

**(DIGITAL ASSIGNMENT - 3) exceptions and multithreading**

**CSE1007(JAVA PROGRAMMING)LAB:L31-L32**



**March 13, 2022**

**ANIRUDH VADERA**

**20BCE2940**

**EXCEPTION:**

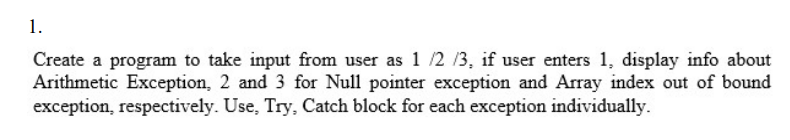
* The **Exception Handling in Java** is one of the powerful mechanism to handle the runtime errors so that the normal flow of the application can be maintained.
* In Java, an exception is an event that disrupts the normal flow of the program. It is an object which is thrown at runtime.
* Exception Handling is a mechanism to handle runtime errors such as ClassNotFoundException, IOException, SQLException, RemoteException, etc.

**MULTITHREADING:**

* **Multithreading** is a process of executing multiple threads simultaneously.
* A thread is a lightweight sub-process, the smallest unit of processing. Multiprocessing and multithreading, both are used to achieve multitasking.
* However, we use multithreading than multiprocessing because threads use a shared memory area. They don't allocate separate memory area so saves memory, and context-switching between the threads takes less time than process.
* Java Multithreading is mostly used in games, animation, etc.

**ACTIVITY – 6:**

**QUESTION 1:**

****

**CODE:**

import java.util.Scanner;

public class activity6q1 {

    public static void main(String[] args) {

        Scanner in = new Scanner(System.in);

        System.out.println("Enter Your Choice : ");

        System.out.println("1 : Arithmetic Exception");

        System.out.println("2 : Null Pointer Exception");

        System.out.println("3 : Array Index Out of Range Exception");

        int n = in.nextInt();

        System.out.println("ANIRUDH VADERA (20BCE2940)");

        switch (n) {

            case 1:

                int a = 0, b = 10;

                int c = 0;

                try {

                    c = b / a;

                    System.out.println("Result(In Try) = " + c);

                } catch (ArithmeticException e) {

                    e.printStackTrace();

                    System.out.println("We are just printing the stack trace.\n"

                            + "ArithmeticException is handled. But take care of the variable \"c\"");

                }

                System.out.println("Value of c(Outside Try) :" + c);

                break;

            case 2:

                try {

                    String temp = null; // null value

                    System.out.println(temp.charAt(0));

                } catch (NullPointerException e) {

                    e.printStackTrace();

                    System.out.println("NullPointerException..");

                }

                break;

            case 3:

                try {

                    int[] array = { 1, 2, 3, 4, 5 }; // length is 5

                    int test = array[6]; // accessing 25th element

                    System.out.println(test);

                } catch (StringIndexOutOfBoundsException e) {

                    e.printStackTrace();

                    System.out.println("ArrayIndexOutOfBoundsException");

                }

                break;

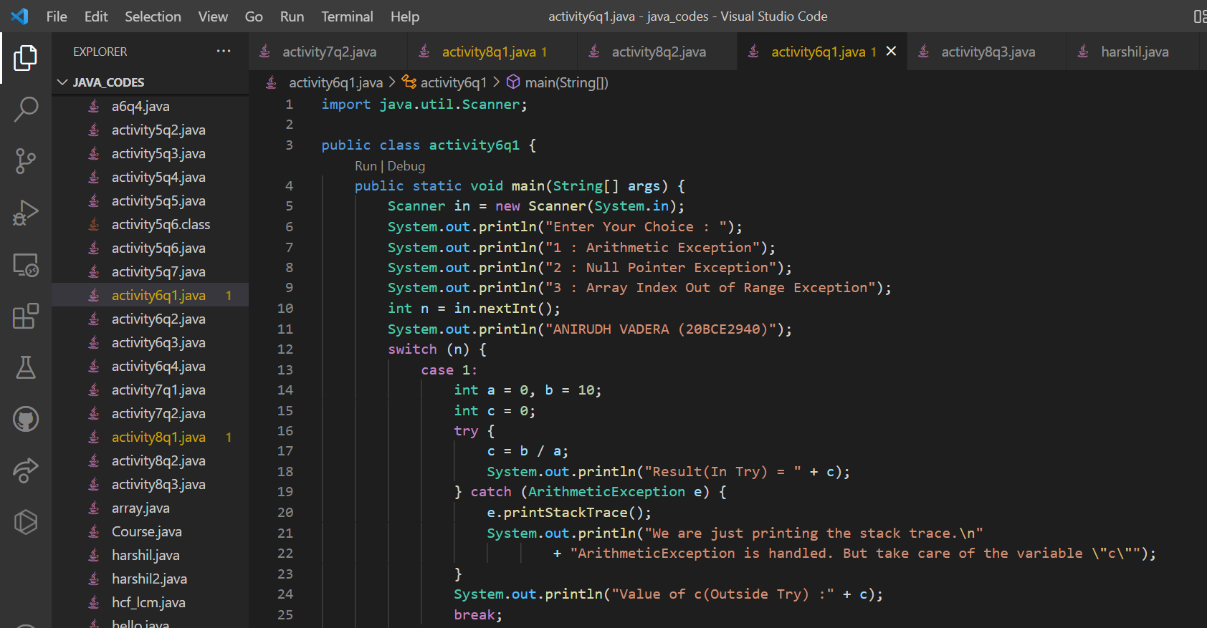
        }

        in.close();

