1. Create a functional Interface with a function operation(). Write a Program to do basic Math Operation (add, sub, multiply, divide) using Lambda Expression (each object should perform each operation)

Program: -

```java
import java.util.function.BiFunction;

/**
 * Java 8 program to perform arithmetic operation of two numbers using inbuilt BiFunction Functional Interface
 *
 * @author D.V
 */
public class Arithmetic_Operation_Java8_Function_Example {

    public static void main(String[] args) {

        //Lambda expression for addition
        BiFunction <Integer, Integer, Integer> funcAddObj = (i1, i2) -> i1 + i2;

        //Lambda expression for subtract
        BiFunction <Integer, Integer, Integer> funcSubtractObj = (i1, i2) -> i1 - i2;

        //Lambda expression for multiply
        BiFunction <Integer, Integer, Integer> funcMuliplyObj = (i1, i2) -> i1 * i2;

        //Lambda expression for division
        BiFunction <Integer, Integer, Integer> funcDivideObj = (i1, i2) -> i1 / i2;

        //Lambda expression for division
        BiFunction <Integer, Integer, Integer> funcModuloObj = (i1, i2) -> i1 % i2;

        System.out.println("Addition of 10 and 5: " + funcAddObj.apply(10, 5));

        System.out.println("Subtract of 10 and 5: " + funcSubtractObj.apply(10, 5));

        System.out.println("Multiply of 10 and 5: " + funcMuliplyObj.apply(10, 5));

        System.out.println("Division of 10 and 5: " + funcDivideObj.apply(10, 5));

        System.out.println("Modulo of 10 and 5: " + funcModuloObj.apply(10, 5));

    }

}
```

Output :-

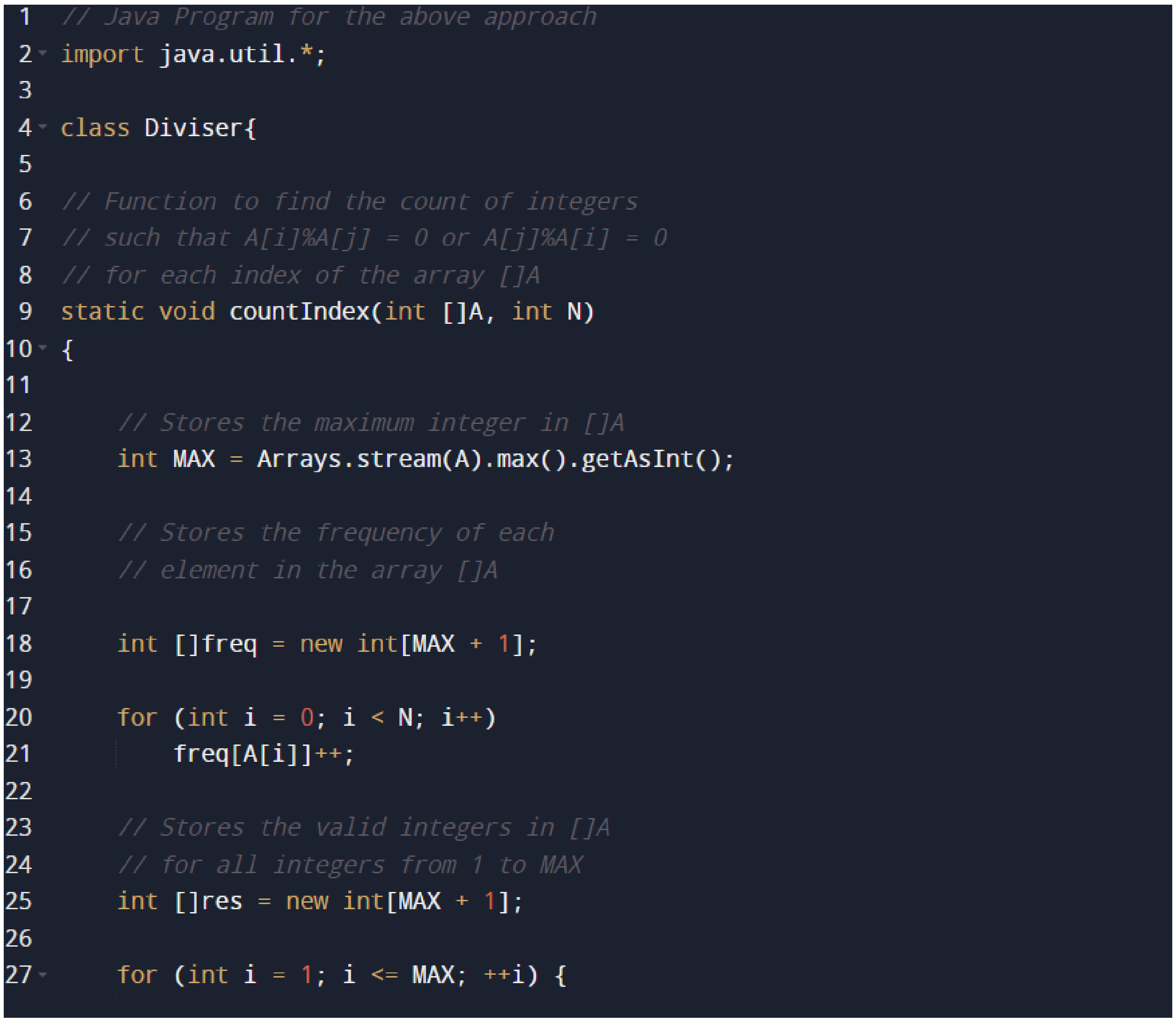
```
Addition of 10 and 5: 15
Subtract of 10 and 5: 5
Multiply of 10 and 5: 50
Division of 10 and 5: 2
Modulo of 10 and 5: 0
```

2. Create a function- find Power (int num) in a class which prints the power of numbers from 1 to 6. Create 3 threads that calls the function in a synchronized as well as non-synchronized way.

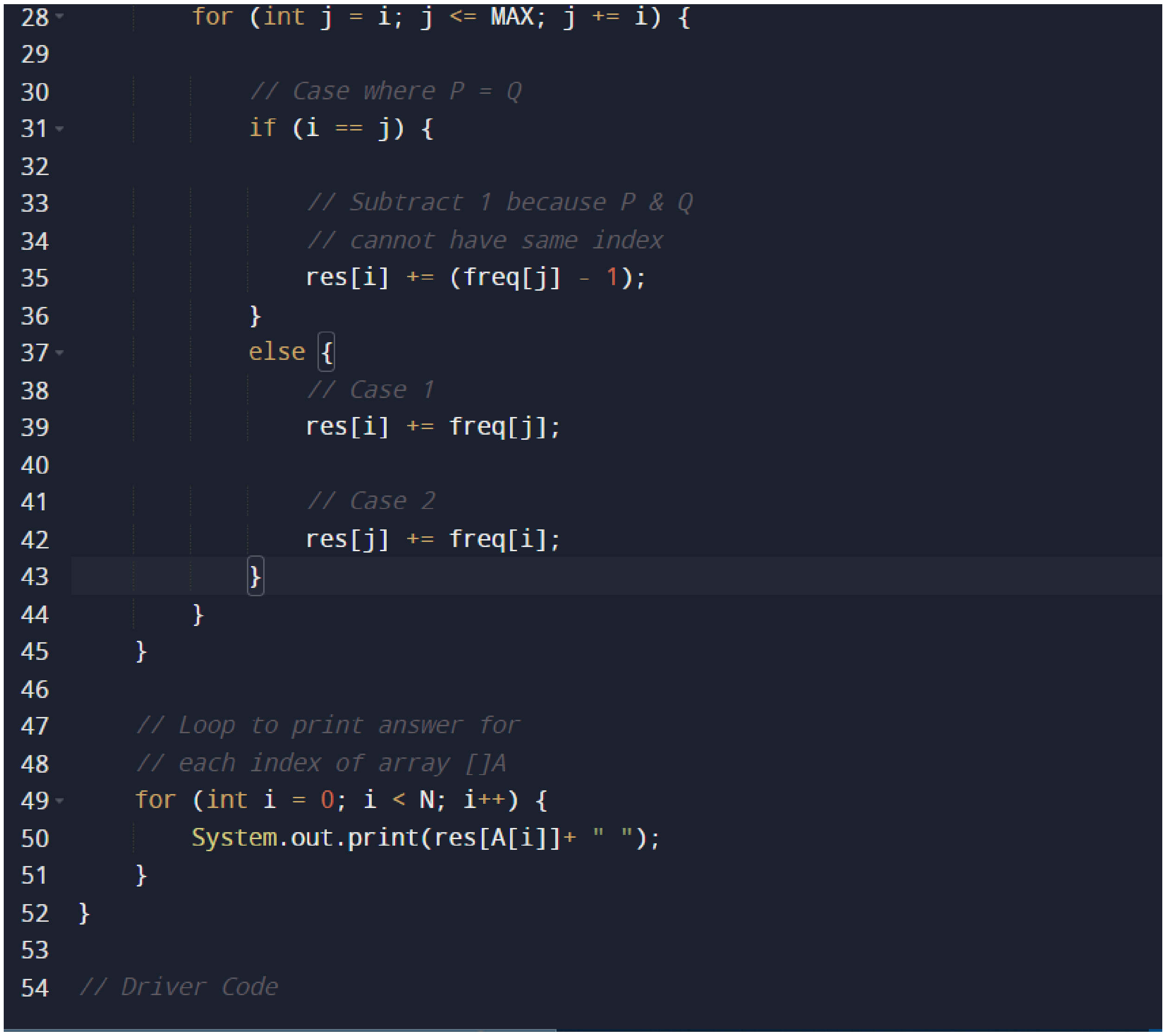
3. Perform Exception Handling for this scenario wherever it is required.

