**JFS TASK-4**

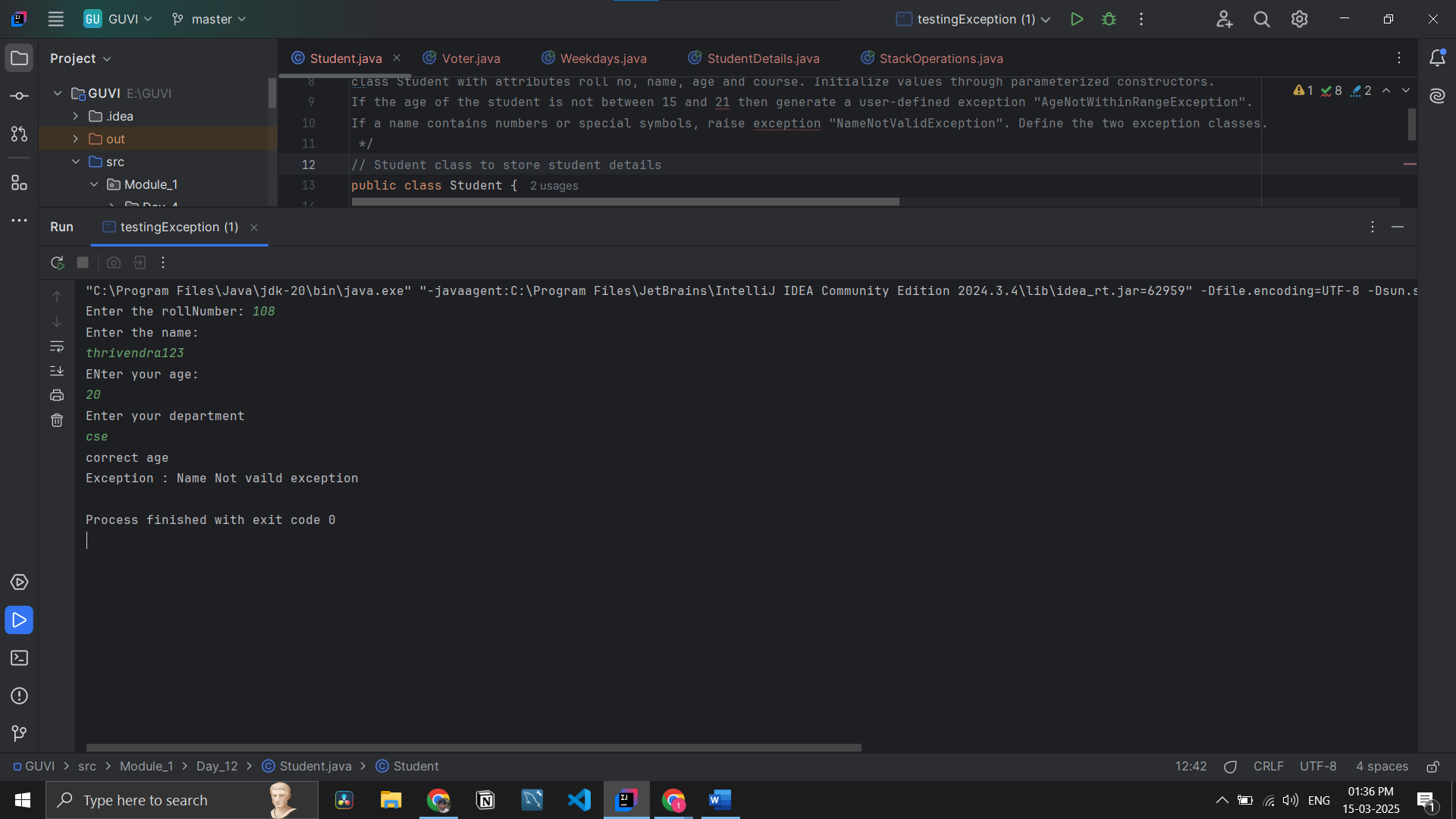
**QUESTION 1**

Q1. Ramesh is developing a student management system for a university. In this system, you have a Student class to represent student information. You are asked to help Ramesh to handle exception which can be occurred into program according to following Scenarios:  
  
class Student with attributes roll no, name, age and course. Initialize values through parameterized constructors.  
If the age of the student is not between 15 and 21 then generate a user-defined exception "AgeNotWithinRangeException".  
If a name contains numbers or special symbols, raise exception "NameNotValidException". Define the two exception classes.

