C-DAC Mumbai

Date 25/09/2024

Subject: Algorithm and Data Structure
Assignment 1

Solve the assignment with following thing to be added in each question.

-Program
-Flow chart
-Explanation
-Output
-Time and Space complexity

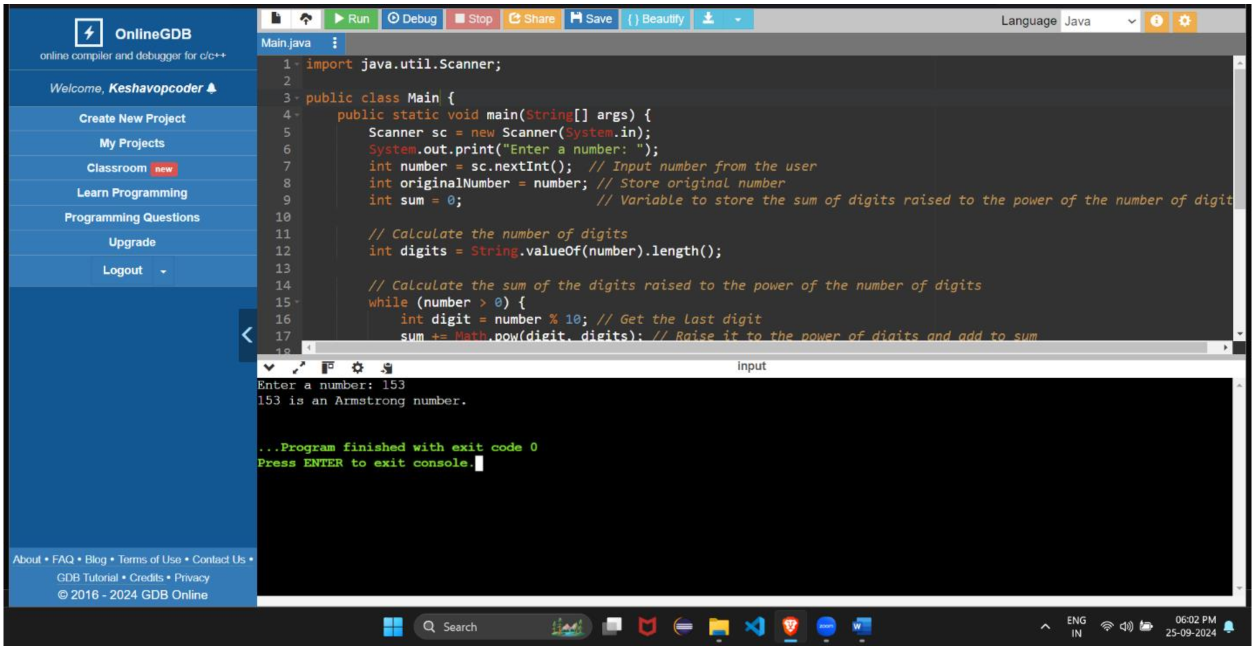
1. Armstrong Number
Problem: Write a Java program to check if a given number is an Armstrong number.

Test Cases:

Input: 153
Output: true
Input: 123
Output: false

Code:

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int number = sc.nextInt();  // Input number from the user
        int originalNumber = number; // Store original number
        int sum = 0;                 // Variable to store the sum of digits raised to the power of the number of
digits

        // Calculate the number of digits
        int digits = String.valueOf(number).length();

        // Calculate the sum of the digits raised to the power of the number of digits
        while (number > 0) {
            int digit = number % 10; // Get the last digit
            sum += Math.pow(digit, digits); // Raise it to the power of digits and add to sum
            number /= 10; // Remove the last digit
        }

        // Check if the original number is equal to the sum
        if (sum == originalNumber) {
            System.out.println(originalNumber + " is an Armstrong number.");
        } else {
            System.out.println(originalNumber + " is not an Armstrong number.");
        }

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



```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int number = sc.nextInt();  // Input number from the user
        int originalNumber = number; // Store original number
        int sum = 0;                // Variable to store the sum of digits raised to the power of the number of digit

        // Calculate the number of digits
        int digits = String.valueOf(number).length();

        // Calculate the sum of the digits raised to the power of the number of digits
        while (number > 0) {
            int digit = number % 10; // Get the last digit
            sum += Math.pow(digit, digits); // Raise it to the power of digits and add to sum
```

```
Enter a number: 153
153 is an Armstrong number.


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

Time Complexity:

- **O(d)**: The time complexity is linear in terms of the number of digits d in the input number since we process each digit once.

Space Complexity:

- **O(1)**: The space complexity is constant as we are using a fixed amount of space regardless of the input size.

```
Start
 |
 V
Input number
 |
 V
Store original number
 |
 V
Count the number of digits
 |
 V
Initialize sum to 0
 |
 V
While number > 0:
 |--> Get last digit
 |--> Add digit^number_of_digits to sum
 |--> Remove last digit
 |
 V
If sum == original number
 |--> Print "is an Armstrong number."
 |
Else
 |--> Print "is not an Armstrong number."
 |
 V
End
```

Explanation:

1. The program prompts the user to enter a number.
2. It stores the original number for comparison later.
3. It calculates the number of digits in the input number.
4. The program then uses a loop to extract each digit, raise it to the power of the number of digits, and accumulate this value into a sum.
5. After exiting the loop, it compares the sum with the original number.
6. If they are equal, it indicates that the number is an Armstrong number; otherwise, it indicates that it is not.

2. Prime Number
Problem: Write a Java program to check if a given number is prime.

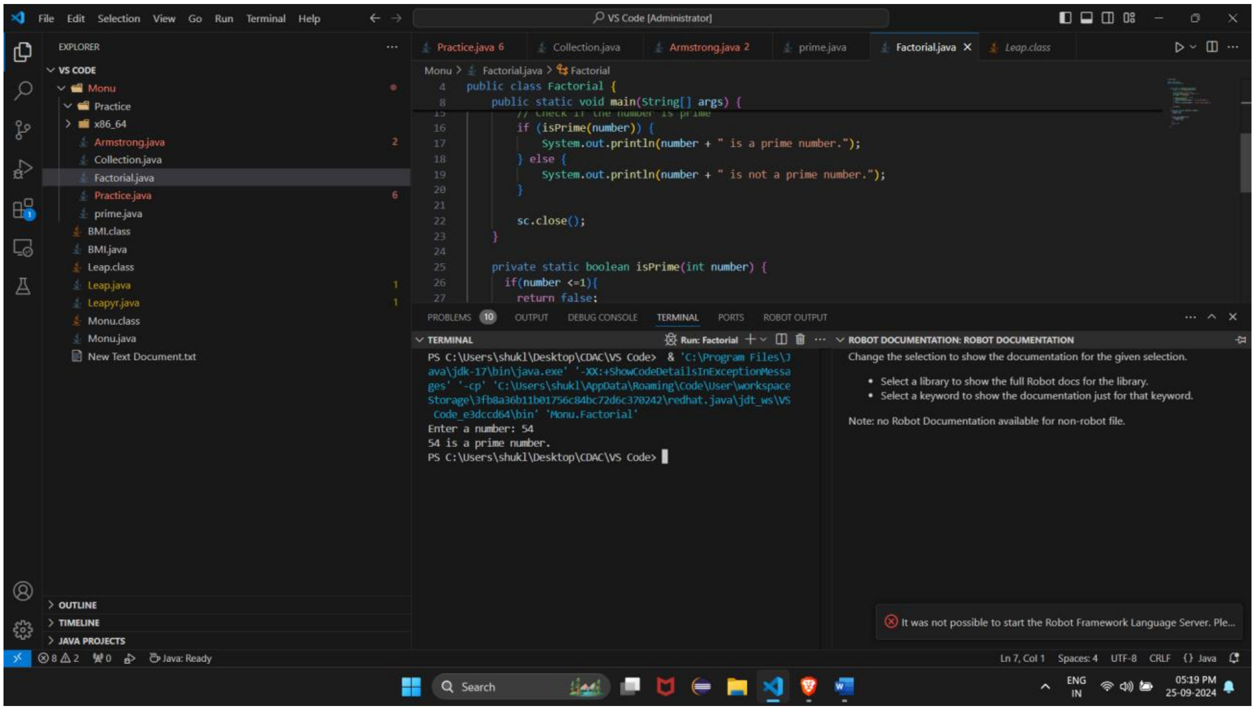
Test Cases:

Input: 29
Output: true
Input: 15
Output: false

Code:

```
package Monu;

import java.util.*;
public class Factorial {
```

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

    // Input number from the user
    System.out.print("Enter a number: ");
    int number = sc.nextInt();

    // Check if the number is prime
    if (isPrime(number)) {
        System.out.println(number + " is a prime number.");
    } else {
        System.out.println(number + " is not a prime number.");
    }

    sc.close();
}

private static boolean isPrime(int number) {
  if(number <=1){
    return false;
  }

  for(int i =2;i<number;i++){
    if(number%i==0){
        return true;
    }
  }
  return true;

}
}
```



```java
public class Factorial {
    public static void main(String[] args) {
        // Check if the number is prime
        if (isPrime(number)) {
            System.out.println(number + " is a prime number.");
        } else {
            System.out.println(number + " is not a prime number.");
        }

        sc.close();
    }

    private static boolean isPrime(int number) {
      if(number <=1){
        return false;
```

```
PS C:\Users\shukl\Desktop\CDAC\VS Code>  & 'C:\Program Files\J
ava\jdk-17\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessa
ges' '-cp' 'C:\Users\shukl\AppData\Roaming\Code\User\workspace
Storage\3fb8a36b11b01756c84bc72d6c370242\redhat.java\jdt_ws\VS
 Code_e3dccd64\bin' 'Monu.Factorial'
Enter a number: 54
54 is a prime number.
PS C:\Users\shukl\Desktop\CDAC\VS Code> 
```

Change the selection to show the documentation for the given selection.

- Select a library to show the full Robot docs for the library.
- Select a keyword to show the documentation just for that keyword.

Note: no Robot Documentation available for non-robot file.

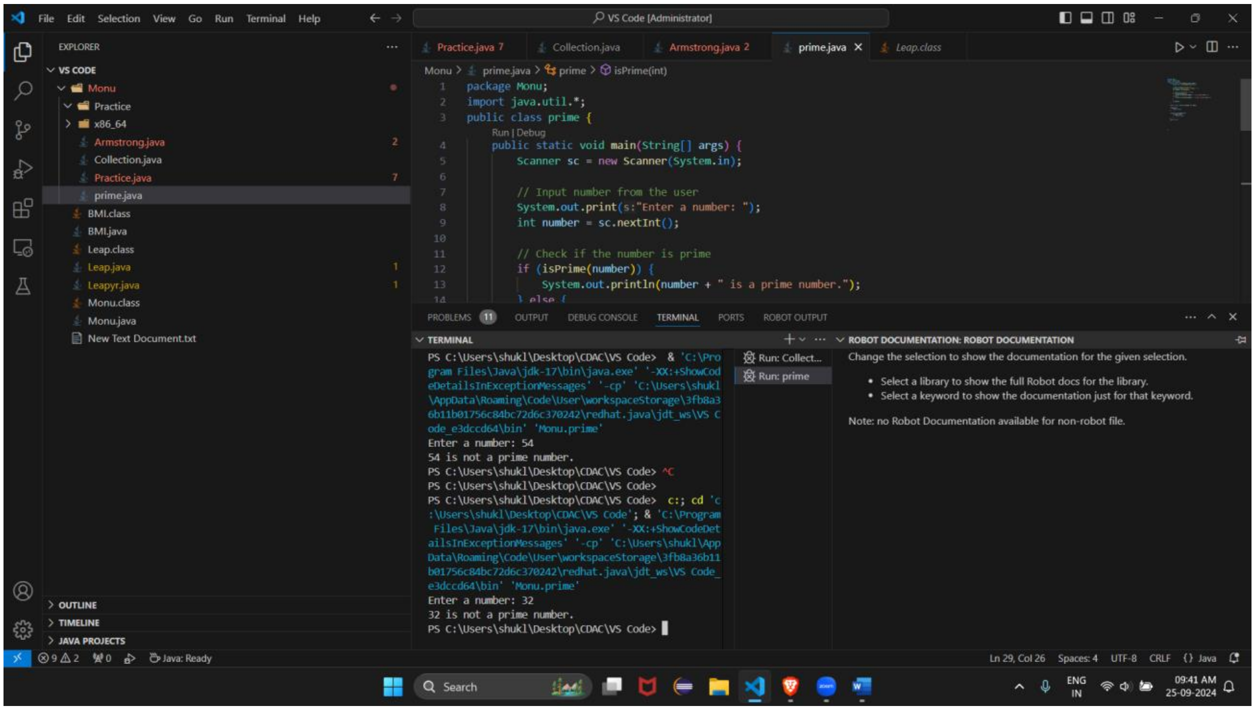
It was not possible to start the Robot Framework Language Server. Ple...

Explanation:

- **Input:** Accepts an integer input from the user.
- **Prime Check Logic:**
  - If the number is less than or equal to 1, it is not prime.
  - For numbers greater than 1, it checks if the number is divisible by any number between 2 and the square root of the number. If a divisor is found, it is not prime.
  - If no divisors are found, the number is prime.