Input an array of numbers and a divisor.

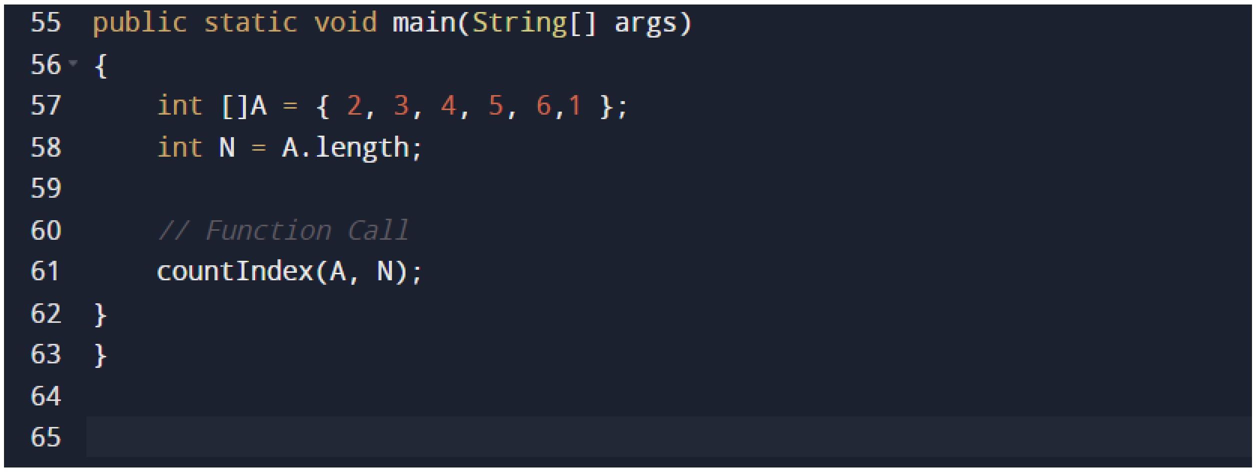
Program: -



```java
// Java Program for the above approach
import java.util.*;

class Diviser{

// Function to find the count of integers
// such that A[i]%A[j] = 0 or A[j]%A[i] = 0
// for each index of the array []A
static void countIndex(int []A, int N)
{

    // Stores the maximum integer in []A
    int MAX = Arrays.stream(A).max().getAsInt();

    // Stores the frequency of each
    // element in the array []A

    int []freq = new int[MAX + 1];

    for (int i = 0; i < N; i++)
        freq[A[i]]++;

    // Stores the valid integers in []A
    // for all integers from 1 to MAX
    int []res = new int[MAX + 1];

    for (int i = 1; i <= MAX; ++i) {
```

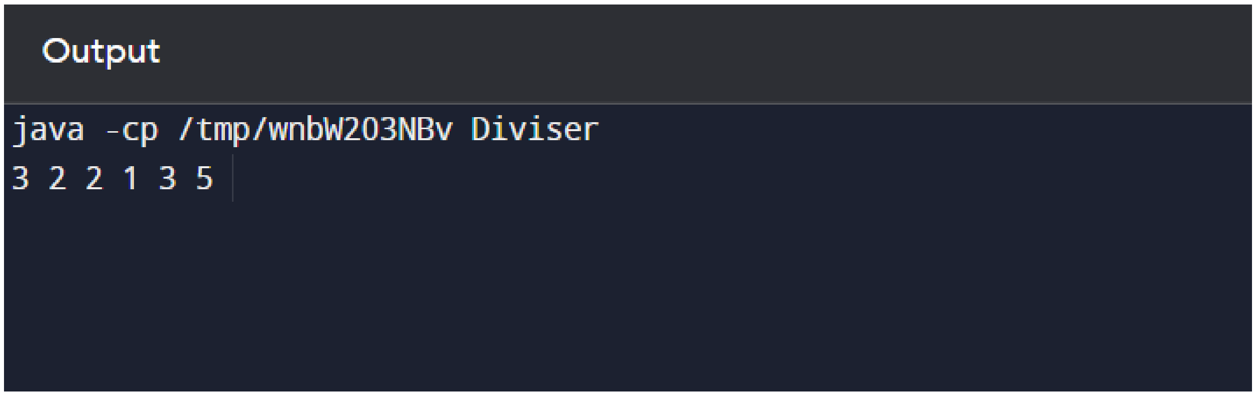


```java
        for (int j = i; j <= MAX; j += i) {

            // Case where P = Q
            if (i == j) {

                // Subtract 1 because P & Q
                // cannot have same index
                res[i] += (freq[j] - 1);
            }
            else {
                // Case 1
                res[i] += freq[j];

                // Case 2
                res[j] += freq[i];
            }
        }
    }

    // Loop to print answer for
    // each index of array []A
    for (int i = 0; i < N; i++) {
        System.out.print(res[A[i]]+ " ");
    }
}

// Driver Code
```



```java
public static void main(String[] args)
{
    int []A = { 2, 3, 4, 5, 6,1 };
    int N = A.length;

    // Function Call
    countIndex(A, N);
}
}


```

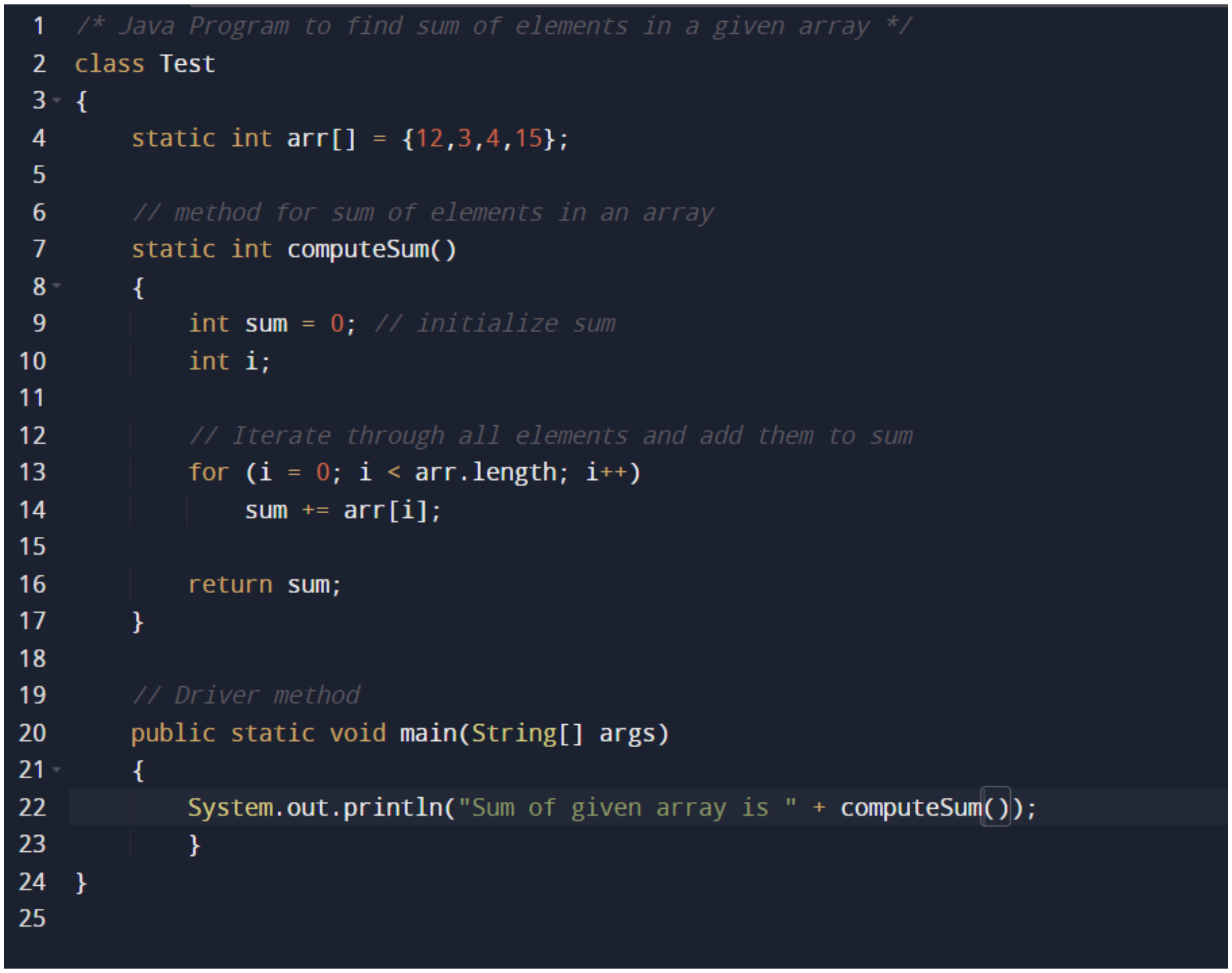
Output :-



```
java -cp /tmp/wnbW203NBv Diviser
3 2 2 1 3 5
```