    }

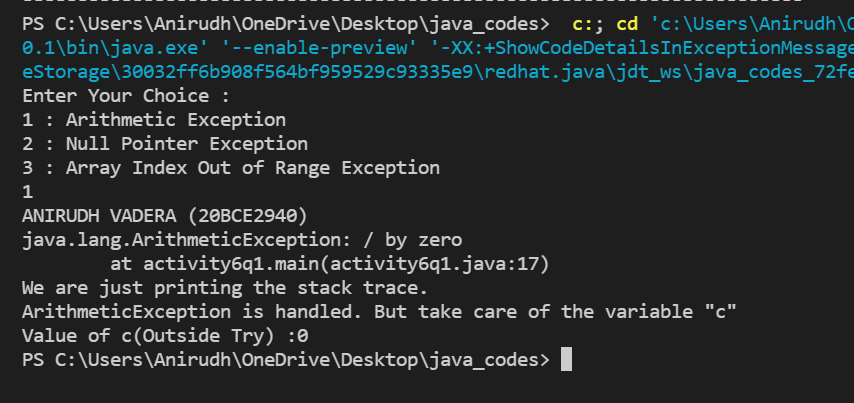
}

**CODE SNAPSHOT:**

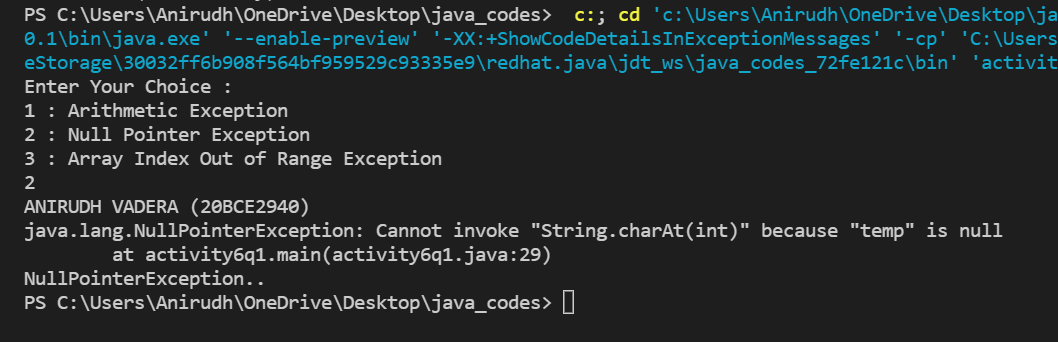
****

**OUTPUT:**

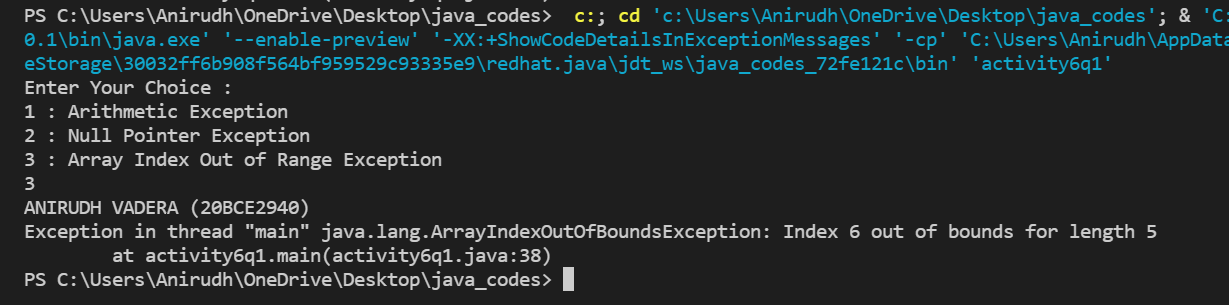
**CASE1(ARITHMATIC EXCEPTION):**

****

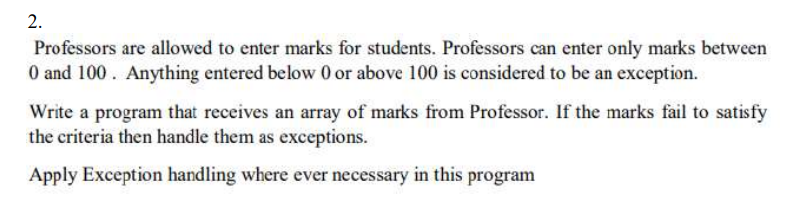
**CASE2(NULL POINTER EXCEPTION):**

****

**CASE3(ARRAY INDEX OUT OF RANGE EXCEPTION):**

****

**QUESTION 2:**

****

**CODE:**

import java.util.Scanner;

class invalidMarks extends Exception {

    public invalidMarks(String message) {

        super(message);

    }

}

public class activity6q2 {

    static void validate(int mark) throws invalidMarks {

        if (mark < 0 || mark > 100) {

            throw new invalidMarks("The Marks Should be between 0 and 100");

        }

    }

    public static void main(String[] args) {

        System.out.println("Enter the number of students : ");

        Scanner in = new Scanner(System.in);

        int n = in.nextInt();

        int array[] = new int[n];

        System.out.println("Enter Marks : ");

        for (int i = 0; i < n; i++) {

            System.out.println("Enter Marks for Student : " + (i + 1));

            array[i] = in.nextInt();

        }

        for (int i = 0; i < n; i++) {

            try {

                validate(array[i]);

            } catch (invalidMarks e) {

                e.printStackTrace();

                System.out.println("exception Caught");

                System.out.println("Exception occured for student " + (i + 1) + " : " + e);

                System.out.println("Enter marks for student " + (i + 1) + " Again ");

                array[i] = in.nextInt();

            }

        }

        System.out.println("After Correcting the Marks : ");

        for (int i = 0; i < n; i++) {

            System.out.println("Marks for Student : " + (i + 1) + " : " + array[i]);

        }

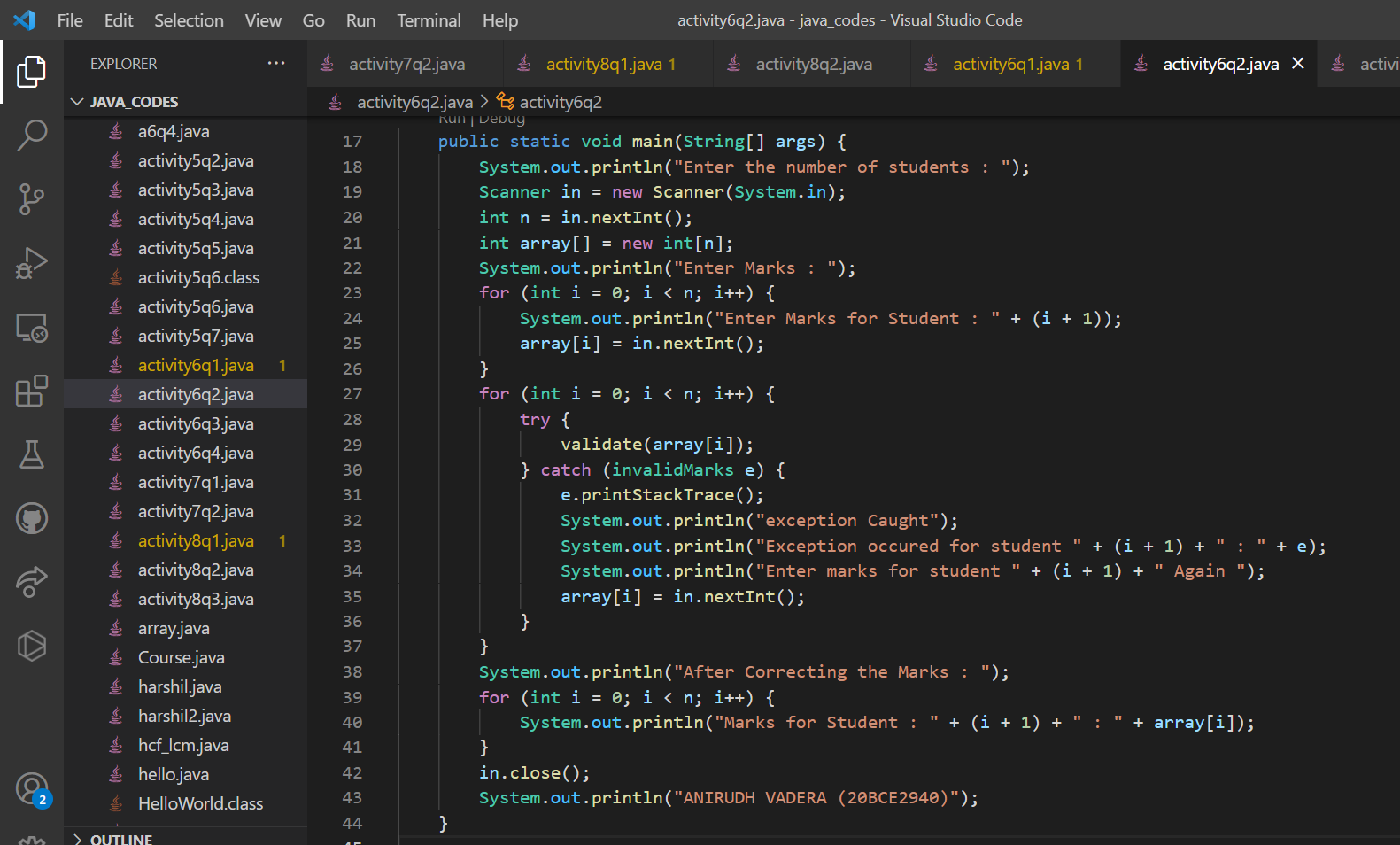
        in.close();

        System.out.println("ANIRUDH VADERA (20BCE2940)");

    }

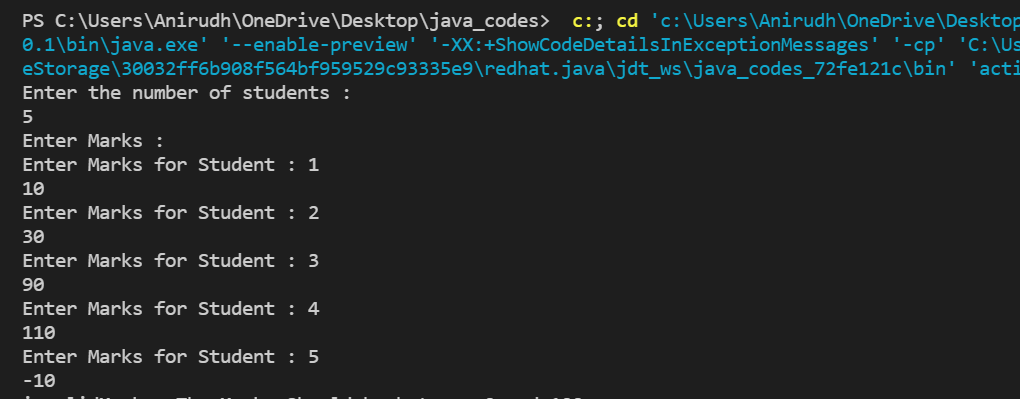
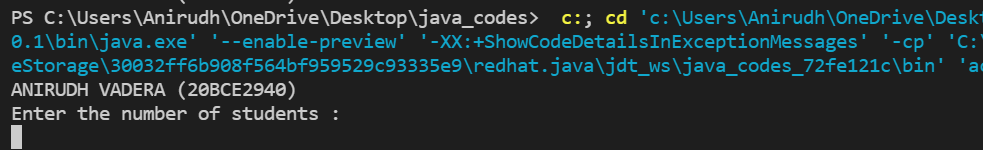
}