Flowchart:

```
   Start
    ↓
   Input Number
    ↓
   Is Number <= 1?
    /      \
  Yes      No
   ↓         ↓
False    Check divisibility from 2 to √n
            ↓
   Is divisible by any number?
   /       \
 Yes      No
  ↓         ↓
False     True
```



```java
package Monu;
import java.util.*;
public class prime {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        // Input number from the user
        System.out.print(s:"Enter a number: ");
        int number = sc.nextInt();

        // Check if the number is prime
        if (isPrime(number)) {
            System.out.println(number + " is a prime number.");
        } else {
```

```
PS C:\Users\shukl\Desktop\CDAC\VS Code>  & 'C:\Program Files\Java\jdk-17\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\shukl\AppData\Roaming\Code\User\workspaceStorage\3fb8a36b11b01756c84bc72d6c370242\redhat.java\jdt_ws\VS Code_e3dccd64\bin' 'Monu.prime'
Enter a number: 54
54 is not a prime number.
PS C:\Users\shukl\Desktop\CDAC\VS Code> ^C
PS C:\Users\shukl\Desktop\CDAC\VS Code>
PS C:\Users\shukl\Desktop\CDAC\VS Code>  c:; cd 'c:\Users\shukl\Desktop\CDAC\VS Code'; & 'C:\Program Files\Java\jdk-17\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\shukl\AppData\Roaming\Code\User\workspaceStorage\3fb8a36b11b01756c84bc72d6c370242\redhat.java\jdt_ws\VS Code_e3dccd64\bin' 'Monu.prime'
Enter a number: 32
32 is not a prime number.
PS C:\Users\shukl\Desktop\CDAC\VS Code>
```

Change the selection to show the documentation for the given selection.

- Select a library to show the full Robot docs for the library.
- Select a keyword to show the documentation just for that keyword.

Note: no Robot Documentation available for non-robot file.

- **Time Complexity**: O(√n), since we only check divisibility up to the square root of the number.
- **Space Complexity**: O(1), as we only use a constant amount of space.

3. Factorial
Problem: Write a Java program to compute the factorial of a given number.

Test Cases:

Input: 5
Output: 120
Input: 0
Output: 1

Code:
package Monu;

import java.util.*;
public class Factorial {

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

            // Input number
            System.out.print("Enter a number: ");
            int num = sc.nextInt();

            // Compute factorial
            System.out.println("Factorial of " + num + " is: " + factorial(num));
            sc.close();
        }




      public static int factorial(int num) {
        if (num == 0 || num == 1) {
            return 1;
        }
        return num * factorial(num - 1);
    }
}
```

Assignment 1 Solution

---

1. Armstrong Number

Problem: Write a Java program to check if a given number is an Armstrong number.

Code:

java

Copy code

```
import java.util.Scanner;

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

        // Input number
        System.out.print("Enter a number: ");
        int num = sc.nextInt();

        // Check if it's an Armstrong number
        if (isArmstrong(num)) {
            System.out.println(num + " is an Armstrong number.");
        } else {
            System.out.println(num + " is not an Armstrong number.");
        }
        sc.close();
    }
    public static boolean isArmstrong(int num) {
        int sum = 0, temp = num, digits = 0;

        // Find the number of digits
        while (temp != 0) {
            temp /= 10;
            digits++;
        }

        temp = num;

        // Calculate the sum of the powers of digits
        while (temp != 0) {
            int digit = temp % 10;
            sum += Math.pow(digit, digits);
            temp /= 10;
        }

        return sum == num;
    }
}
```

Explanation:

- An Armstrong number for a given number of digits is a number that is equal to the sum of its digits each raised to the power of the number of digits.
- Example: 153 = 1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153.

Flowchart:

mathematica

Copy code

```
    Start
      ↓
    Input number
      ↓
    Count digits
      ↓
  Calculate sum of each digit to the power of number of digits
      ↓
   Is sum == original number?
   /     \
  Yes    No
  ↓       ↓
 Armstrong Not Armstrong
```

Test Cases:

Input	**Output**
153	true (Armstrong)
123	false (Not Armstrong)

Output:

csharp
Copy code

```
Enter a number: 153
153 is an Armstrong number.
```

Time and Space Complexity:

- **Time Complexity:** O(d), where d is the number of digits in the number.
- **Space Complexity:** O(1).

---

2. Prime Number

Refer to the earlier provided solution for prime number checking.

---

3. Factorial

Problem: Write a Java program to compute the factorial of a given number.

Code:

java
Copy code

```java
import java.util.Scanner;

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

        // Input number
        System.out.print("Enter a number: ");
        int num = sc.nextInt();

        // Compute factorial
        System.out.println("Factorial of " + num + " is: " + factorial(num));
        sc.close();
    }

    public static int factorial(int num) {
        if (num == 0 || num == 1) {
            return 1;
        }
        return num * factorial(num - 1);
    }
}
```

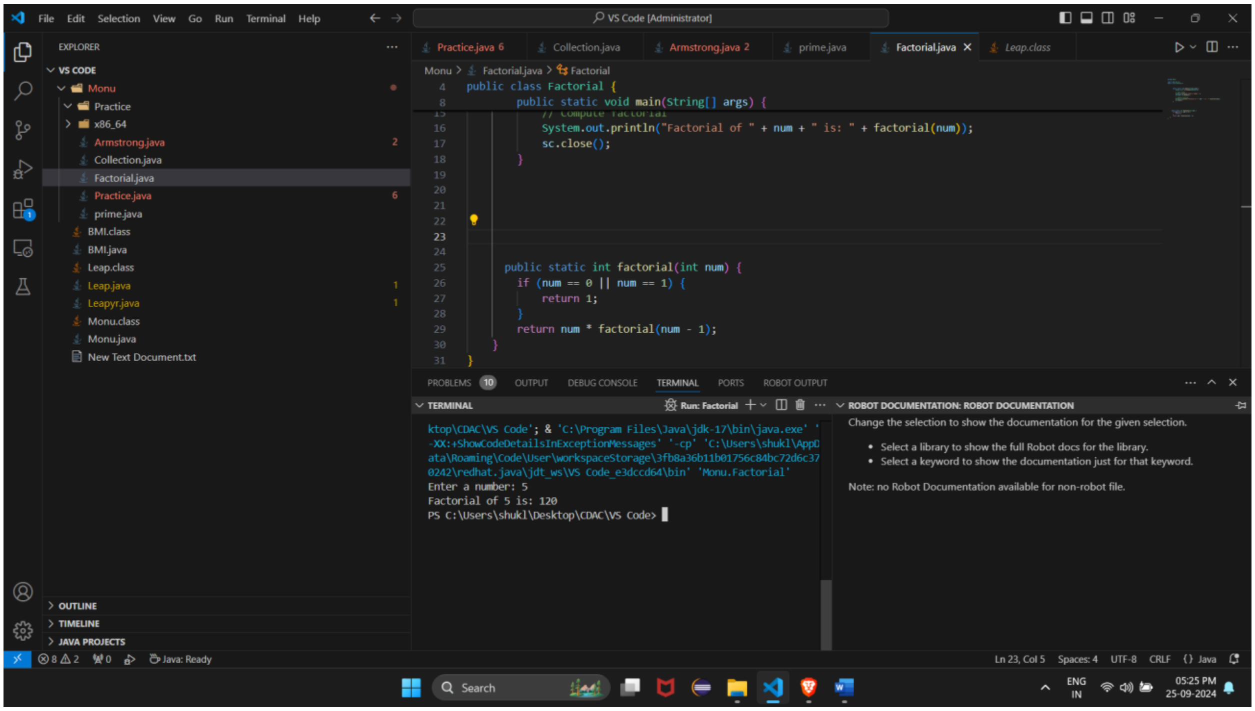
Explanation:

- The factorial of a number is the product of all integers from 1 to that number.
- Example: 5! = 5 × 4 × 3 × 2 × 1 = 120.

Flow:

```
    Start
      ↓
    Input number
      ↓
    Is number == 0 or 1?
```

```
 /      \
Yes      No
↓         ↓
1    Compute factorial recursively
   ↓
 Output factorial
```



```java
public class Factorial {
        public static void main(String[] args) {
            // Compute factorial
            System.out.println("Factorial of " + num + " is: " + factorial(num));
            sc.close();
        }



    public static int factorial(int num) {
      if (num == 0 || num == 1) {
          return 1;
      }
      return num * factorial(num - 1);
    }
}
```

```
ktop\CDAC\VS Code'; & 'C:\Program Files\Java\jdk-17\bin\java.exe' '
-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\shukl\AppD
ata\Roaming\Code\User\workspaceStorage\3fb8a36b11b01756c84bc72d6c37
0242\redhat.java\jdt_ws\VS Code_e3dccd64\bin' 'Monu.Factorial'
Enter a number: 5
Factorial of 5 is: 120
PS C:\Users\shukl\Desktop\CDAC\VS Code> 
```

Change the selection to show the documentation for the given selection.

- Select a library to show the full Robot docs for the library.
- Select a keyword to show the documentation just for that keyword.

Note: no Robot Documentation available for non-robot file.

4. Fibonacci Series
Problem: Write a Java program to print the first n numbers in the Fibonacci series.

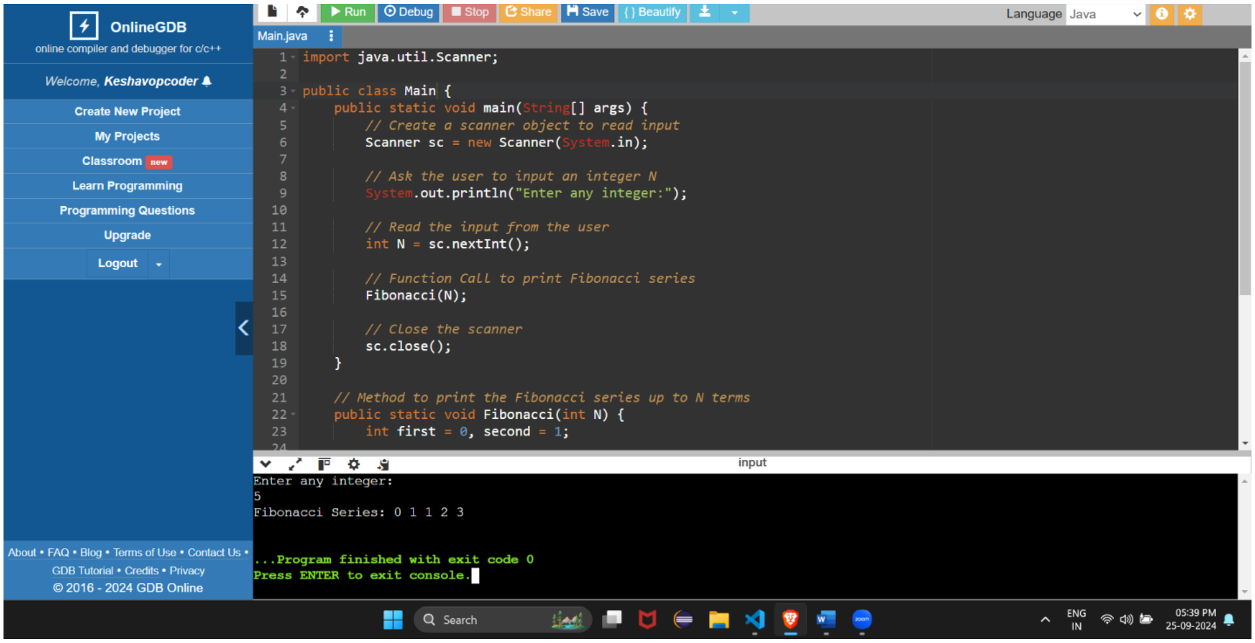
Test Cases:

Input: n = 5
Output: [0, 1, 1, 2, 3]
Input: n = 8
Output: [0, 1, 1, 2, 3, 5, 8, 13]

Code: import java.util.Scanner;

```
public class Main {
    public static void main(String[] args) {
        // Create a scanner object to read input
        Scanner sc = new Scanner(System.in);

        // Ask the user to input an integer N
        System.out.println("Enter any integer:");

        // Read the input from the user
        int N = sc.nextInt();

        // Function Call to print Fibonacci series
        Fibonacci(N);

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

    // Method to print the Fibonacci series up to N terms
    public static void Fibonacci(int N) {
        int first = 0, second = 1;

        // Print the first two Fibonacci numbers
        System.out.print("Fibonacci Series: " + first + " " + second);

        // Loop to generate the rest of the series
        for (int i = 2; i < N; i++) {
            int next = first + second;
            System.out.print(" " + next);
            first = second;
            second = next;
        }
        System.out.println(); // Move to the next line after printing
    }
}
```

Time Complexity: *O(N)*
Auxiliary Space: *O(1)*



```java
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        // Create a scanner object to read input
        Scanner sc = new Scanner(System.in);

        // Ask the user to input an integer N
        System.out.println("Enter any integer:");

        // Read the input from the user
        int N = sc.nextInt();

        // Function Call to print Fibonacci series
        Fibonacci(N);

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

    // Method to print the Fibonacci series up to N terms
    public static void Fibonacci(int N) {
        int first = 0, second = 1;
```