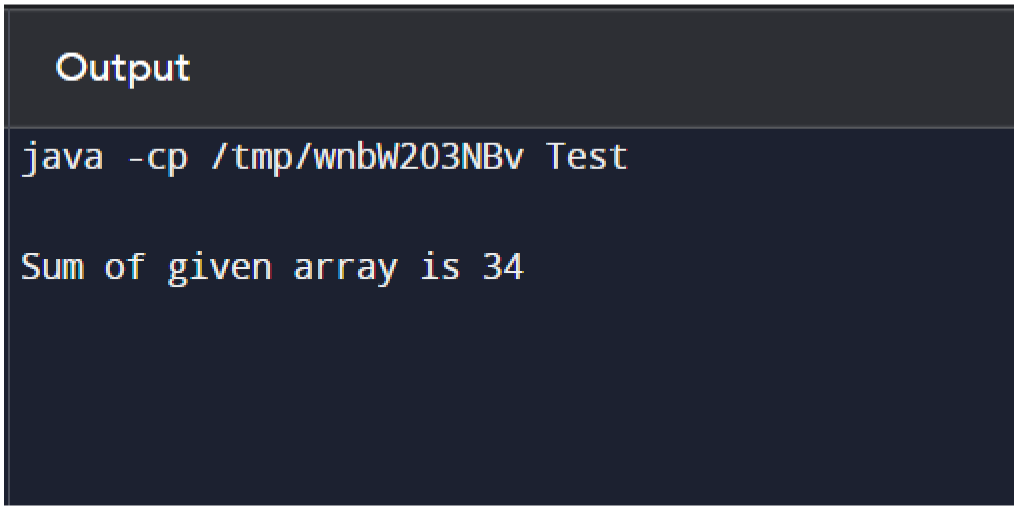
computeSum() function should calculate the sum of arrays.

Program: -



```java
/* Java Program to find sum of elements in a given array */
class Test
{
    static int arr[] = {12,3,4,15};

    // method for sum of elements in an array
    static int computeSum()
    {
        int sum = 0; // initialize sum
        int i;

        // Iterate through all elements and add them to sum
        for (i = 0; i < arr.length; i++)
            sum += arr[i];

        return sum;
    }

    // Driver method
    public static void main(String[] args)
    {
        System.out.println("Sum of given array is " + computeSum());
        }
}
```