**Code:**

package Module\_1.Day\_12;  
  
import java.util.\*;  
  
/\*  
Q1. Ramesh is developing a student management system for a university. In this system, you have a Student class to represent student information. You are asked to help Ramesh to handle exception which can be occurred into program according to following Scenarios:  
  
class Student with attributes roll no, name, age and course. Initialize values through parameterized constructors.  
If the age of the student is not between 15 and 21 then generate a user-defined exception "AgeNotWithinRangeException".  
If a name contains numbers or special symbols, raise exception "NameNotValidException". Define the two exception classes.  
 \*/  
// Student class to store student details  
public class Student {  
  
 public int rollNumber; // Stores roll number of the student  
 public String name; // Stores name of the student  
 public int age; // Stores age of the student  
 public String course; // Stores course of the student  
  
 // Parameterized constructor to initialize Student object  
  
 public Student(int rollNumber, String name, int age, String course) {  
 this.rollNumber = rollNumber;  
 this.name = name;  
 this.age = age;  
 this.course = course;  
 }  
  
  
}  
  
// Custom exception class for invalid age  
class AgeNotWithRangeException extends Exception  
{  
 public AgeNotWithRangeException(String message)  
 {  
 super(message); // Passing message to Exception class constructor  
 }  
}  
  
// Custom exception class for invalid name  
class NameNotVaildException extends Exception  
{  
 public NameNotVaildException(String message)  
 {  
 super(message); // Passing message to Exception class constructor  
 }  
}  
  
  
// Class to test exceptions  
class testingException  
{  
 // Method to check if age is within the valid range (15-21)  
 static void ageChecking(int age) throws AgeNotWithRangeException  
 {  
  
 if (age <15|| age>21)  
 {  
 throw new AgeNotWithRangeException(" age should be in 15 - 21");  
 }  
 System.*out*.println("correct age");  
 }  
  
 // Method to check if the name contains only alphabets  
 static void nameChacking(String name) throws NameNotVaildException  
 {  
 name=name.toLowerCase();  
  
 for (int i=0;i<name.length();i++)  
 {  
 char c=name.charAt(i);  
 if(c<'a' || c>'z')  
 {  
 throw new NameNotVaildException("Name Not vaild exception");  
 }  
 }  
 }  
  
 // Main method to test the functionality  
 public static void main(String[] args) {  
 Scanner input=new Scanner(System.*in*);  
  
 System.*out*.print("Enter the rollNumber: ");  
 int rollnumber=input.nextInt();  
 input.nextLine();  
  
 System.*out*.println("Enter the name: ");  
 String Name=input.nextLine();  
  
 System.*out*.println("ENter your age:");  
 int Age=input.nextInt();  
 input.nextLine();  
  
 System.*out*.println("Enter your department");  
  
 String course=input.nextLine();  
  
  
 try {  
 // Checking if age and name are valid  
 testingException.*ageChecking*(Age);  
 testingException.*nameChacking*(Name);  
  
 Student student=new Student(rollnumber,Name,Age,course);  
 System.*out*.println("Student registered successfully!");  
 } catch (AgeNotWithRangeException | NameNotVaildException e) {  
 System.*out*.println("Exception : "+e.getMessage()); // Handling exceptions  
 }  
  