**CODE SNAPSHOT:**

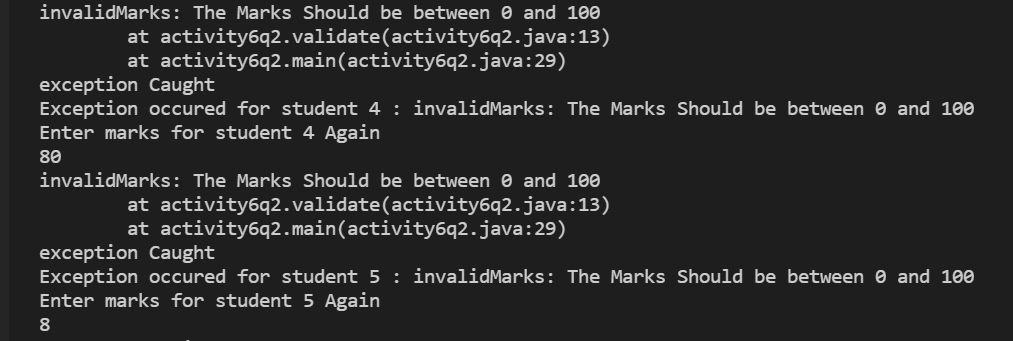
****

**OUTPUT:**

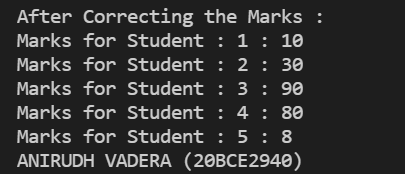
**GIVING WRONG MARKS FOR STUDENT 4 AND 5:**

****

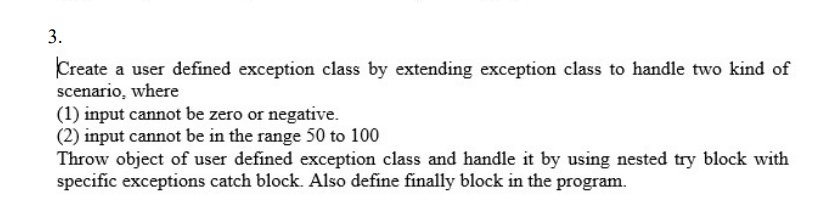
**EXCEPTION CAUGHT FOR MARKS FOR STUDENT 4 AND 5 RE-ENTERING THEIR MARKS:**

****

**FINALLY DISPLAYING THE MARKS:**

****

**QUESTION 3:**

****

**CODE:**

import java.util.Scanner;

class invalidInputLess0 extends Exception {

    public invalidInputLess0(String message) {

        super(message);

    }

}

class invalidInputNotRange extends Exception {

    public invalidInputNotRange(String message) {

        super(message);

    }

}

public class activity6q3 {

    static void validate(int input) throws invalidInputLess0, invalidInputNotRange {

        if (input <= 0) {

            throw new invalidInputLess0("The Input is equal to 0 or Negative");

        }

        if (input > 50 && input < 100) {

            throw new invalidInputNotRange("The Input cannot be in the range 50 to 100");

        }

    }

    public static void main(String[] args) {

        System.out.println("ANIRUDH VADERA (20BCE2940)");

        System.out.println("Enter the Input : ");

        Scanner in = new Scanner(System.in);

        int n = in.nextInt();

        try {

            validate(n);

            try {

                validate(n);

            } catch (Exception e) {

                e.printStackTrace();

                System.out.println("Exception Caught :");

                System.out.println("Exception occured for the Input : ");

            }

        } catch (Exception e) {

            e.printStackTrace();

            System.out.println("Exception Caught :");

            System.out.println("Exception occured for the Input : ");

        } finally {

            System.out.println("Inside the finally block performing the cleanup and closing the Scanner object :");

            in.close();

        }

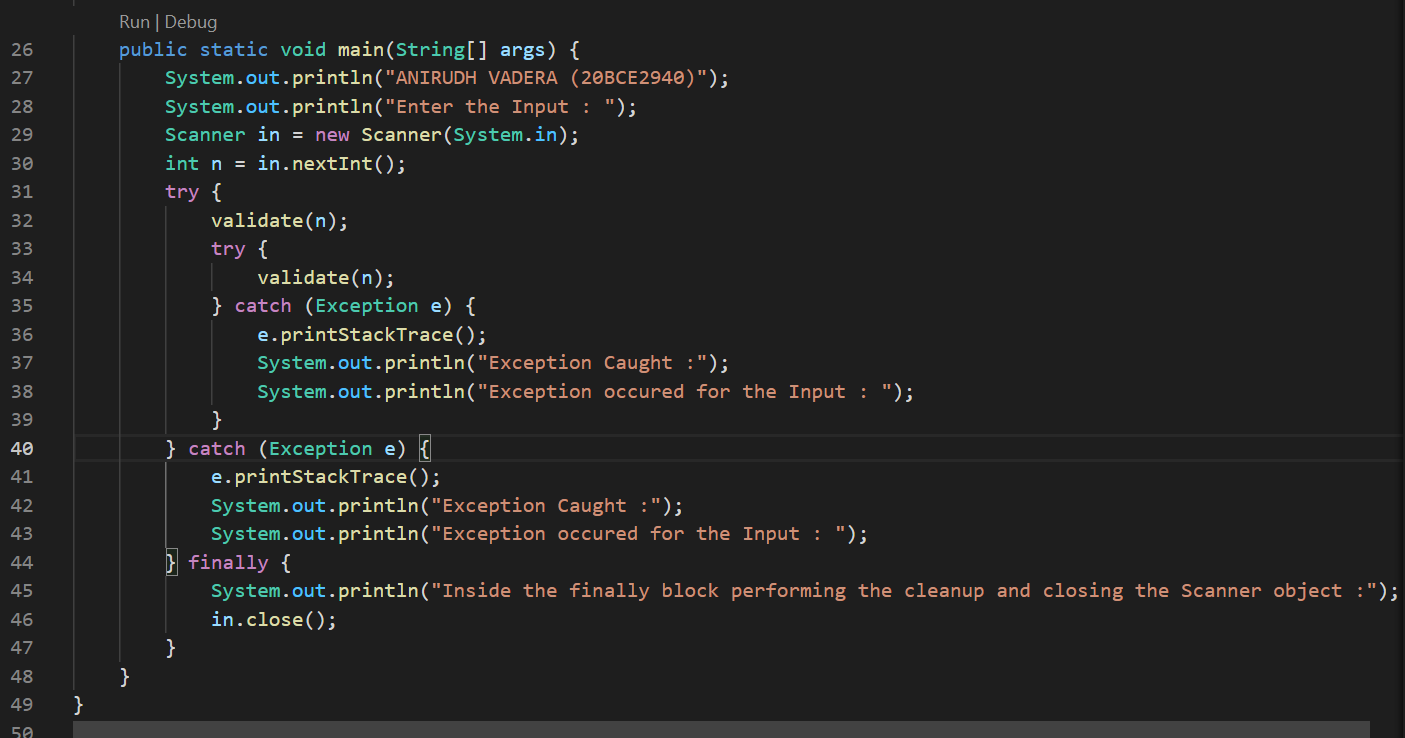
    }

}

**CODE SNAPSHOT:**

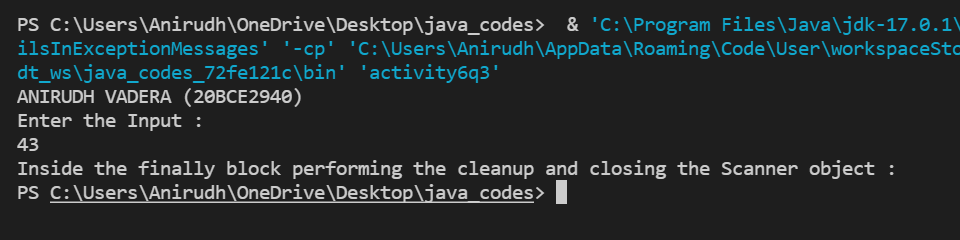
**INCLUDED THE NESTED TRY\_CATCH BLOCK:**