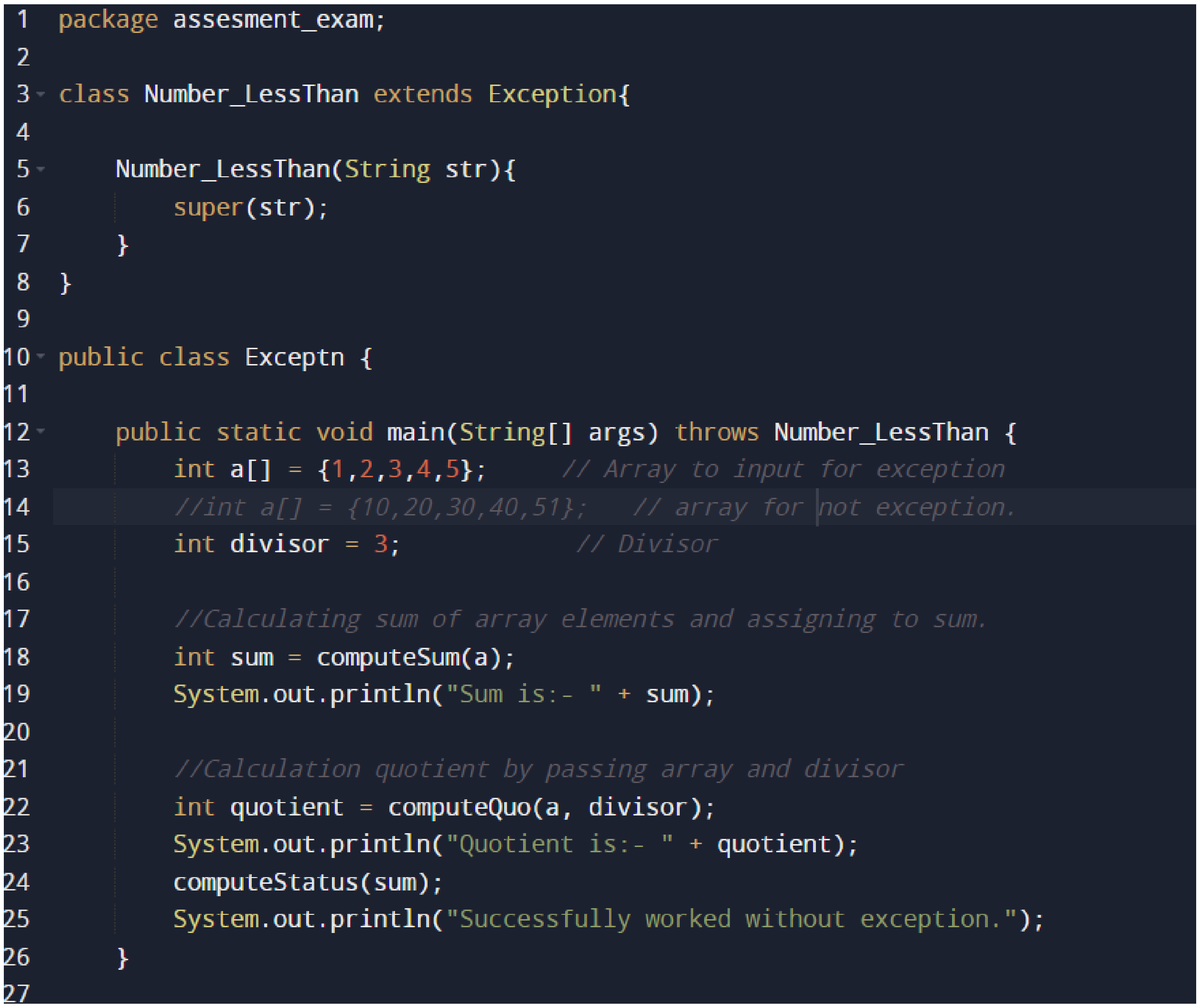
Output :-



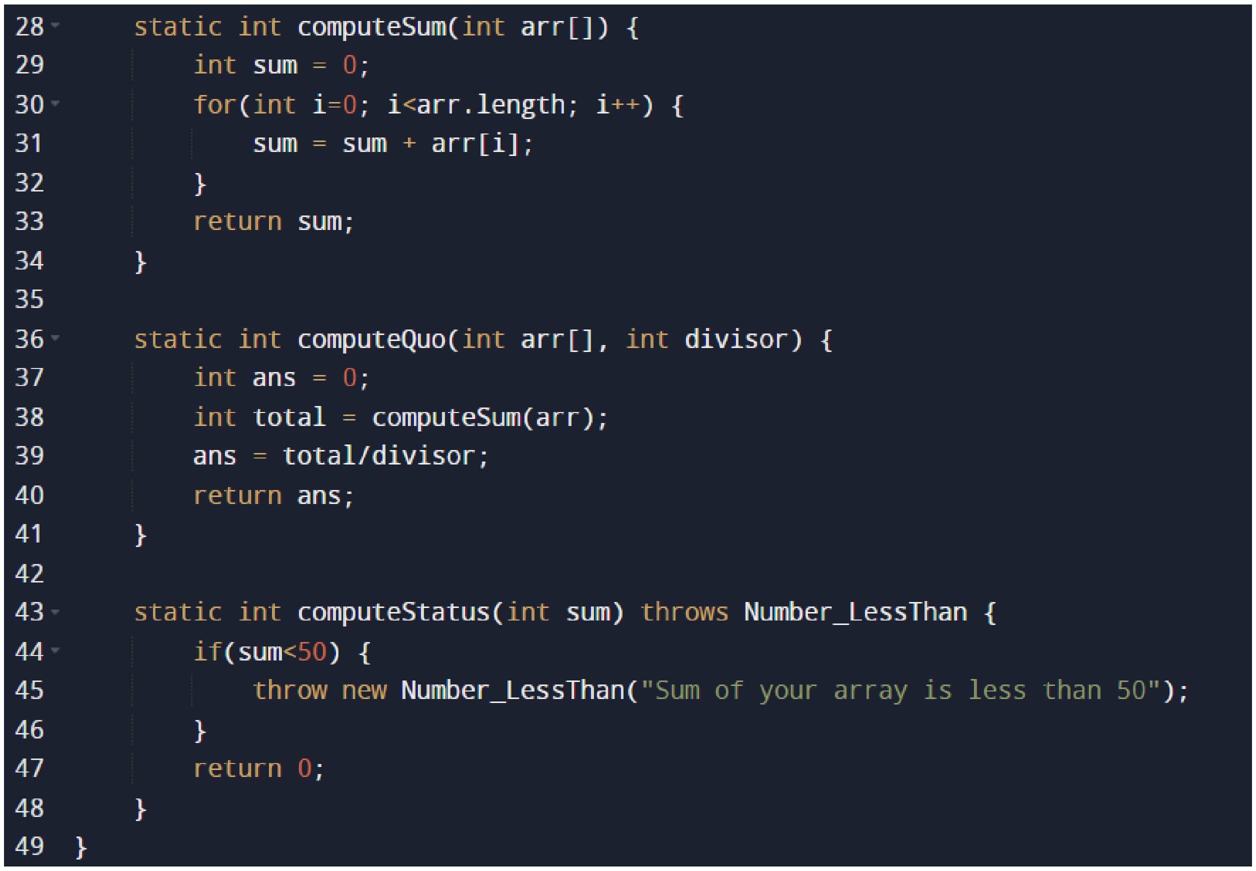
```
java -cp /tmp/wnbW2O3NBv Test

Sum of given array is 34
```

computeQuo() should find the quotient obtained by dividing the sum of numbers by divisor.

Program: -

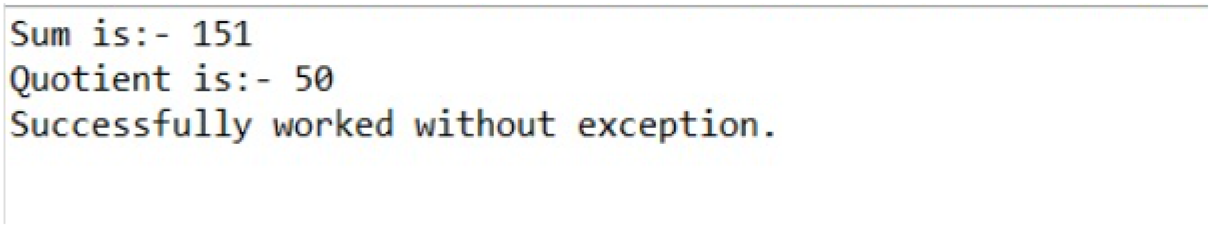


```java
package assesment_exam;

class Number_LessThan extends Exception{

    Number_LessThan(String str){
        super(str);
    }
}

public class Exceptn {

    public static void main(String[] args) throws Number_LessThan {
        int a[] = {1,2,3,4,5};     // Array to input for exception
        //int a[] = {10,20,30,40,51};   // array for not exception.
        int divisor = 3;            // Divisor

        //Calculating sum of array elements and assigning to sum.
        int sum = computeSum(a);
        System.out.println("Sum is:- " + sum);

        //Calculation quotient by passing array and divisor
        int quotient = computeQuo(a, divisor);
        System.out.println("Quotient is:- " + quotient);
        computeStatus(sum);
        System.out.println("Successfully worked without exception.");
    }

```

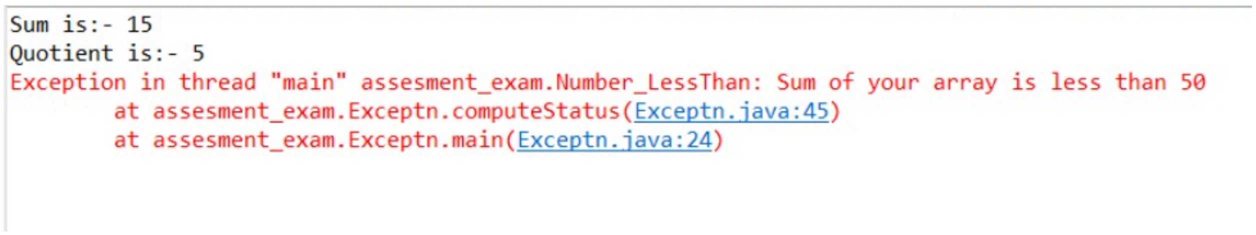


```
28      static int computeSum(int arr[]) {
29          int sum = 0;
30          for(int i=0; i<arr.length; i++) {
31              sum = sum + arr[i];
32          }
33          return sum;
34      }
35
36      static int computeQuo(int arr[], int divisor) {
37          int ans = 0;
38          int total = computeSum(arr);
39          ans = total/divisor;
40          return ans;
41      }
42
43      static int computeStatus(int sum) throws Number_LessThan {
44          if(sum<50) {
45              throw new Number_LessThan("Sum of your array is less than 50");
46          }
47          return 0;
48      }
49  }
```