 input.close(); // Closing Scanner to prevent resource leaks  
  
 }  
}

**Output Screen Shot**:



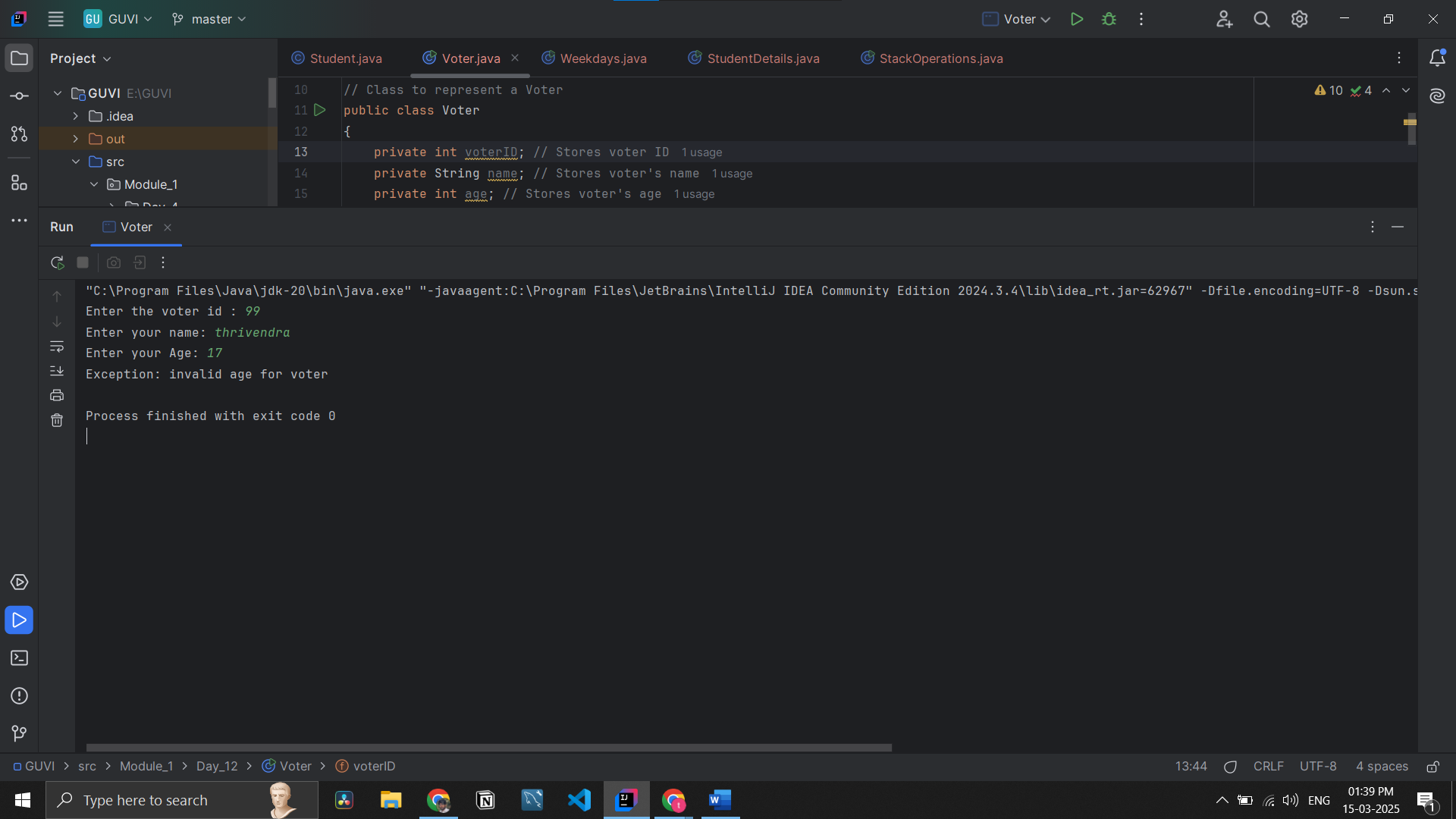
**QUESTION 2**

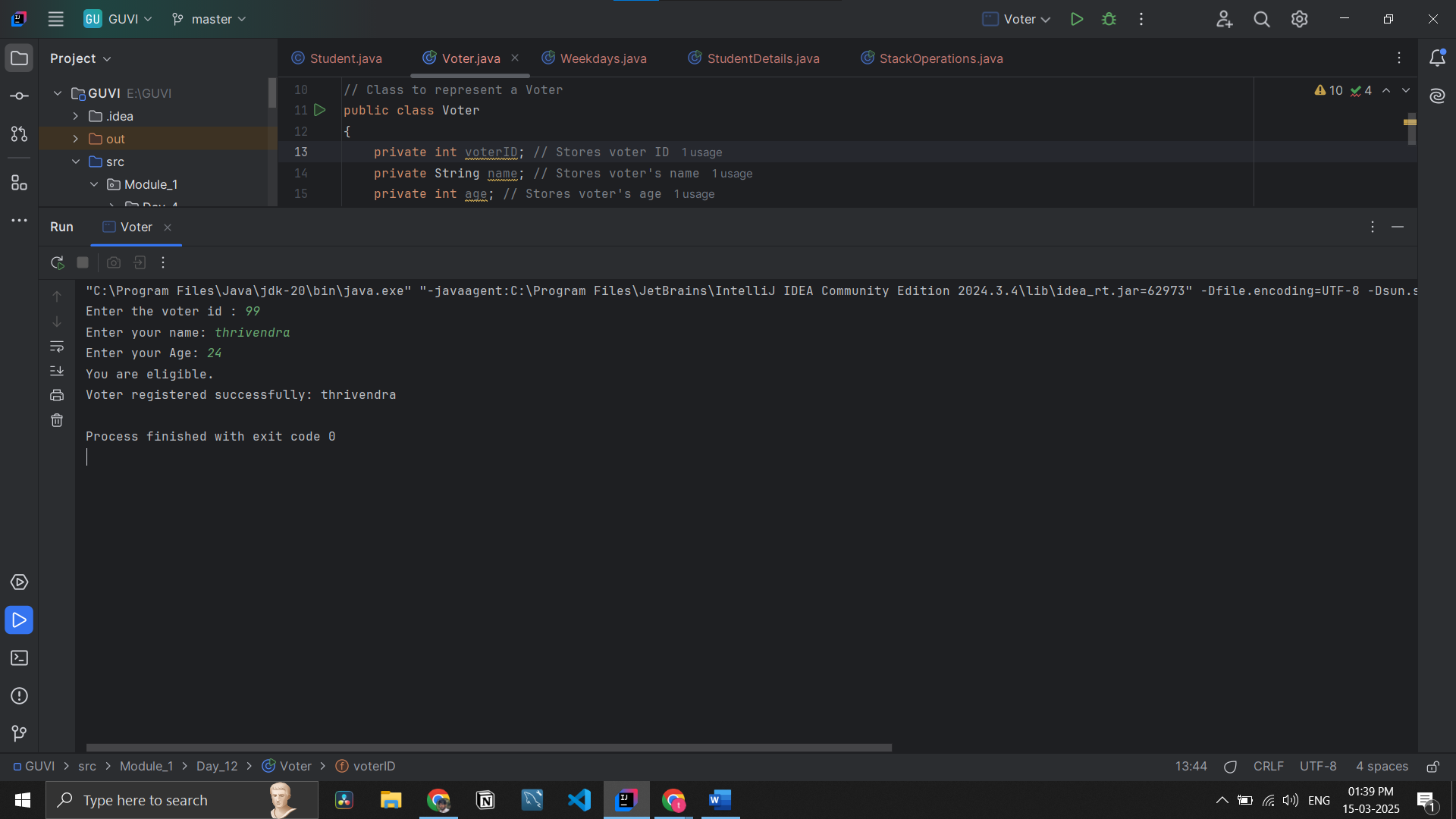
Q2. Create a class Voter(voterId, name, age) with parameterized constructor.  
 The parameterized constructor should throw a checked/Unchecked exception if age is less than 18.  
 The message of exception is “invalid age for voter

**Code :**

package Module\_1.Day\_12;  
  
import java.util.Scanner;  
  
/\*  
Q2. Create a class Voter(voterId, name, age) with parameterized constructor.  
 The parameterized constructor should throw a checked/Unchecked exception if age is less than 18.  
 The message of exception is “invalid age for voter  
 \*/  
// Class to represent a Voter  
public class Voter  
{  
 private int voterID; // Stores voter ID  
 private String name; // Stores voter's name  
 private int age; // Stores voter's age  
  
  
 // Parameterized Constructor with Exception Handling  
 public Voter(int voterID, String name, int age) {  
 this.voterID = voterID;  
 this.name = name;  
 this.age = age;  
 }  
  
 //checking the age  
 public static void checkage(int age) throws Exception  
 {  
 if(age<18)  
 {  
 throw new Exception("invalid age for voter");  
 }  
 else {  
 System.*out*.println("You are eligible.");  
 }  
 }  
  
 public static void main(String[] args) {  
 Scanner input=new Scanner(System.*in*);  
 // Take voter ID as input  
 System.*out*.print("Enter the voter id : ");  
 int voterid=input.nextInt();  
 input.nextLine(); // Consume the newline character after integer input  
  
 // Take name as input  
 System.*out*.print("Enter your name: ");  
 String Name=input.nextLine();  
  
 // Take age as input  
 System.*out*.print("Enter your Age: ");  
 int Age=input.nextInt();  
 input.nextLine();  
  