**THE FINALLY BLOCK ALWAYS EXECUTE AND CLOSES THE SCANNER OBJECT NO MATTER WHAT:**

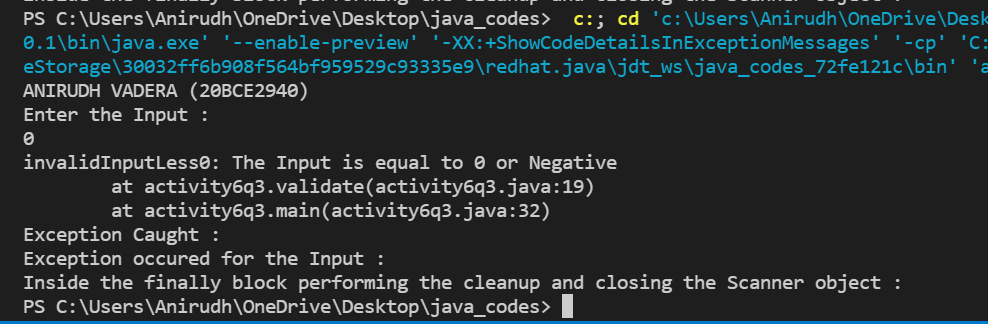
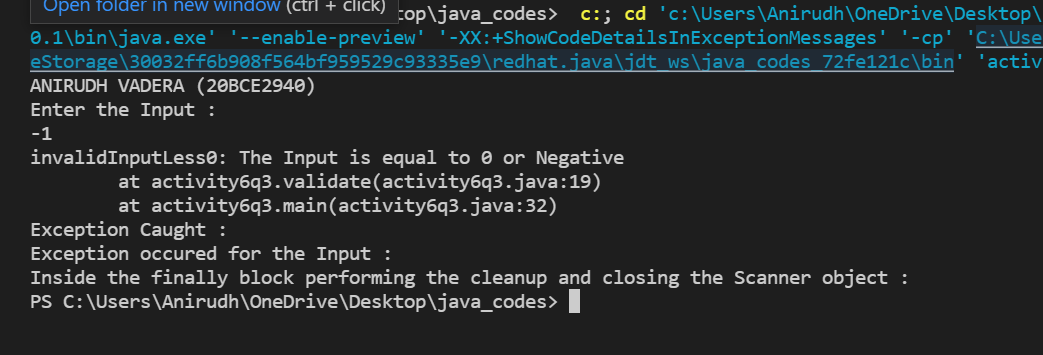
****

**OUTPUT:**

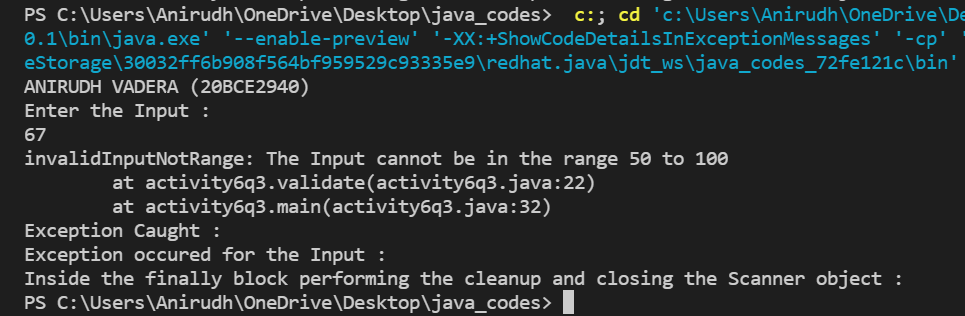
**CORRECT INPUT:**

****

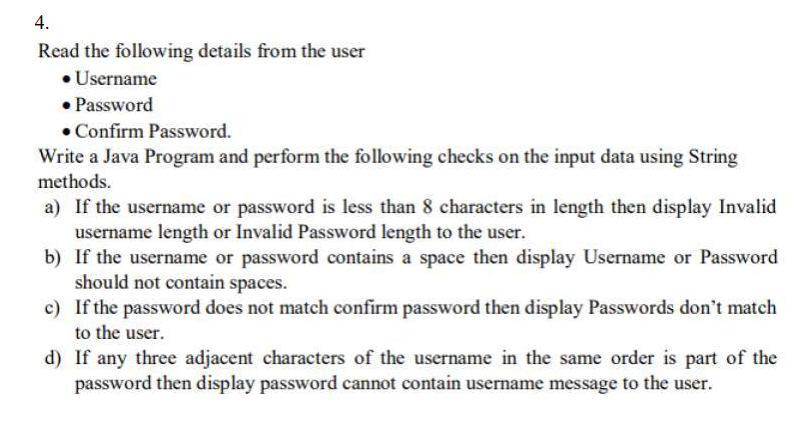
**WRONG INPUT (ZERO OR NEGATIVE):**

****

**WRONG INPUT (IN RANGE 50 TO 100):**

****

**QUESTION 4:**

****

**CODE:**

import java.util.Scanner;

class invalidInput extends Exception {

    public invalidInput(String message) {

        super(message);

    }

}

public class activity6q4 {

    static int check = 0;

    static void validate(String userName, String password, String confirmPassword, int flag) throws invalidInput {

        switch (flag) {

            case 1:

                if (userName.length() < 8 || password.length() < 8) {

                    check = 0;

                    throw new invalidInput("Invalid UserName Lnegth or Invalid Password Lnegth");

                }

                break;

            case 2:

                if (userName.contains(" ") || password.contains(" ")) {

                    check = 0;

                    throw new invalidInput("UserName or Password should not contain space.");

                }

                break;

            case 3:

                if (!(password.equals(confirmPassword))) {

                    check = 0;

                    throw new invalidInput("Passwords dont match : ");

                }

                break;

            case 4:

                String temp = new String();

                for (int i = 0; i < (userName.length() - 2); i++) {

                    temp = "";

                    temp = temp.concat(userName.substring(i, i + 3));

                    if (password.contains(temp)) {

                        check = 0;

                        throw new invalidInput("Passwords cannot contain userName : ");

                    }

                }

                break;

        }

        check = 1;

    }

    public static void main(String[] args) {

        System.out.println("ANIRUDH VADERA (20BCE2940)");

        Scanner in = new Scanner(System.in);

        System.out.println("Enter the UserName : ");

        String userName = in.nextLine();

        System.out.println("Enter the Password : ");

        String password = in.nextLine();

        System.out.println("Confirm your Password : ");

        String confirmPassword = in.nextLine();

        int iterator = 1;

        while (check == 0) {

            try {

                validate(userName, password, confirmPassword, iterator);

                iterator++;

                validate(userName, password, confirmPassword, iterator);

                iterator++;

                validate(userName, password, confirmPassword, iterator);

                iterator++;

                validate(userName, password, confirmPassword, iterator);

            } catch (invalidInput e) {

                e.printStackTrace();

                System.out.println("Exception Caught :");

                if (iterator == 1) {

                    System.out.println("Enter Your UserName and Password again : ");

                    System.out.print("UserName : ");

                    userName = in.nextLine();

                    System.out.println();

                    System.out.print("Password : ");

                    password = in.nextLine();

                }

                if (iterator == 2) {

                    System.out.println("Enter Your UserName and Password again : ");

                    System.out.print("UserName : ");

                    userName = in.nextLine();

                    System.out.println();

                    System.out.print("Password : ");

                    password = in.nextLine();

                }

                if (iterator == 3) {

                    System.out.println("Enter Your Passwords again : ");

                    System.out.print("Password : ");

                    password = in.nextLine();

                    System.out.println();

                    System.out.print("Confirm Password : ");

                    confirmPassword = in.nextLine();

                }

                if (iterator == 4) {

                    System.out.println("Enter Your UserName and Password again : ");

                    System.out.print("UserName : ");

                    userName = in.nextLine();

                    System.out.println();

                    System.out.print("Password : ");

                    password = in.nextLine();