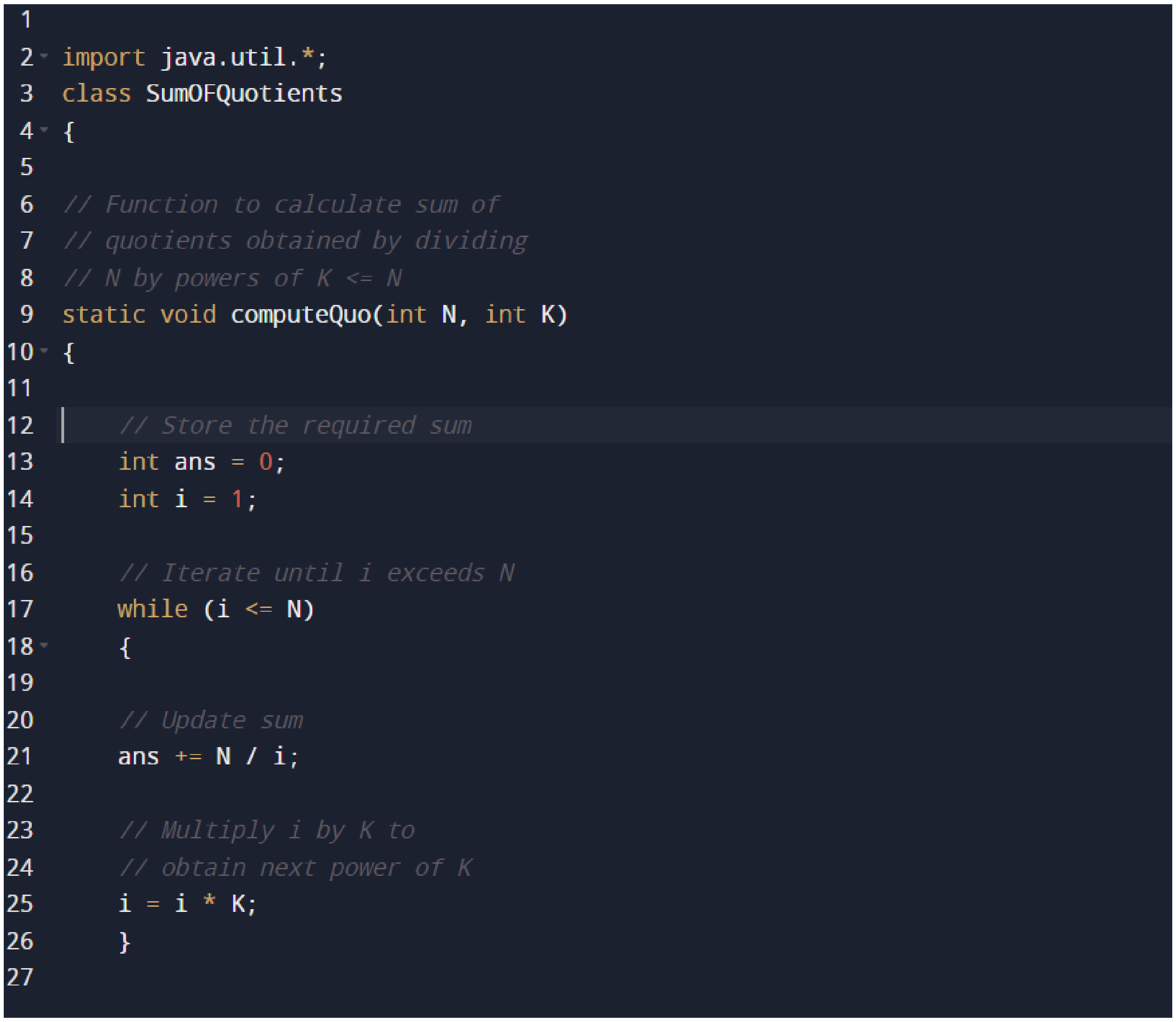
Output:-



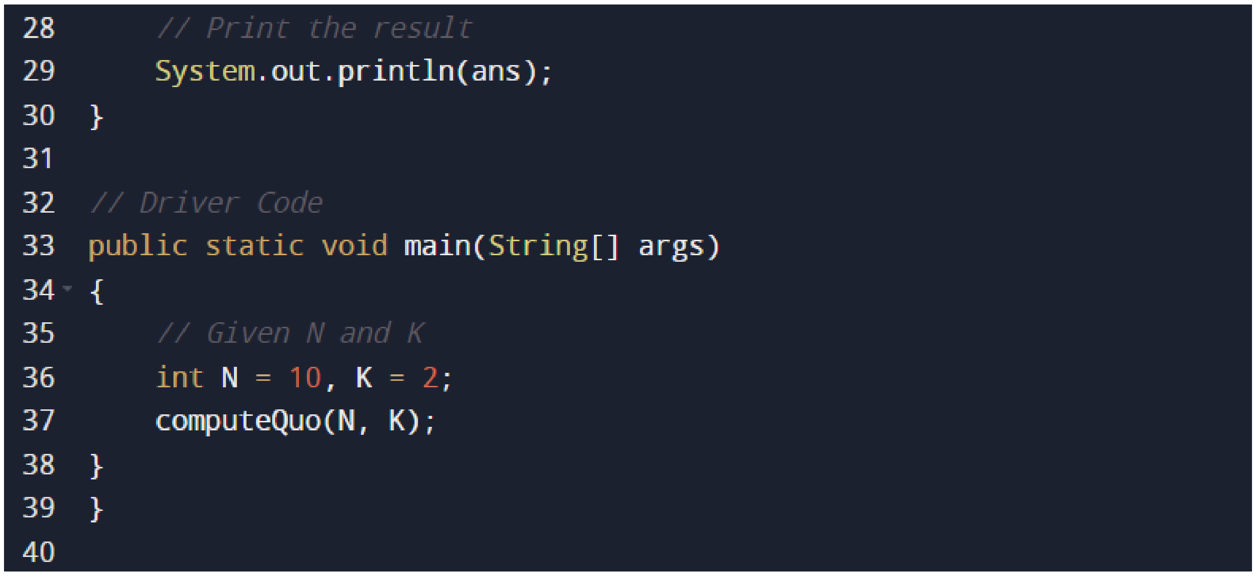
```
Sum is:- 151
Quotient is:- 50
Successfully worked without exception.
```



```
Sum is:- 15
Quotient is:- 5
Exception in thread "main" assesment_exam.Number_LessThan: Sum of your array is less than 50
	at assesment_exam.Exceptn.computeStatus(Exceptn.java:45)
	at assesment_exam.Exceptn.main(Exceptn.java:24)
```

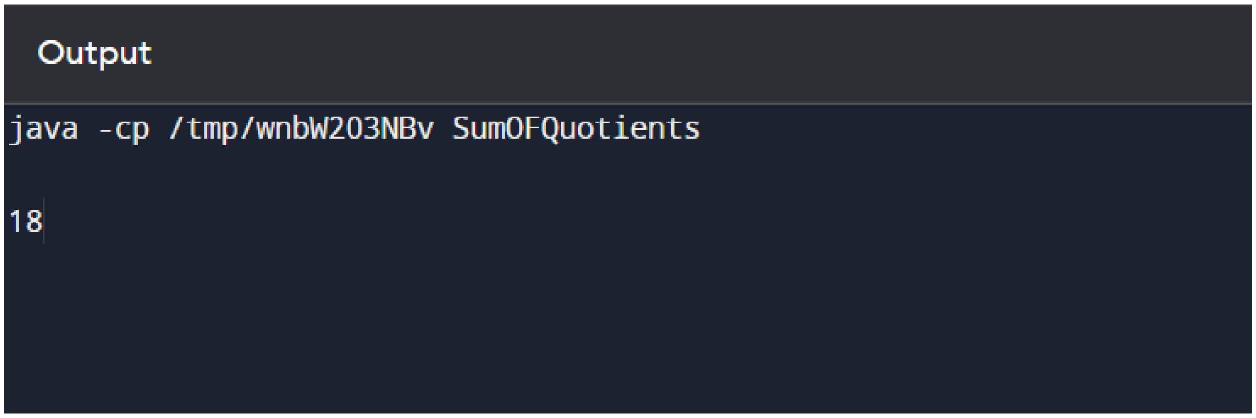


```java

import java.util.*;
class SumOFQuotients
{

// Function to calculate sum of
// quotients obtained by dividing
// N by powers of K <= N
static void computeQuo(int N, int K)
{

    // Store the required sum
    int ans = 0;
    int i = 1;

    // Iterate until i exceeds N
    while (i <= N)
    {

    // Update sum
    ans += N / i;

    // Multiply i by K to
    // obtain next power of K
    i = i * K;
    }

```



```java
    // Print the result
    System.out.println(ans);
}

// Driver Code
public static void main(String[] args)
{
    // Given N and K
    int N = 10, K = 2;
    computeQuo(N, K);
}
}

```

Output :-


Output

```
java -cp /tmp/wnbW203NBv SumOFQuotients

18
```

4. User is prompted to enter a password. If the password does not satisfy the following criteria, an exception is thrown as a weak password. A password is said to be strong if it satisfies the following criteria:

1. It contains at least one lowercase English character.

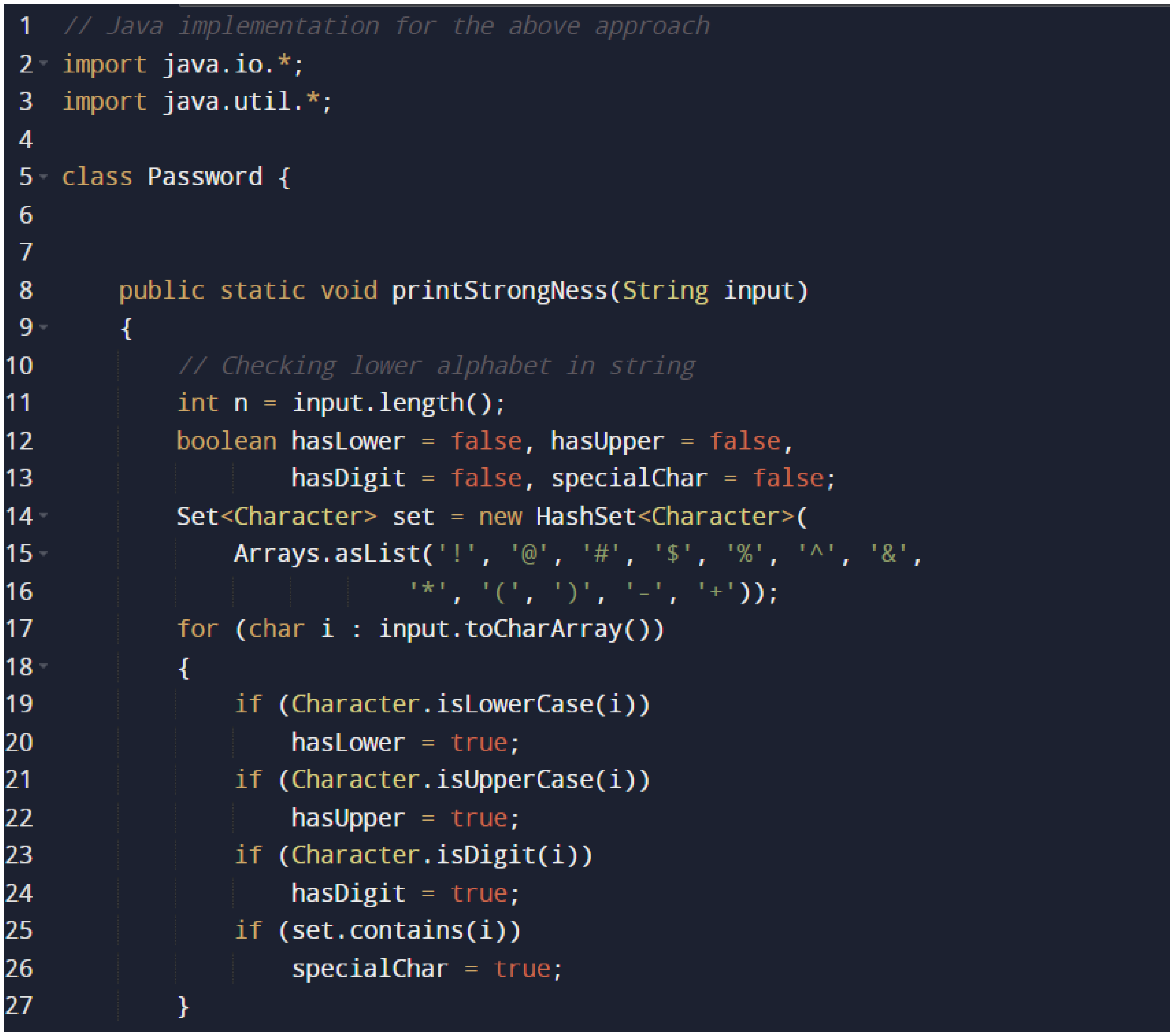
2. It contains at least one uppercase English character.