 //using try catch to checking the data  
 try {  
 Voter.*checkage*(Age);  
 Voter voter=new Voter(voterid,Name,Age);  
 System.*out*.println("Voter registered successfully: " + Name);  
 } catch (Exception e) {  
 System.*out*.println("Exception: "+e.getMessage());  
 }  
  
 input.close(); // Closing Scanner to prevent resource leaks  
 }  
}

**Output Screen Shot:**

****

****

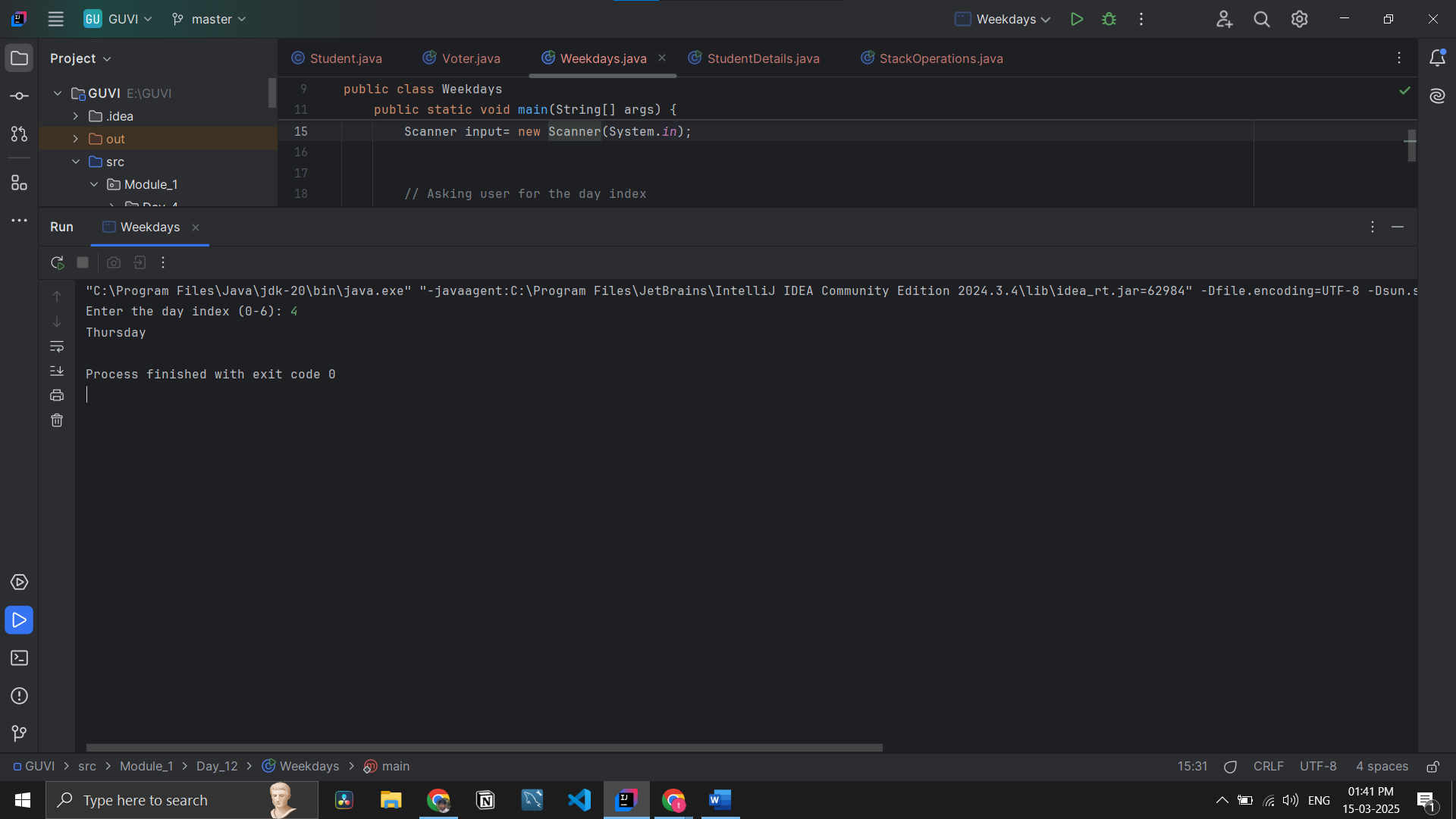
**QUESTION 3**

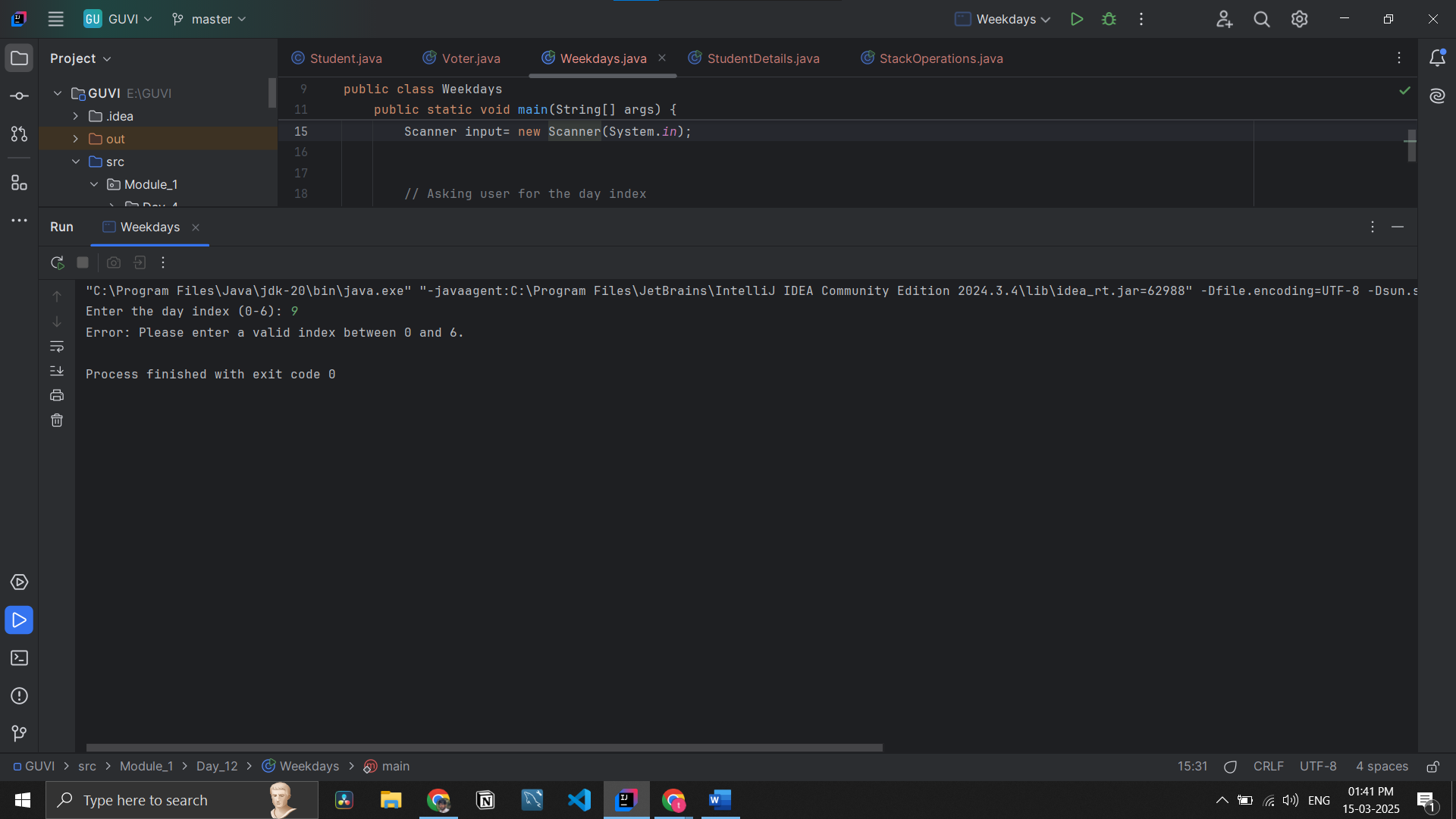
Q3.Store name of weekdays in an array (starting from “Sunday” at 0 index). Ask day position from user and print day name.  
 Handle array index out of bound exception and give a proper message if a user enters day index outside range (0-6).