                }

            }

        }

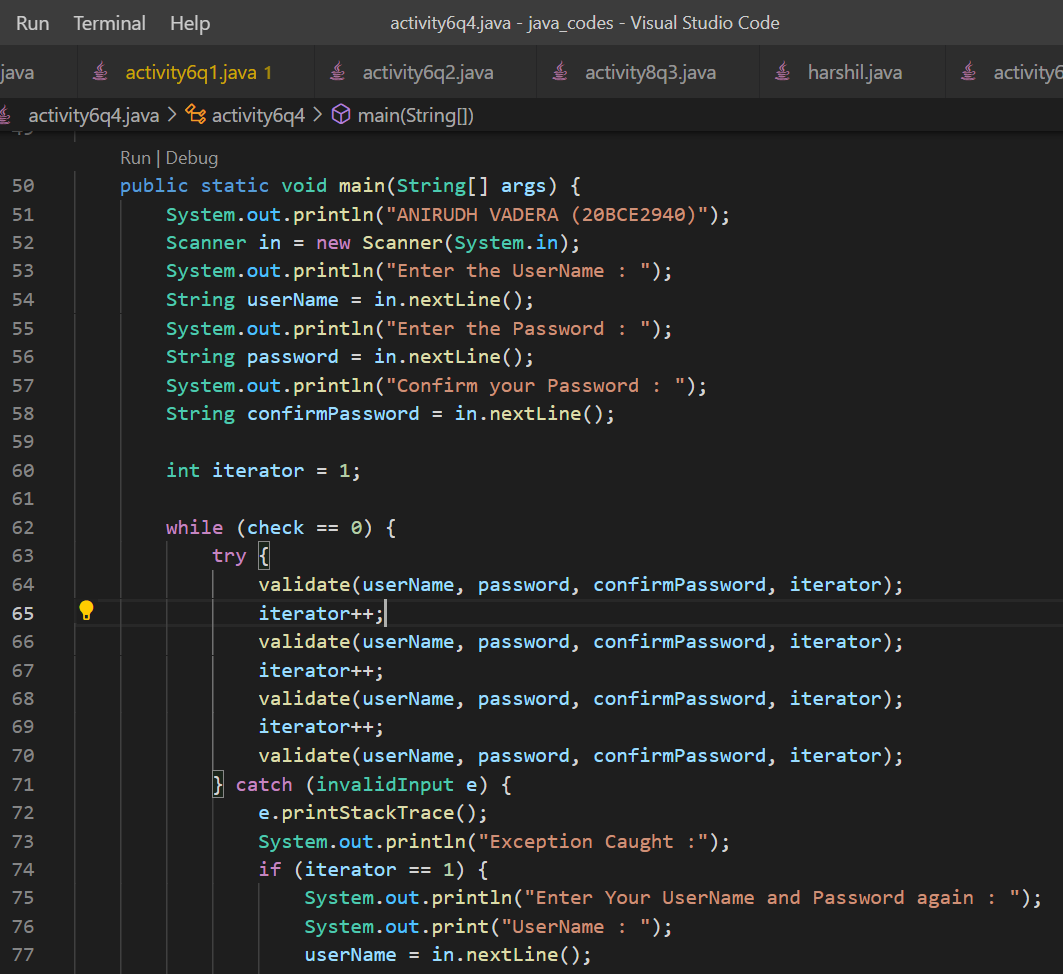
        System.out.println("Everything is verified and correct");

        in.close();

    }

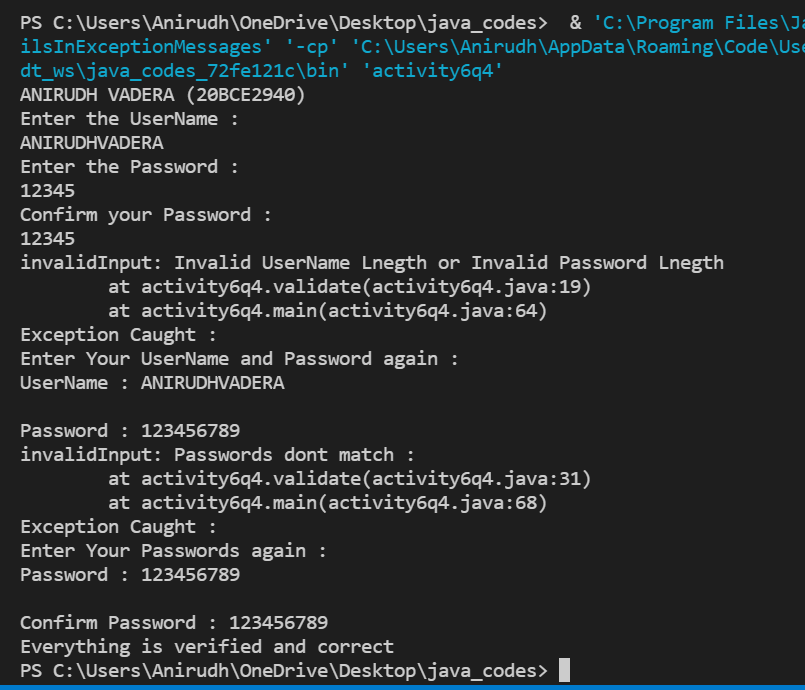
}

**CODE SNAPSHOT:**

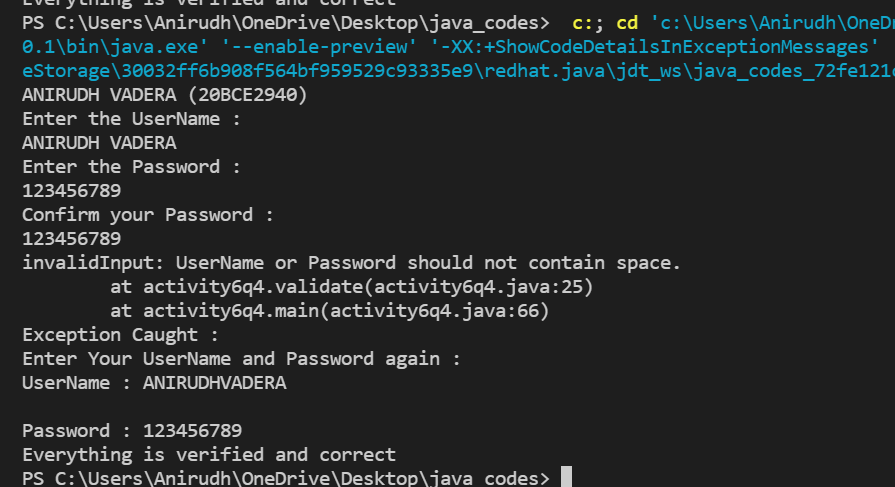
****

**OUTPUT:**

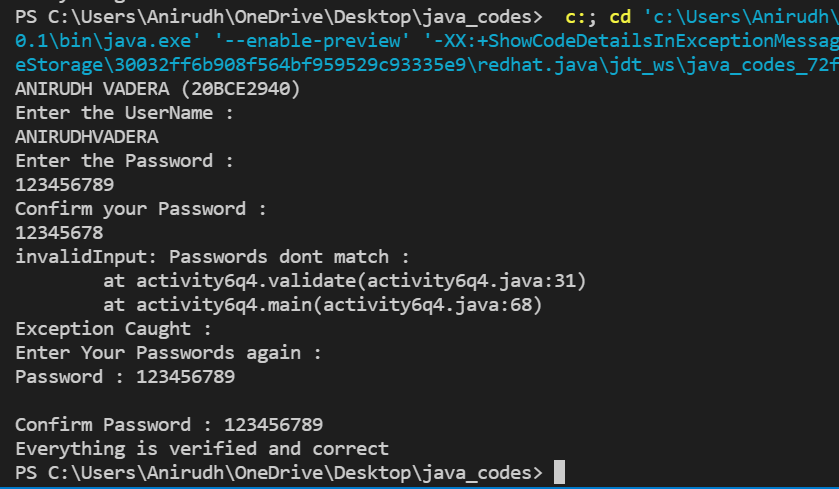
**CASE1(USERNAME OR PASSWORD LENGTH IS LESS THAN 8 RE-ENTERING THE REQUIRED):**