3. It contains at least one special character. The special characters are: ! @ # $ % ^ & * ( ) - +

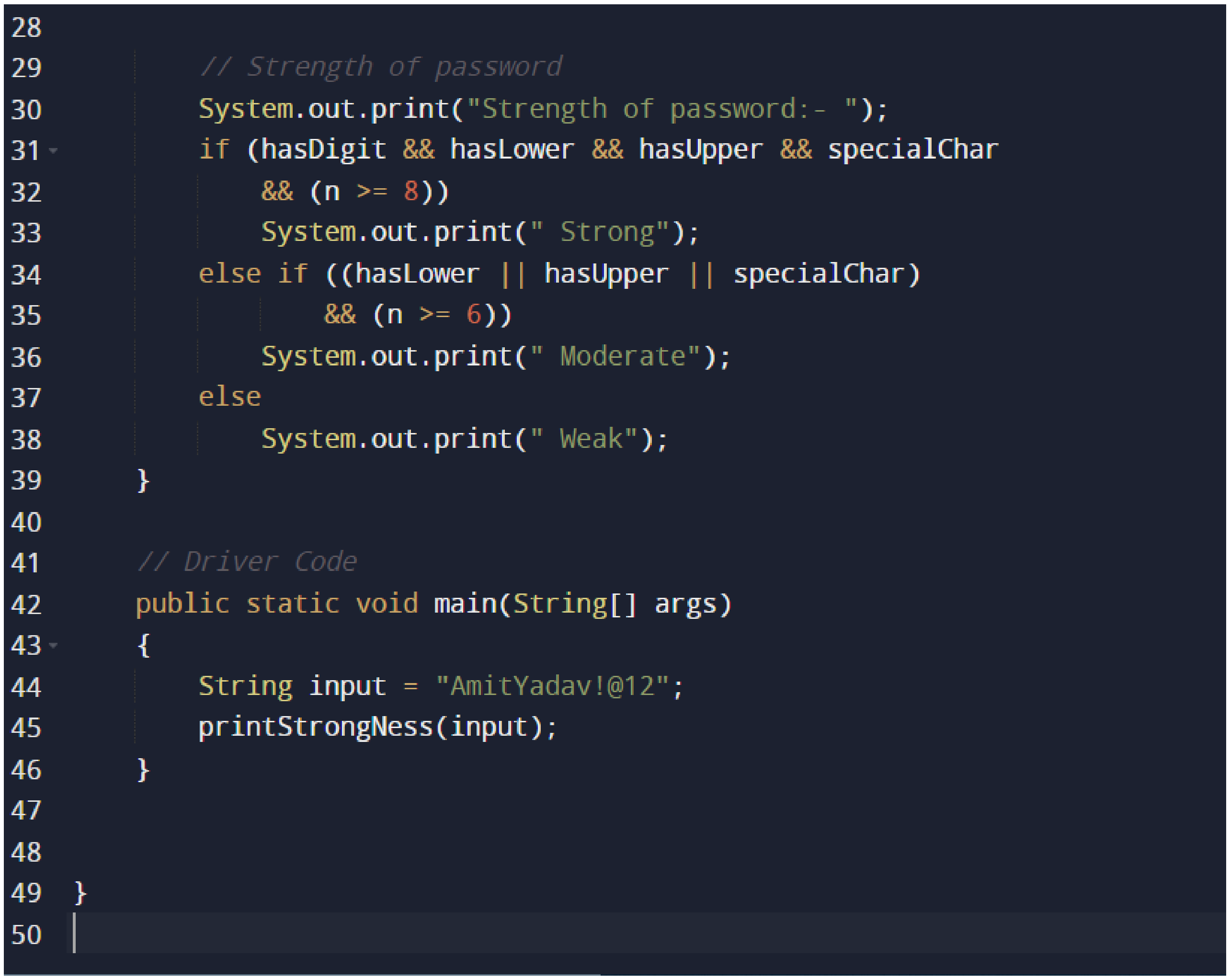
4. Its length is at least 8.

5. It contains at least one digit.

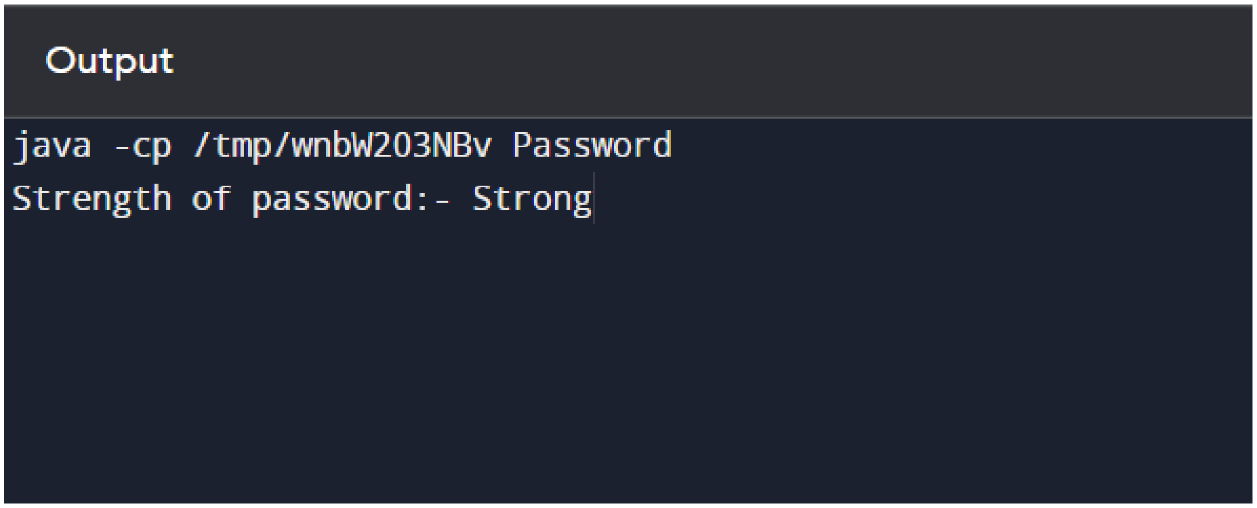
Program: -



```java
// Java implementation for the above approach
import java.io.*;
import java.util.*;

class Password {


    public static void printStrongNess(String input)
    {
        // Checking lower alphabet in string
        int n = input.length();
        boolean hasLower = false, hasUpper = false,
                hasDigit = false, specialChar = false;
        Set<Character> set = new HashSet<Character>(
            Arrays.asList('!', '@', '#', '$', '%', '^', '&',
                        '*', '(', ')', '-', '+'));
        for (char i : input.toCharArray())
        {
            if (Character.isLowerCase(i))
                hasLower = true;
            if (Character.isUpperCase(i))
                hasUpper = true;
            if (Character.isDigit(i))
                hasDigit = true;
            if (set.contains(i))
                specialChar = true;
        }
```


```java

        // Strength of password
        System.out.print("Strength of password:- ");
        if (hasDigit && hasLower && hasUpper && specialChar
            && (n >= 8))
            System.out.print(" Strong");
        else if ((hasLower || hasUpper || specialChar)
                && (n >= 6))
            System.out.print(" Moderate");
        else
            System.out.print(" Weak");
    }

    // Driver Code
    public static void main(String[] args)
    {
        String input = "AmitYadav!@12";
        printStrongNess(input);
    }


}