**Code:**

package Module\_1.Day\_12;  
  
import java.util.Scanner;  
  
/\*  
Q3.Store name of weekdays in an array (starting from “Sunday” at 0 index). Ask day position from user and print day name.  
 Handle array index out of bound exception and give a proper message if a user enters day index outside range (0-6).  
 \*/  
public class Weekdays  
{  
 public static void main(String[] args) {  
 String [] weekdays={"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"};  
  
 // Array to store the weekdays (Starting from Sunday at index 0)  
 Scanner input= new Scanner(System.*in*);  
  
  
 // Asking user for the day index  
 System.*out*.print("Enter the day index (0-6): ");  
  
 try {  
 int index=input.nextInt();  
 System.*out*.println(weekdays[index]);  
 }catch (ArrayIndexOutOfBoundsException e)  
 {  
 System.*out*.println("Error: Please enter a valid index between 0 and 6.");  
 } catch (Exception e) {  
 System.*out*.println("Error: Invalid input. Please enter a number.");  
 }  
  
 }  
}

**Output Screen Shot:**

****

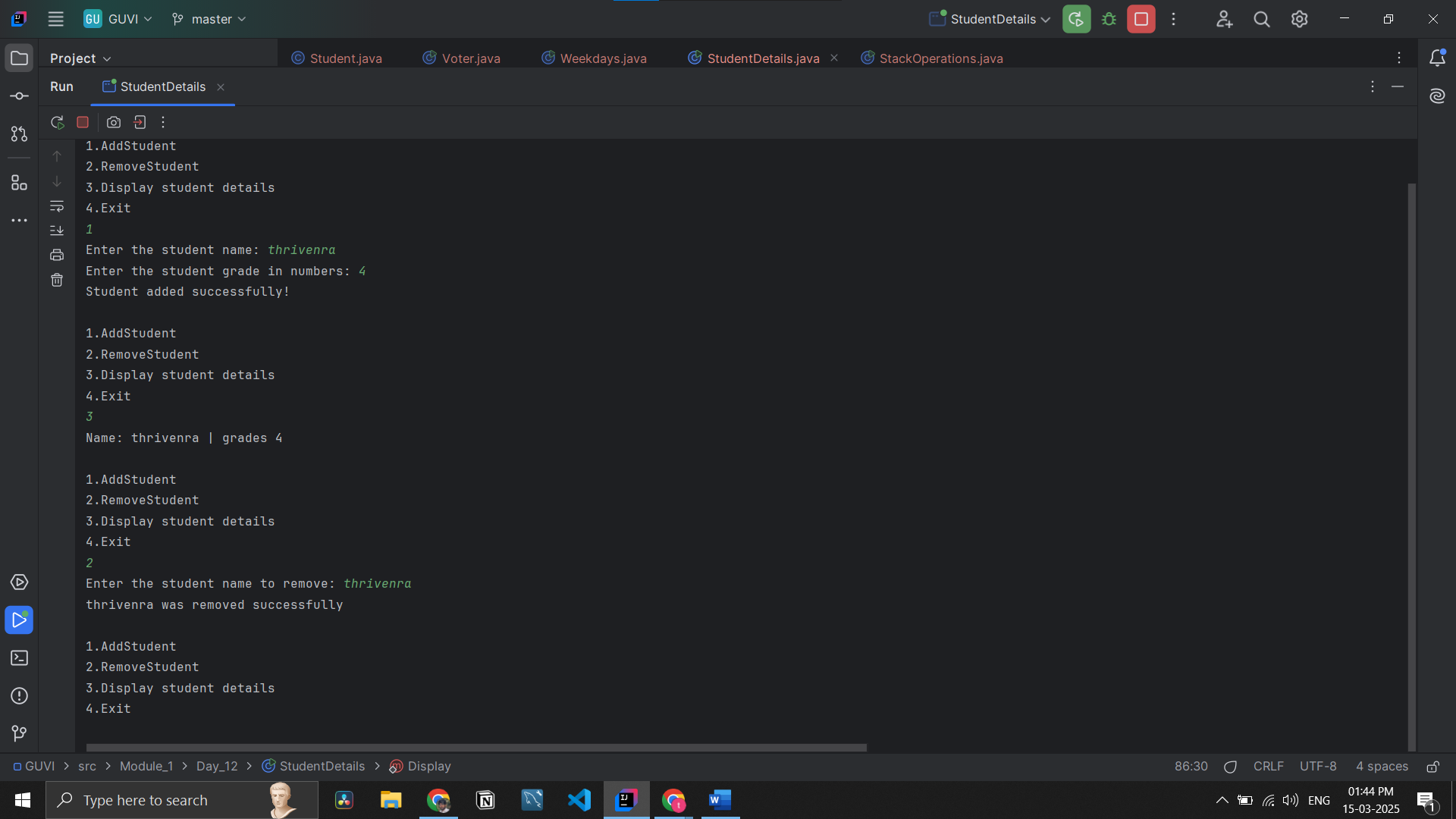


**QUESTION 4**

Q4. Create a HashMap where keys are student names (strings) and values are their corresponding grades (integers).  
Create methods to add a new student, remove a student, and Display up a student's grade by name.

