Exception caught: Age must be above 18. Provided age: 15
Age is valid: 20
Exception caught: Age must be above 18. Provided age: 18
Age is valid: 25
```

Q38. Create a Java program that demonstrates various string functions and string handling techniques in Java. This program includes common operations like: Length of a string, Concatenation, Character extraction, Substring, Searching, String comparison, Changing case, Trimming, Replacing, Splitting



```
1 import java.util.Scanner;
2
3 public class ShortStringOperations {
4     public static void main(String[] args) {
5         Scanner scanner = new Scanner(System.in);
6
7         System.out.print("Enter a string: ");
8         String str = scanner.nextLine();
9
10        System.out.println("Length: " + str.length());
11
12        System.out.print("Concatenate with: ");
13        String str2 = scanner.nextLine();
14        System.out.println("Result: " + str + str2);
15
16        System.out.print("Index for character: ");
17        int index = scanner.nextInt();
18        System.out.println("Character: " + (index < str.length() ? str.charAt(index) : "Out of bounds"));
19
20        System.out.print("Start index for substring: ");
21        int start = scanner.nextInt();
22        System.out.print("End index for substring: ");
23        int end = scanner.nextInt();
24        System.out.println("Substring: " + (start < end && end <= str.length() ? str.substring(start, end) : "Invalid range"));
25
26        System.out.print("Character to search: ");
27        char searchChar = scanner.next().charAt(0);
28        System.out.println("Found at: " + str.indexOf(searchChar));
29
30        scanner.nextLine();
31        System.out.print("String to compare: ");
32        String compareStr = scanner.nextLine();
33        System.out.println("Equal: " + str.equals(compareStr));
34
35        scanner.close();
36    }
37 }
```

Output:

```
Enter a string: ad
Length: 2
Concatenate with: t1
Result: adt1
Index for character: |
```