****

**CASE2(USERNAME OR PASSWORD CONTAINS SPACE RE-ENTERING THE REQUIRED):**

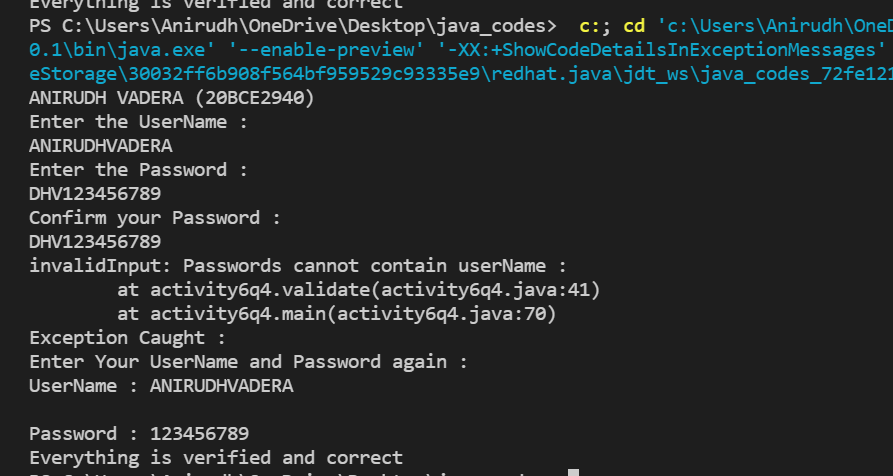
****

**CASE3(PASSWORDS DOESN’T MATCH):**

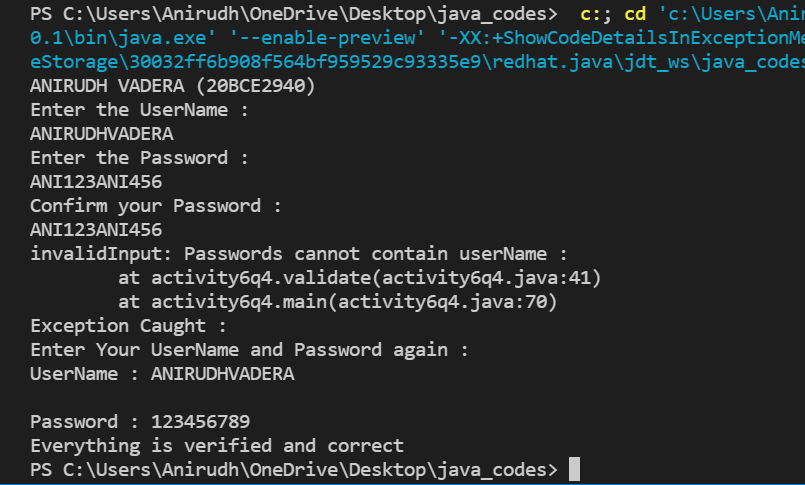
****

**CASE3(3 ADJACENT CHARACTERS OF USERNAME IN ORDER IS PART OF THE PASSWORD):**

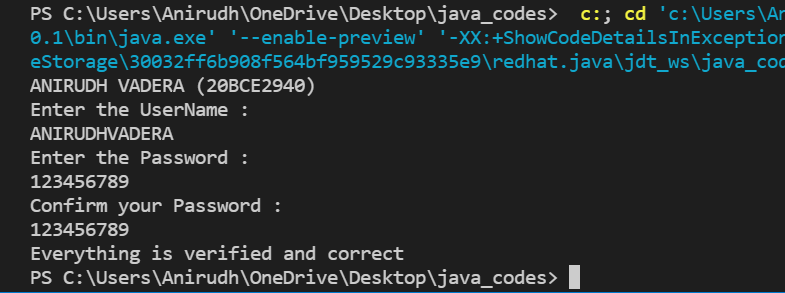
**IN THIS EXAMPLE FROM USERNAME(DHV) IS A PART OF PASSWORD**

****

**IN THIS EXAMPLE FROM USERNAME(ANI) IS A PART OF PASSWORD 2 TIMES:**

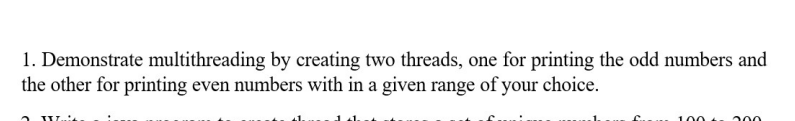
****

**WHEN EVERYTHING IS ENTERED CORRECTLY:**

****

**ACTIVITY 7**

**QUESTION 1:**

****

**CODE:**

class Array {

    static int[] a = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

    public static synchronized void print(String odd\_even) {

        if (odd\_even.equals("ODD")) {

            System.out.println("Printing Odd Numbers");

            System.out.print("[ ");

            for (int i = 0; i < a.length; i++) {

                if (a[i] % 2 != 0) {

                    System.out.print(a[i] + " ");

                }

            }

            System.out.println("]");

        } else {

            System.out.println("Printing Even Numbers");

            System.out.print("[ ");

            for (int i = 0; i < a.length; i++) {

                if (a[i] % 2 == 0) {

                    System.out.print(a[i] + " ");

                }

            }

            System.out.println("]");

        }

    }

}

class Thread1 implements Runnable {

    public synchronized void run() {

    }

}

class Thread2 implements Runnable {

    public synchronized void run() {

    }

}

public class activity7q1 {

    public static void main(String[] args) throws InterruptedException {

        System.out.println("ANIRUDH VADERA(20BCE2940)");

        Array a = new Array();

        Thread t1 = new Thread(() -> {

            System.out.println("Thread 1 - ODD");

            System.out.println(Thread.currentThread().getName());

            Array.print("ODD");

        });

        Thread t2 = new Thread(() -> {

            System.out.println("Thread 2 - EVEN");

            System.out.println(Thread.currentThread().getName());

            Array.print("EVEN");

        });

        t1.start();

        t2.start();

        t1.join();

        t2.join();

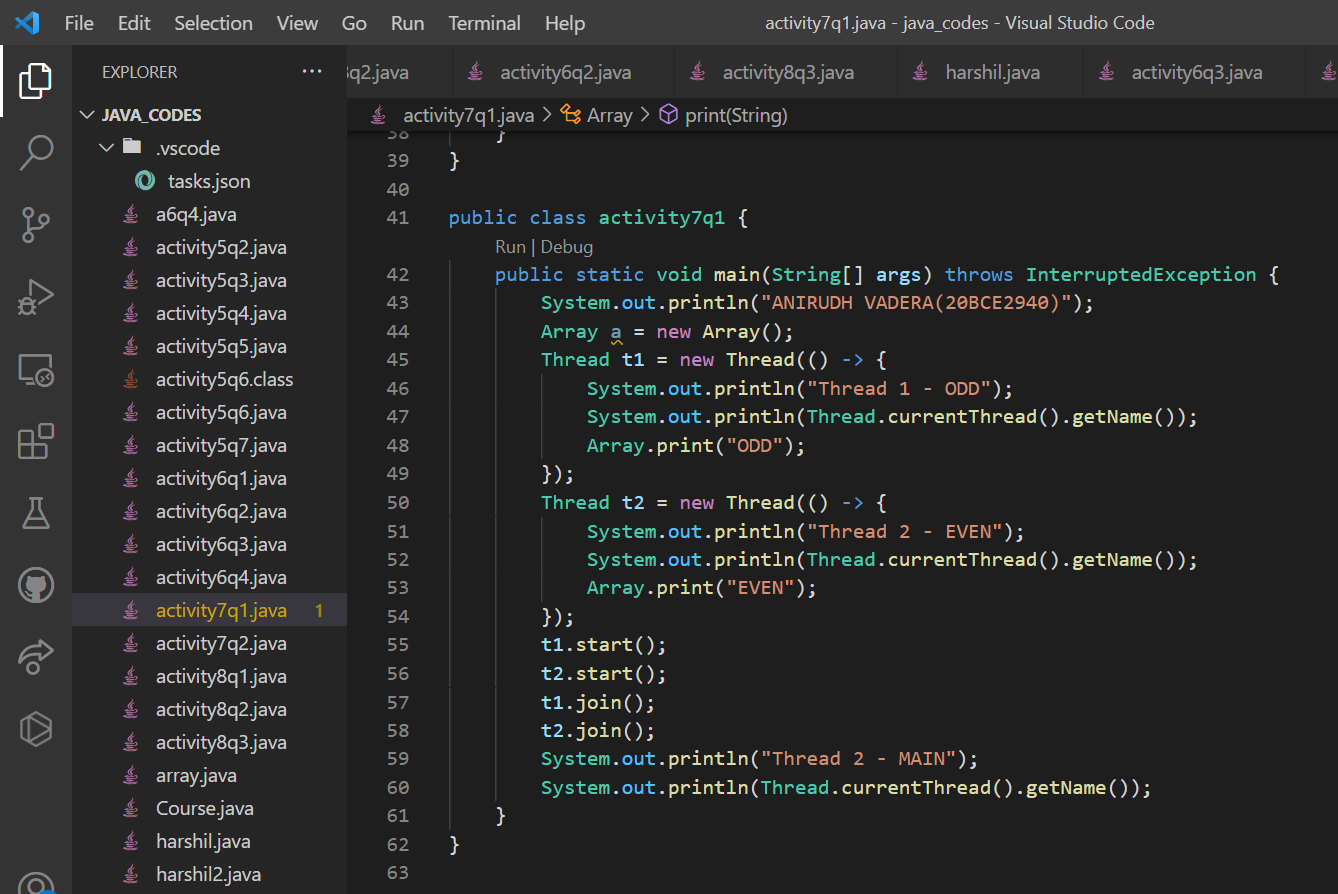
        System.out.println("Thread 2 - MAIN");