```

Output :-

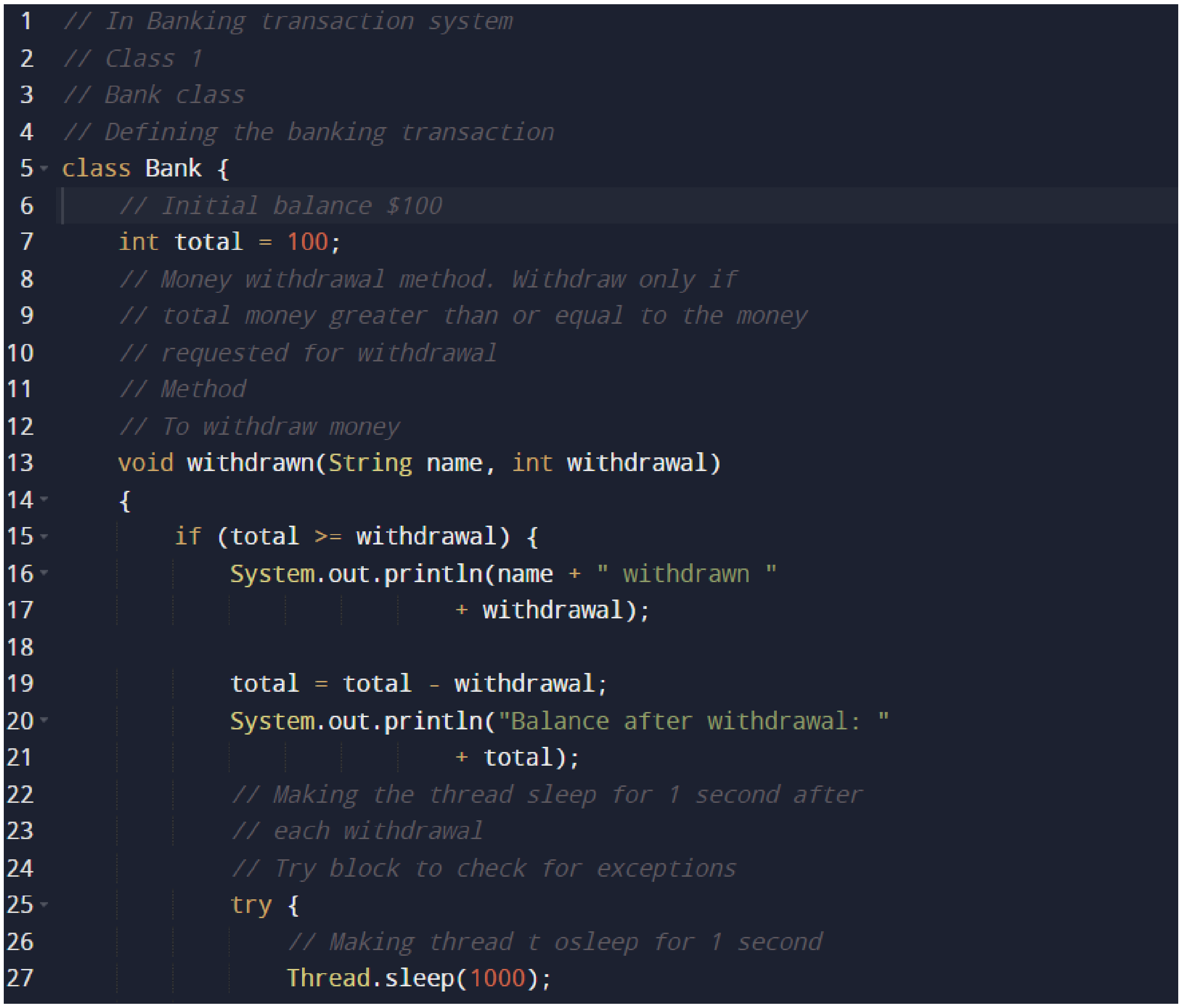

Output

```
java -cp /tmp/wnbW2O3NBv Password
Strength of password:- Strong
```

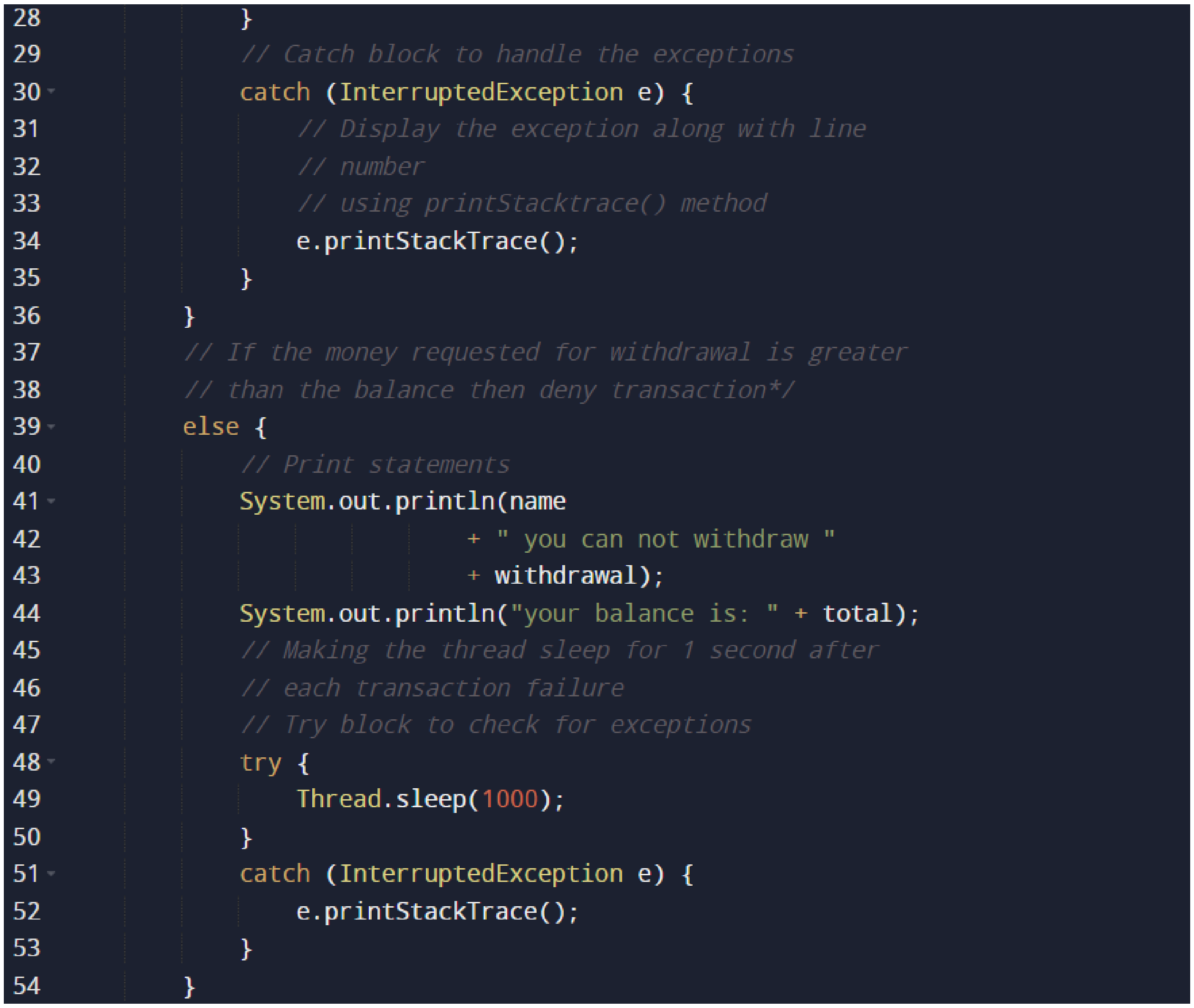
5. Banking Application - Create a simple java code to implement Banking Application. Make sure these 4 functions are present (Transfer amount, View Balance, Deposit, Withdraw) Apply the following java concepts in your program and update the following inference

- **Classes & Objects**
- **Default Constructor**
- **Parameterized constructor**
- **Constructor Overloading**
- **Inheritance (with the type used & diagram)**
- **Java Concept Where u used Y u used**
- **Use of getter & setter method.**
- **This**
- **Static variable**
- **Static function**