```
Enter any integer:
5
Fibonacci Series: 0 1 1 2 3


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

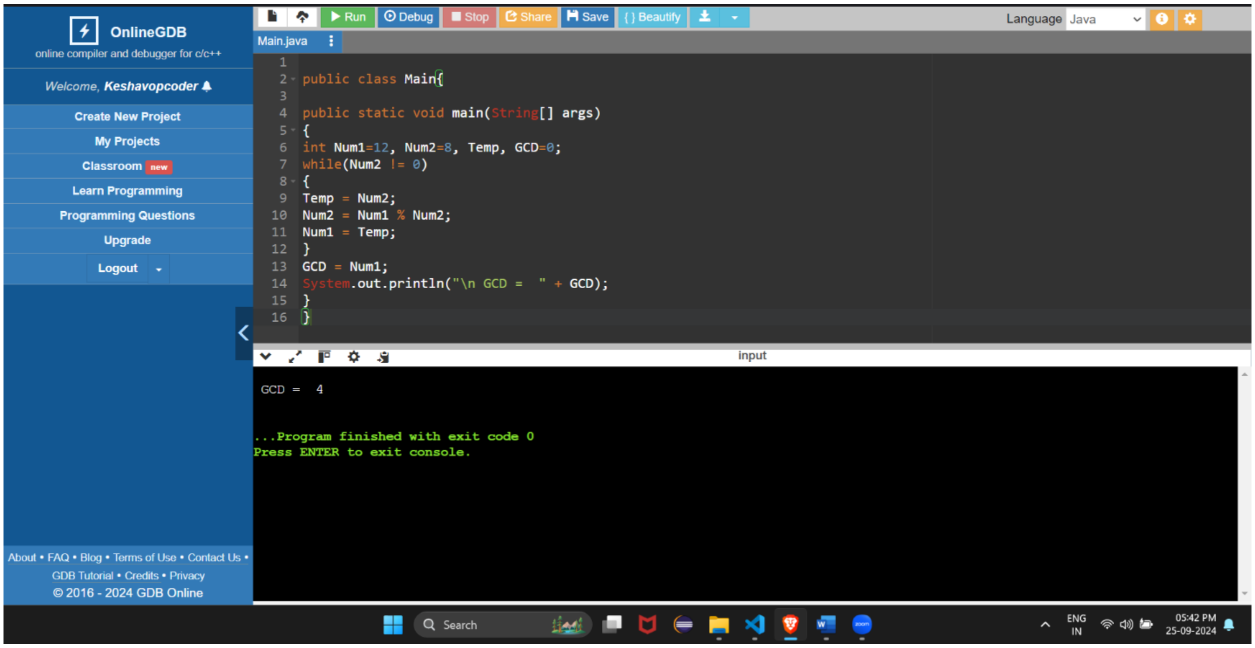
5. Find GCD
Problem: Write a Java program to find the Greatest Common Divisor (GCD) of two numbers.

Test Cases:

Input: a = 54, b = 24
Output: 6
Input: a = 17, b = 13
Output: 1

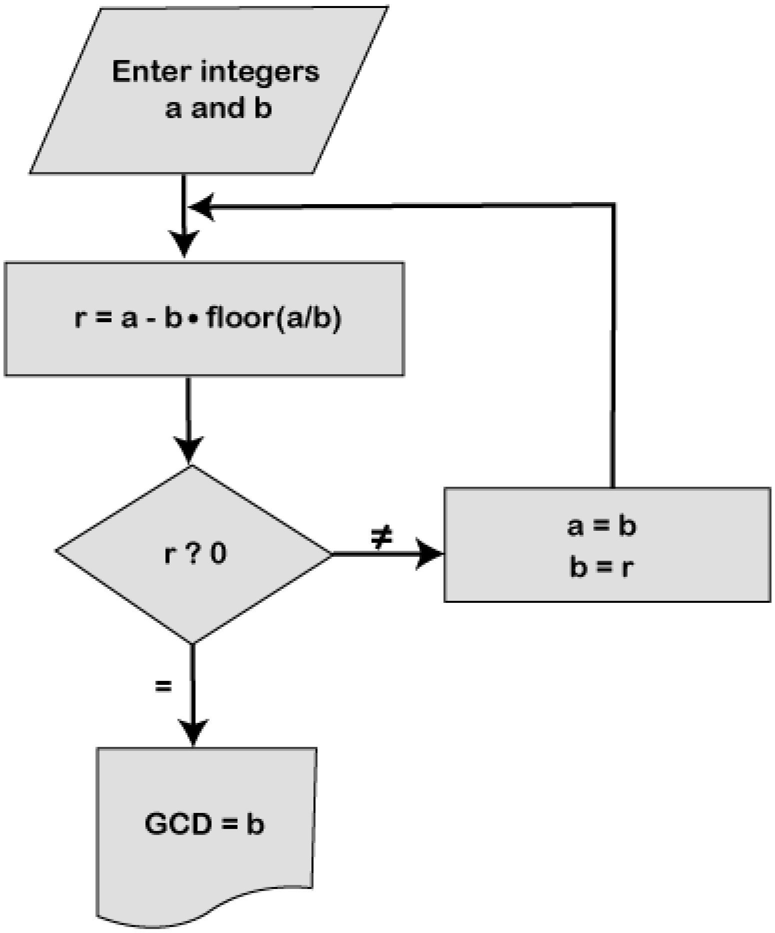
Code:

```
public class Main{

public static void main(String[] args)
{
int Num1=12, Num2=8, Temp, GCD=0;
while(Num2 != 0)
{
Temp = Num2;
Num2 = Num1 % Num2;
Num1 = Temp;
}
GCD = Num1;
System.out.println("\n GCD =  " + GCD);
}
}
```



```java

public class Main{

public static void main(String[] args)
{
int Num1=12, Num2=8, Temp, GCD=0;
while(Num2 != 0)
{
Temp = Num2;
Num2 = Num1 % Num2;
Num1 = Temp;
}
GCD = Num1;
System.out.println("\n GCD =  " + GCD);
}
}
```

```
GCD =  4


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



6. Find Square Root

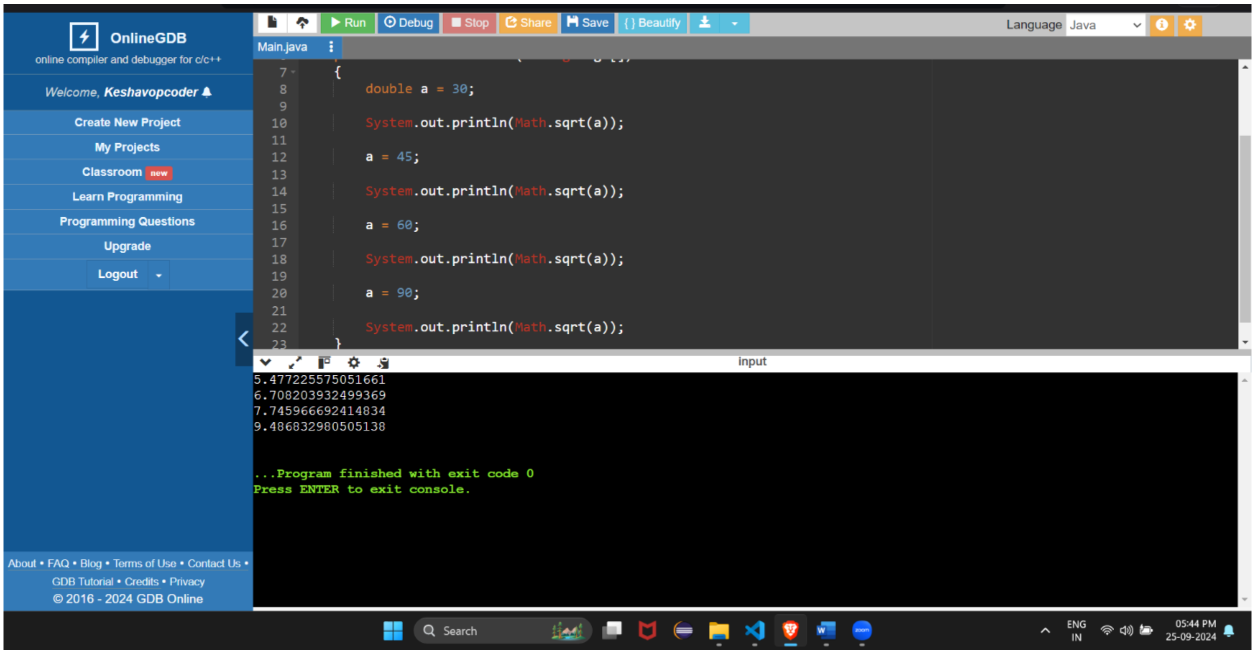
Problem: Write a Java program to find the square root of a given number (using integer approximation).

Test Cases:

Input: x = 16
Output: 4
Input: x = 27
Output: 5

Code:

```
import java.lang.Math;

class Main {

    // driver code
    public static void main(String args[])
    {
        double a = 30;

        System.out.println(Math.sqrt(a));

        a = 45;

        System.out.println(Math.sqrt(a));

        a = 60;

        System.out.println(Math.sqrt(a));

        a = 90;

        System.out.println(Math.sqrt(a));
    }
}
```



```
    {
        double a = 30;

        System.out.println(Math.sqrt(a));

        a = 45;

        System.out.println(Math.sqrt(a));

        a = 60;

        System.out.println(Math.sqrt(a));

        a = 90;

        System.out.println(Math.sqrt(a));
    }
```

```
5.477225575051661
6.708203932499369
7.745966692414834
9.486832980505138


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

7. Find Repeated Characters in a String

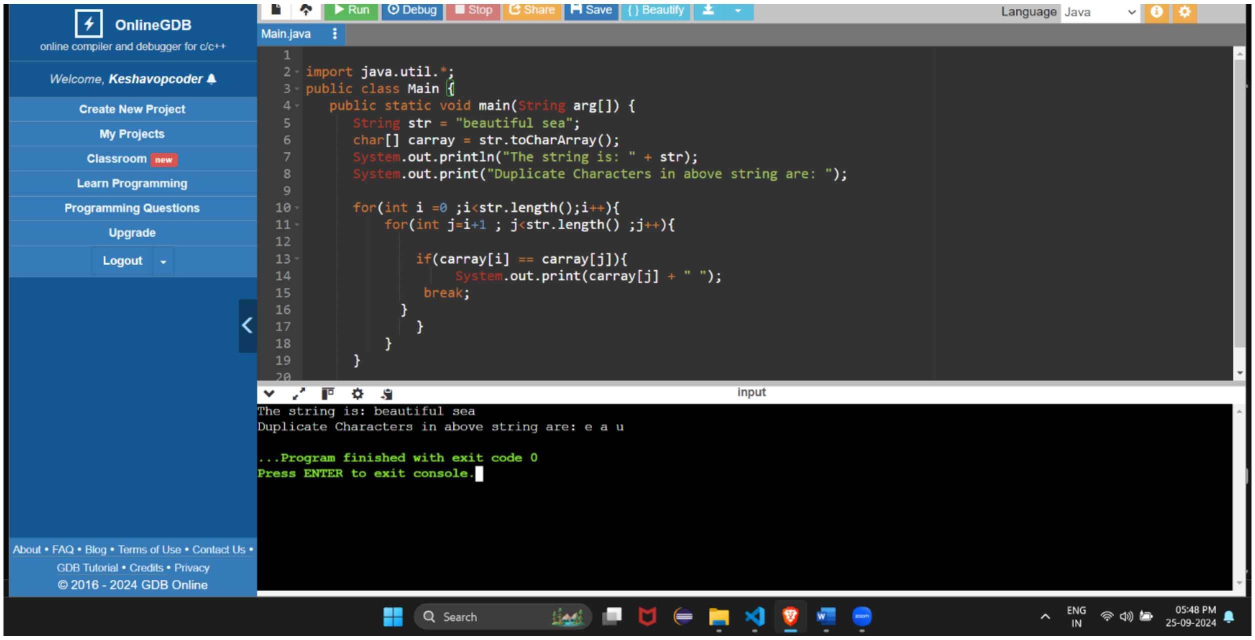
Problem: Write a Java program to find all repeated characters in a string.

Test Cases:

Input: "programming"
Output: ['r', 'g', 'm']
Input: "hello"
Output: ['l']

Code:

```
import java.util.*;
public class Main {
  public static void main(String arg[]) {
    String str = "beautiful sea";
    char[] carray = str.toCharArray();
    System.out.println("The string is: " + str);
    System.out.print("Duplicate Characters in above string are: ");

    for(int i =0 ;i<str.length();i++){
       for(int j=i+1 ; j<str.length() ;j++){

          if(carray[i] == carray[j]){
              System.out.print(carray[j] + " ");
           break;
         }
           }
       }
    }

}
```



```

import java.util.*;
public class Main {
  public static void main(String arg[]) {
    String str = "beautiful sea";
    char[] carray = str.toCharArray();
    System.out.println("The string is: " + str);
    System.out.print("Duplicate Characters in above string are: ");

    for(int i =0 ;i<str.length();i++){
        for(int j=i+1 ; j<str.length() ;j++){

            if(carray[i] == carray[j]){
                System.out.print(carray[j] + " ");
             break;
            }
              }
          }
    }
```

```
The string is: beautiful sea
Duplicate Characters in above string are: e a u

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

- Convert the given string into character array.
- Run the outer for loop from **index 0** to length of character array.
- The inner for loop will run from **current+1** index to length of the character array.
- Next, we use **if block** to check whether current character is equal to the next character.
- If found equal, print the duplicate character.

The time complexity of this algorithm is **O(n^2)**
The space complexity is **O(n)**