        System.out.println(Thread.currentThread().getName());

    }

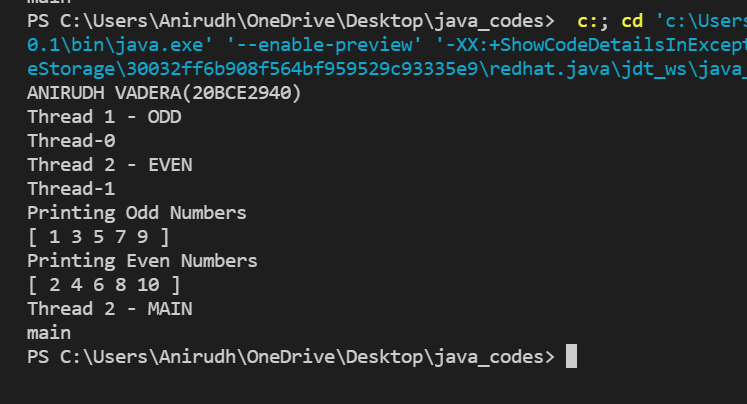
}

**CODE SNAPSHOT:**

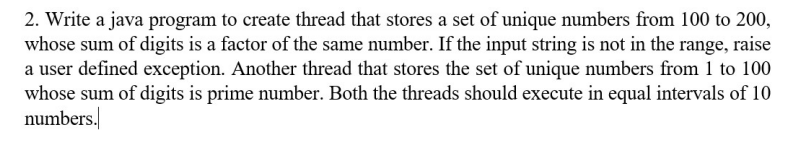
****

**OUTPUT:**

**Created an array object which has a static synchronized function to print odd and even numbers from the array as it is a static synchronized function different objects can acquire the lock and there will be no discrepancy.**

****

**QUESTION 2:**

****

**CODE:**

import java.util.Scanner;

class invalidInputNotRange1 extends Exception {

    public invalidInputNotRange1(String message) {

        super(message);

    }

}

class invalidInputNotRange2 extends Exception {

    public invalidInputNotRange2(String message) {

        super(message);

    }

}

class Thread1 implements Runnable {

    int[] number;

    int range1\_low;

    int range1\_high;

    int sum = 0;

    int[] array\_factor = new int[100];

    int factor\_elements = 0;

    int[] unique = new int[100];

    int unique\_elements = 0;

    int start;

    int end;

    Thread1(int[] number, int start, int end, int range1\_low, int range1\_high) {

        this.number = number;

        this.range1\_high = range1\_high;

        this.range1\_low = range1\_low;

        this.start = start;

        this.end = end;

    }

    void validate1(int input) throws invalidInputNotRange1 {

        if (input >= range1\_low && input <= range1\_high) {

        } else {

            throw new invalidInputNotRange1(

                    "The Input (" + input + ") is not in the Specified Range of " + "[" +

                            range1\_low + "," +

                            range1\_high + "]");

        }

    }

    public void generate() {

        for (int number = range1\_low; number < (range1\_high + 1); number++) {

            sum = 0;

            int temp = number;

            while (temp != 0) {

                int c = temp % 10;

                sum = sum + c;

                temp = temp / 10;

            }

            if (number % sum == 0) {

                array\_factor[factor\_elements++] = number;

            }

        }

    }

    public void run() {

        generate();

        System.out.println("Thread 1 - Numbers whose sum of digits is a factor ofnumber itself");

        for (int j = start; j < end; j++) {

            try {

                validate1(number[j]);

                int flag = 0;

                for (int i = 0; i < factor\_elements; i++) {

                    if (number[j] == array\_factor[i]) {

                        flag = 1;

                        break;

                    }

                }

                if (flag == 1) {

                    unique[unique\_elements++] = number[j];

                }

            } catch (invalidInputNotRange1 e) {

                System.out.println(e);

            }

        }

        print();

    }

    public void print() {

        System.out.println("The Numbers that satisfies the conditions are : ");

        System.out.println("Printing Numbers whose sum of digits is a factor ofnumber itself");

        System.out.print("[ ");

        for (int i = 0; i < unique\_elements; i++) {

            System.out.print(unique[i] + " ");

        }

        System.out.println("]");

    }

}

class Thread2 implements Runnable {

    int[] number;

    int range2\_low;

    int range2\_high;

    int sum = 0;

    int[] array\_factor = new int[100];

    int factor\_elements = 0;

    int[] unique = new int[100];

    int unique\_elements = 0;

    int start;

    int end;

    Thread2(int[] number, int start, int end, int range2\_low, int range2\_high) {

        this.number = number;

        this.range2\_high = range2\_high;

        this.range2\_low = range2\_low;

        this.start = start;

        this.end = end;

    }

    void validate2(int input) throws invalidInputNotRange2 {

        if (input >= range2\_low && input <= range2\_high) {

        } else {

            throw new invalidInputNotRange2(

                    "The Input (" + input + ") is not in the Specified Range of " + "[" + range2\_low + "," + range2\_high

                            + "]");

        }

    }

    public void generate() {

        for (int number = range2\_low; number < (range2\_high + 1); number++) {

            sum = 0;

            int temp = number;

            while (temp != 0) {

                int c = temp % 10;

                sum = sum + c;

                temp = temp / 10;

            }

            int flag = 0;

            for (int i = 2; i < sum; i++) {

                if (sum % i == 0) {

                    flag = 1;

                    break;

                }

            }

            if (flag == 0) {

                array\_factor[factor\_elements++] = number;

            }

        }

    }

    public void run() {

        generate();

        System.out.println("Thread 2 - Numbers whose sum of digits is a PrimeNumber");

        for (int j = start; j < end; j++) {

            try {

                validate2(number[j]);

                int flag = 0;

                for (int i = 0; i < factor\_elements; i++) {

                    if (number[j] == array\_factor[i]) {

                        flag = 1;

                        break;

                    }

                }

                if (flag == 1) {

                    unique[unique\_elements++] = number[j];

                }

            } catch (invalidInputNotRange2 e) {

                System.out.println(e);

            }

        }

        print();

    }

    public void print() {

        System.out.println("The Numbers that satisfies the conditions are : ");

        System.out.println("Printing Numbers whose sum of digits is a Prime Number");

        System.out.print("[ ");

        for (int i = 0; i < unique\_elements; i++) {

            System.out.print(unique[i] + " ");

        }

        System.out.println("]");

    }

}

public class activity7q2 {

    static int range1\_low;

    static int range1\_high;

    static int range2\_low;

    static int range2\_high;

    public static void main(String[] args) throws InterruptedException {

        Scanner in = new Scanner(System.in);

        System.out.println("Enter the range1\_low : ");

        activity7q2.range1\_low = in.nextInt();

        System.out.println("Enter the range1\_high : ");

        activity7q2.range1\_high = in.nextInt();

        System.out.println("Enter the range2\_low : ");

        activity7q2.range2\_low = in.nextInt();

        System.out.println("Enter the range2\_high : ");

        activity7q2.range2\_high = in.nextInt();

        System.out.println("Enter The set of Number Strings to be checked(Inmultiples of 10) : ");

        int n = in.nextInt();

        n = n \* 10;

        int[] set = new int[n];

        for (int i = 0; i < (n); i++) {

            set[i] = (i + 1) \* 10;

        }

        for (int i = 0; i < n / 10; i++) {

            int start = (i \* 10);

            int end = (i \* 10) + 10;

            Thread1 factor = new Thread1(set, start, end, range1\_low, range1\_high);