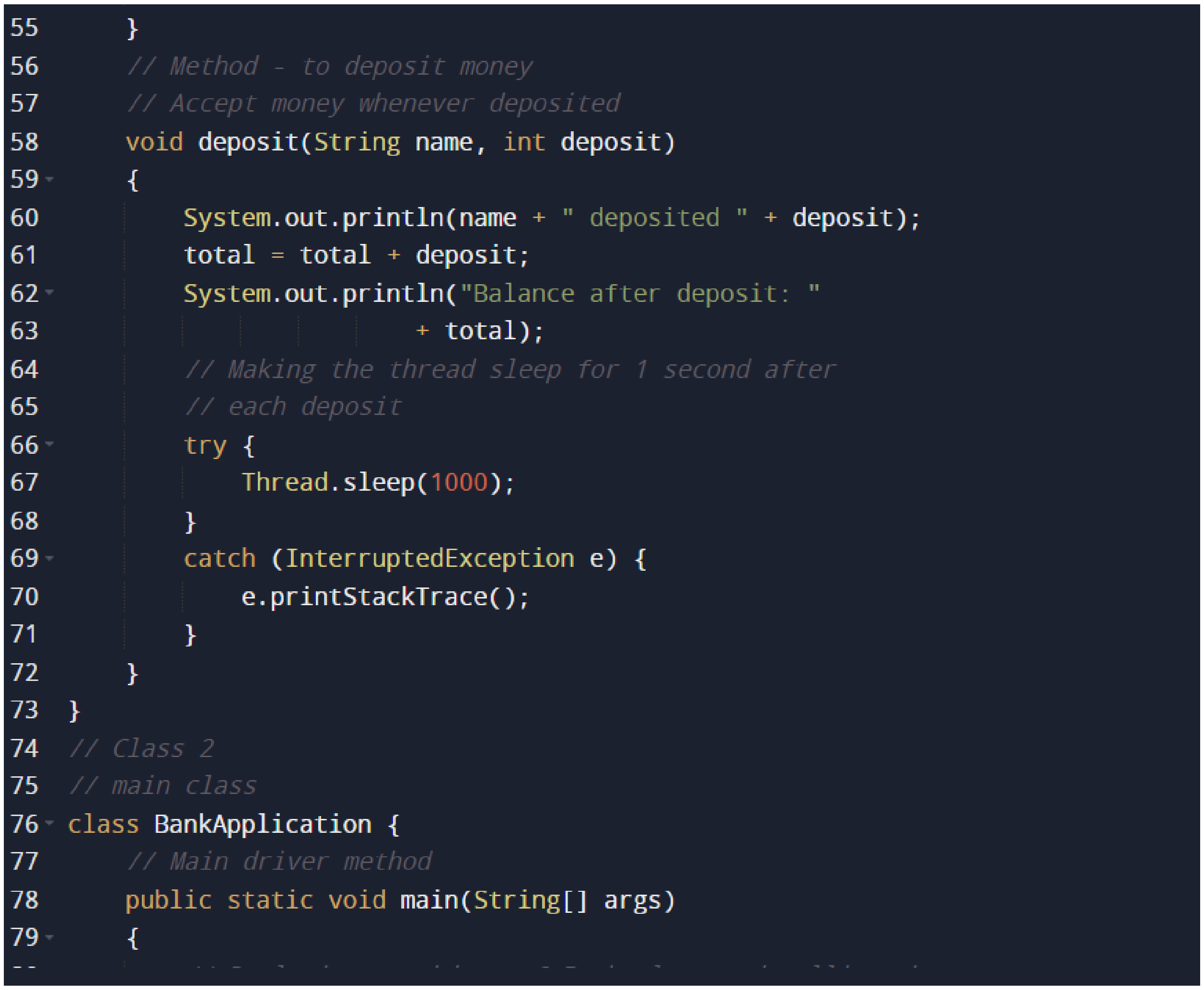
Program: -



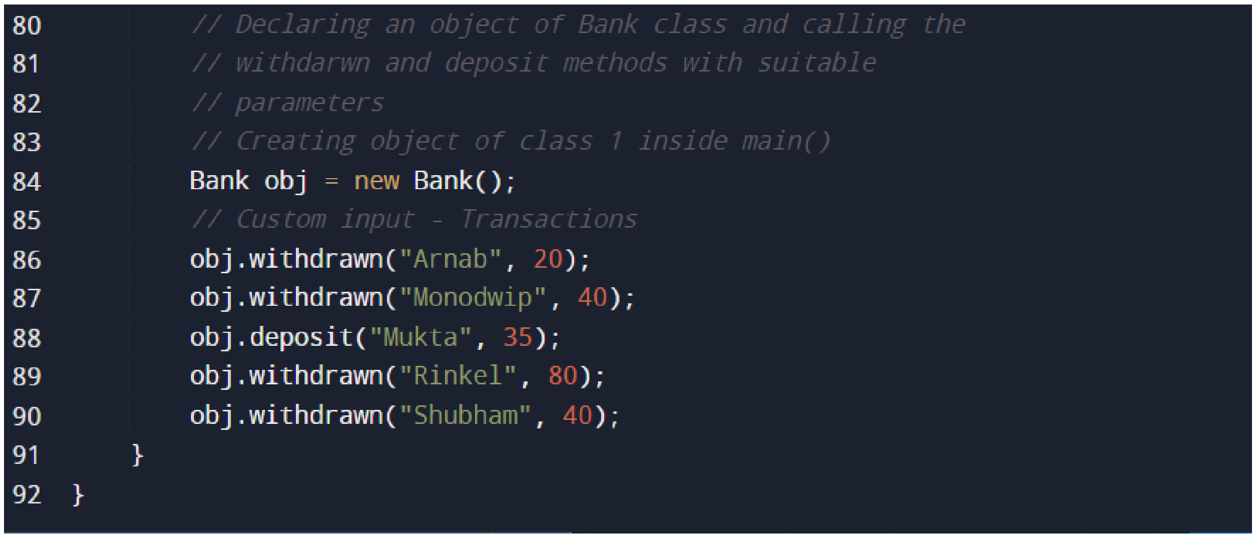
```java
// In Banking transaction system
// Class 1
// Bank class
// Defining the banking transaction
class Bank {
    // Initial balance $100
    int total = 100;
    // Money withdrawal method. Withdraw only if
    // total money greater than or equal to the money
    // requested for withdrawal
    // Method
    // To withdraw money
    void withdrawn(String name, int withdrawal)
    {
        if (total >= withdrawal) {
            System.out.println(name + " withdrawn "
                               + withdrawal);

            total = total - withdrawal;
            System.out.println("Balance after withdrawal: "
                               + total);
            // Making the thread sleep for 1 second after
            // each withdrawal
            // Try block to check for exceptions
            try {
                // Making thread t osleep for 1 second
                Thread.sleep(1000);
```



```java
            }
            // Catch block to handle the exceptions
            catch (InterruptedException e) {
                // Display the exception along with line
                // number
                // using printStacktrace() method
                e.printStackTrace();
            }
        }
        // If the money requested for withdrawal is greater
        // than the balance then deny transaction*/
        else {
            // Print statements
            System.out.println(name
                            + " you can not withdraw "
                            + withdrawal);
            System.out.println("your balance is: " + total);
            // Making the thread sleep for 1 second after
            // each transaction failure
            // Try block to check for exceptions
            try {
                Thread.sleep(1000);
            }
            catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
```



```java
    }
    // Method - to deposit money
    // Accept money whenever deposited
    void deposit(String name, int deposit)
    {
        System.out.println(name + " deposited " + deposit);
        total = total + deposit;
        System.out.println("Balance after deposit: "
                           + total);
        // Making the thread sleep for 1 second after
        // each deposit
        try {
            Thread.sleep(1000);
        }
        catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
// Class 2
// main class
class BankApplication {
    // Main driver method
    public static void main(String[] args)
    {
```



```java
        // Declaring an object of Bank class and calling the
        // withdarwn and deposit methods with suitable
        // parameters
        // Creating object of class 1 inside main()
        Bank obj = new Bank();
        // Custom input - Transactions
        obj.withdrawn("Arnab", 20);
        obj.withdrawn("Monodwip", 40);
        obj.deposit("Mukta", 35);
        obj.withdrawn("Rinkel", 80);
        obj.withdrawn("Shubham", 40);
    }
}
```

Output :-

```
Arnab withdrawn 20
Balance after withdrawal: 80

//After 1 Second
Monodwip withdrawn 40
Balance after withdrawal: 40

//After 1 Second
Mukta deposited 35
Balance after deposit: 75

//After 1 Second
Rinkel you can not withdraw 80
your balance is: 75

//After 1 Second
Shubham withdrawn 40
Balance after withdrawal: 35
```