```
        Start
          ↓
    Input String "beautiful sea"
          ↓
    Convert String to char[] array
          ↓
    Outer loop: i = 0 to n-1 (length of string)
          ↓
    Inner loop: j = i+1 to n
          ↓
    Compare array[i] == array[j]
       ↓Yes          ↓No
   Print array[j]    Move to next character
          ↓
    Continue until all characters checked
          ↓
         End
```

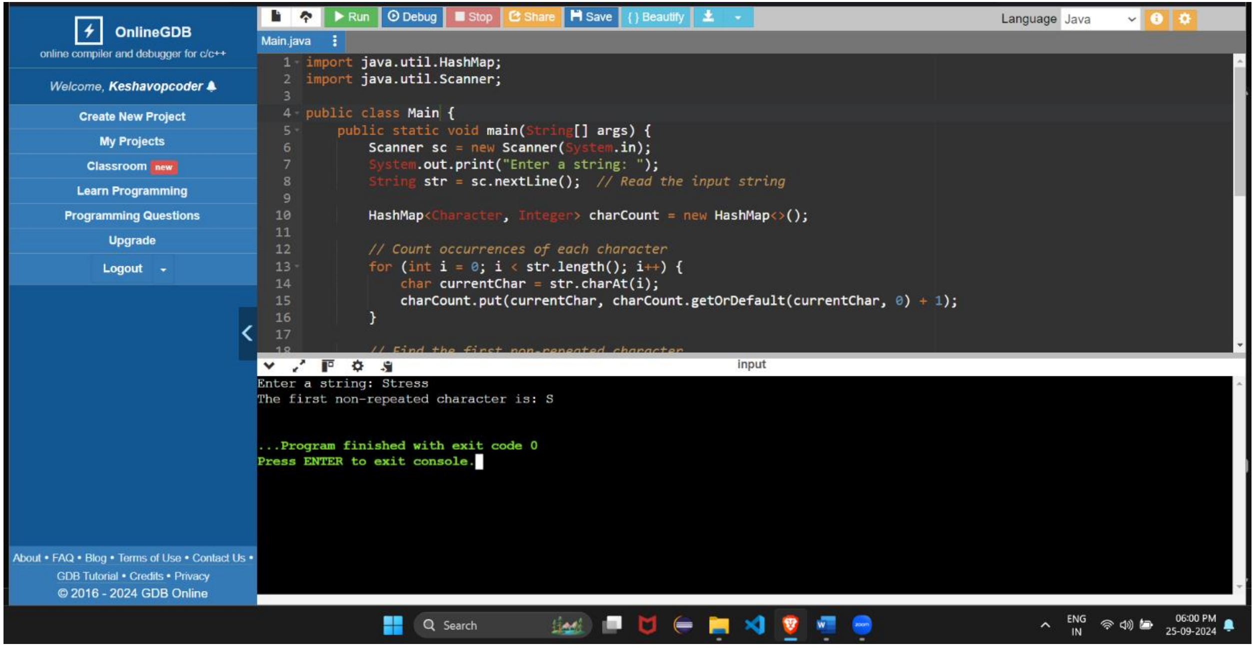
8. First Non-Repeated Character
Problem: Write a Java program to find the first non-repeated character in a string.

Test Cases:

Input: "stress"
Output: 't'
Input: "aabbcc"
Output: null

Code:

```
import java.util.HashMap;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a string: ");
        String str = sc.nextLine();  // Read the input string

        HashMap<Character, Integer> charCount = new HashMap<>();

        // Count occurrences of each character
        for (int i = 0; i < str.length(); i++) {
            char currentChar = str.charAt(i);
            charCount.put(currentChar, charCount.getOrDefault(currentChar, 0) + 1);
        }

        // Find the first non-repeated character
        char firstNonRepeated = '\0';  // Default value to indicate no non-repeated character found
        for (int i = 0; i < str.length(); i++) {
            if (charCount.get(str.charAt(i)) == 1) {
                firstNonRepeated = str.charAt(i);
                break;  // Stop at the first non-repeated character
            }
        }

        // Output the result
        if (firstNonRepeated != '\0') {
            System.out.println("The first non-repeated character is: " + firstNonRepeated);
        } else {
            System.out.println("No non-repeated characters found.");
        }

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



```
import java.util.HashMap;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a string: ");
        String str = sc.nextLine();  // Read the input string

        HashMap<Character, Integer> charCount = new HashMap<>();

        // Count occurrences of each character
        for (int i = 0; i < str.length(); i++) {
            char currentChar = str.charAt(i);
            charCount.put(currentChar, charCount.getOrDefault(currentChar, 0) + 1);
        }

        // Find the first non-repeated character
```

```
Enter a string: Stress
The first non-repeated character is: S


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

```
Start
  |
  V
Input String (str)
  |
  V
Create a HashMap to store character counts
  |
  V
```

```
For each character in the string:
  |--> Increment count in HashMap
  |
  V
For each character in the string:
  |--> If count == 1, return as first non-repeated character
  |
  V
If no non-repeated character found, return null
  |
  V
End
```

Time Complexity:

- **O(n)**:

Space Complexity:

- **O(k)**:

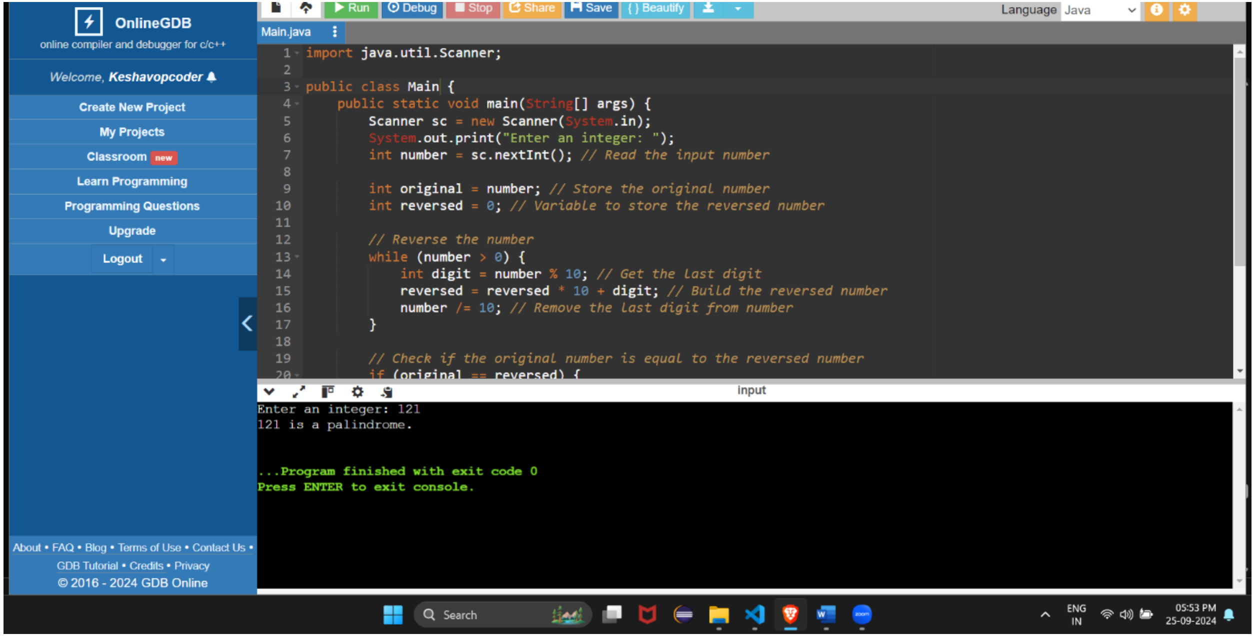
9. Integer Palindrome
Problem: Write a Java program to check if a given integer is a palindrome.

Test Cases:

Input: 121
Output: true
Input: -121
Output: false

Code:

```
import java.util.Scanner;

public class PalindromeCheck {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter an integer: ");
        int number = sc.nextInt(); // Read the input number

        int original = number; // Store the original number
        int reversed = 0; // Variable to store the reversed number

        // Reverse the number
        while (number > 0) {
            int digit = number % 10; // Get the last digit
            reversed = reversed * 10 + digit; // Build the reversed number
            number /= 10; // Remove the last digit from number
        }

        // Check if the original number is equal to the reversed number
        if (original == reversed) {
            System.out.println(original + " is a palindrome.");
        } else {
            System.out.println(original + " is not a palindrome.");
        }

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



```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter an integer: ");
        int number = sc.nextInt(); // Read the input number

        int original = number; // Store the original number
        int reversed = 0; // Variable to store the reversed number

        // Reverse the number
        while (number > 0) {
            int digit = number % 10; // Get the last digit
            reversed = reversed * 10 + digit; // Build the reversed number
            number /= 10; // Remove the last digit from number
        }

        // Check if the original number is equal to the reversed number
        if (original == reversed) {
```

```
Enter an integer: 121
121 is a palindrome.


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

O(d) where d is the number of digits in the number.
O(1) since it uses a constant amount of space.

Explanation:

1. **Input**: The program prompts the user to enter an integer.
2. **Reversal Logic**: It uses a while loop to reverse the number by extracting the last digit and building the reversed number.
3. **Comparison**: After reversing, it compares the original number with the reversed number.
4. **Output**: The program outputs whether the number is a palindrome or not.

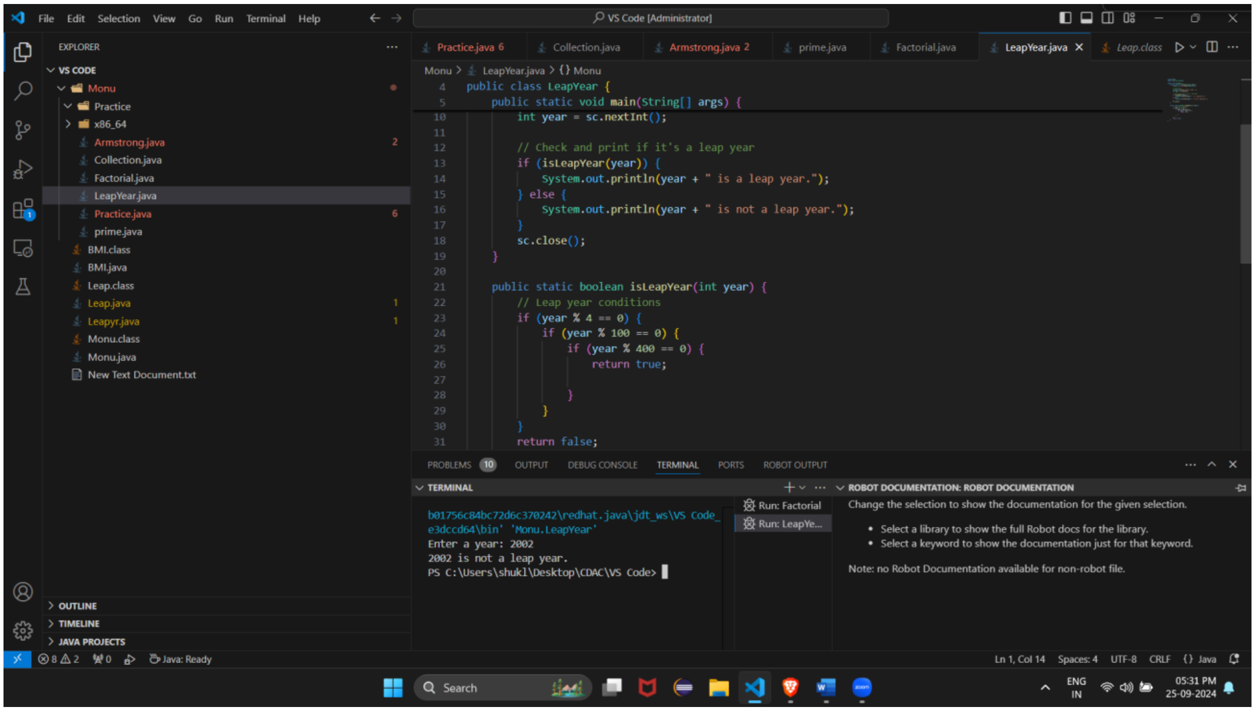
10. Leap Year
Problem: Write a Java program to check if a given year is a leap year.

Test Cases:

Input: 2020
Output: true
Input: 1900
Output: false

Code:

```
import java.util.Scanner;



public class LeapYear {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);



        // Input year

        System.out.print("Enter a year: ");

        int year = sc.nextInt();
        // Check and print if it's a leap year
        if (isLeapYear(year)) {
            System.out.println(year + " is a leap year.");
        } else {
            System.out.println(year + " is not a leap year.");
        }
        sc.close();
    }


    public static boolean isLeapYear(int year) {
        // Leap year conditions
        if (year % 4 == 0) {
            if (year % 100 == 0) {
                if (year % 400 == 0) {
                    return true
```



```java
public class LeapYear {
    public static void main(String[] args) {
        int year = sc.nextInt();

        // Check and print if it's a leap year
        if (isLeapYear(year)) {
            System.out.println(year + " is a leap year.");
        } else {
            System.out.println(year + " is not a leap year.");
        }
        sc.close();
    }

    public static boolean isLeapYear(int year) {
        // Leap year conditions
        if (year % 4 == 0) {
            if (year % 100 == 0) {
                if (year % 400 == 0) {
                    return true;

                }
            }
        }
        return false;
```

```
b01756c84bc72d6c370242\redhat.java\jdt_ws\VS Code_
e3dccd64\bin' 'Monu.LeapYear'
Enter a year: 2002
2002 is not a leap year.
PS C:\Users\shukl\Desktop\CDAC\VS Code> 
```

Change the selection to show the documentation for the given selection.

- Select a library to show the full Robot docs for the library.
- Select a keyword to show the documentation just for that keyword.

Note: no Robot Documentation available for non-robot file.