            Thread t1 = new Thread(factor);

            Thread2 prime = new Thread2(set, start, end, activity7q2.range2\_low,

                    range2\_high);

            Thread t2 = new Thread(prime);

            t1.start();

            t1.join();

            t2.start();

            t2.join();

        }

        System.out.println("ANIRUDH VADERA(20BCE2940)");

        System.out.println("Thread 2 - MAIN");

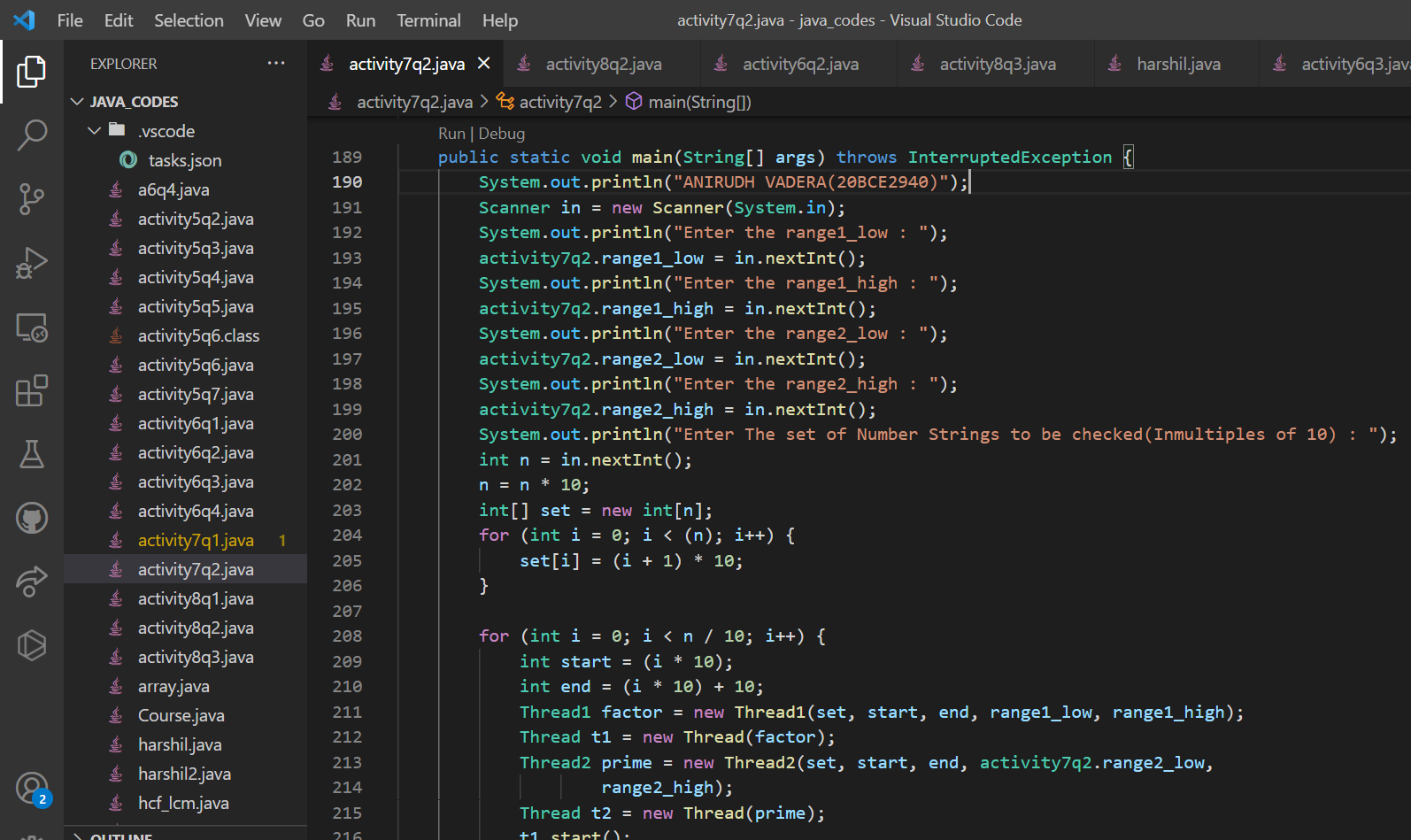
        System.out.println(Thread.currentThread().getName());

        in.close();

    }

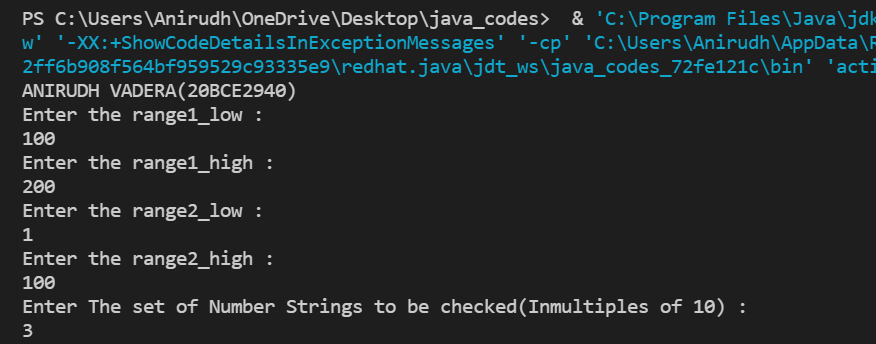
}

**CODE SNAPSHOT:**

****

**OUTPUT:**

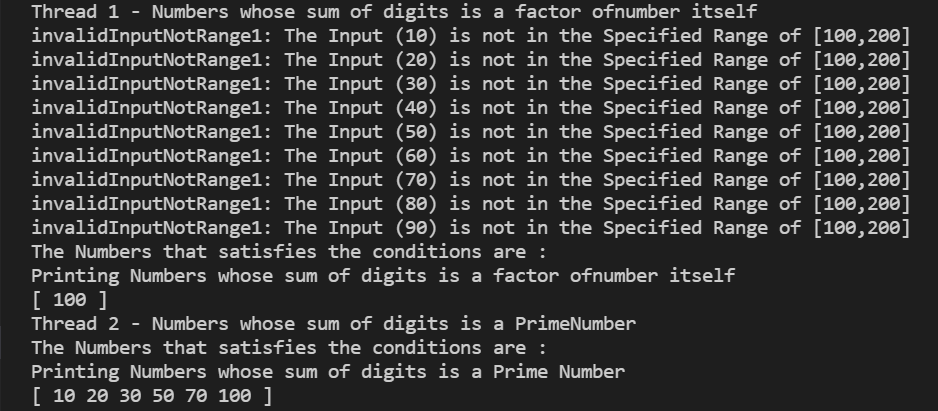
**Getting input:**

****

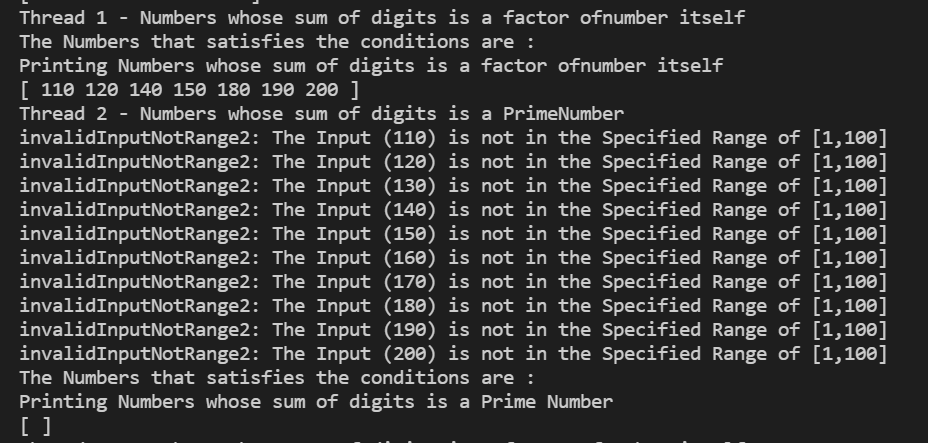
**We are sending an array like [10,20,30,40……..,300]**

**The first 10 numbers will first go to Thread1 and then to Thread2.**

* **Sending [10,20………,100] to Thread1 and Thread2:**



* **Sending [110,120………,200] to Thread1 and Thread2:**



* **Sending [210,220………,300] to Thread1 and Thread2:**