**Code:**package Module\_1.Day\_12;  
  
import java.util.\*;  
  
/\*  
Q4. Create a HashMap where keys are student names (strings) and values are their corresponding grades (integers).  
Create methods to add a new student, remove a student, and Display up a student's grade by name.  
 \*/  
public class StudentDetails {  
 static Scanner *input*=new Scanner(System.*in*);  
 public static void main(String[] args) {  
  
 Map<String,Integer> student=new HashMap<>(); //storing the details of the students  
  
 boolean flag=true; // using the loop the while  
  
 while (flag)  
 {  
 //choice  
 System.*out*.println("\n" +  
 "1.AddStudent\n" +  
 "2.RemoveStudent\n" +  
 "3.Display student details\n" +  
 "4.Exit");  
  
 int choice=Integer.*parseInt*(*input*.nextLine());  
 switch (choice)  
 {  
 case 1:  
 {  
 *addStudent*(student); // calling the add function  
 break;  
 }  
 case 2:  
 {  
 *removeStudent*(student); //calling the remove function  
 break;  
 }  
 case 3:  
 {  
 *Display*(student); //calling the display  
 break;  
 }  
 case 4:  
 {  
 flag=false;  
 *input*.close();  
 break;  
 }  
 default:  
 {  
 System.*out*.println("Enter number between 1 to 4");  
 }  
 }  
 }  
 }  
  
 //adding the student details  
 public static void addStudent(Map<String, Integer> student)  
 {  
 System.*out*.print("Enter the student name: ");  
 String Name=*input*.nextLine();  
  
 System.*out*.print("Enter the student grade in numbers: ");  
 int grade=*input*.nextInt();  
 *input*.nextLine();  
  
  
 student.put(Name,grade);  
 System.*out*.println("Student added successfully!");  
 }  
  
 // removing the student detail  
 public static void removeStudent(Map<String, Integer> student)  
 {  
 System.*out*.print("Enter the student name to remove: ");  
 String RemoveName=*input*.nextLine();  
 student.remove(RemoveName);  
  
 System.*out*.println(RemoveName+" was removed successfully");  
 }  
  
 // printing the all data  
 public static void Display(Map<String, Integer> student)  
 {  
 if(student.isEmpty())  
 {  
 System.*out*.println("No student records available.");  
 }  
 else {  
 for (Map.Entry<String,Integer> data:student.entrySet())  
 {  
 System.*out*.println("Name: "+data.getKey()+" | grades "+data.getValue());  
 }  
 }  
 }  
}

**Output Screen Shot:**

****

**QUESTION 5**

Q5. Use Collection Classes to store Integers .Create some methods for following functionalities .  
a. Include functions for pushing elements onto the stack .  
b. popping elements from the stack.  
c.check stack is empty

**Code:**

package Module\_1.Day\_12;  
  
import java.util.\*;  
  
/\*  
Q5. Use Collection Classes to store Integers .Create some methods for following functionalities .  
a. Include functions for pushing elements onto the stack .  
b. popping elements from the stack.  
c.check stack is empty  
 \*/  
public class StackOperations  
{  
 static Scanner *input*=new Scanner(System.*in*);  
 public static void main(String[] args) {  
 Stack<Integer> stack=new Stack<>(); // storing the data  
  
 boolean flag=true; // using the loop the while  
  
 while (flag)  
 {  
 System.*out*.println("\n1. Push Element\n2. Pop Element\n3. Display Stack\n4.checkStack is Empty \n 5. Exit");  
 System.*out*.print("Enter your choice: ");  
 int choice = *input*.nextInt();  
  
 switch (choice)  
 {  
 case 1:{  
 *PushElement*(stack);  
 break;  
 }  
 case 2:  
 {  
 *PopElement*(stack);  
 break;  
 }  
 case 3:  
 {  
 *DisplayStack*(stack);  
 break;  
 }  
 case 4:  
 {  
 *checkStackisEmpty*(stack);  
 }  
 case 5:  
 {  
 flag=false;  
 *input*.close();  
 break;  
  
 }  
 default:  
 {  
 System.*out*.println("Enter number between 1 to 4");  
 }  
 }  
 }  
 }  
  
 private static void checkStackisEmpty(Stack<Integer> stack)  
 {  
 if(stack.isEmpty())  
 {  
 System.*out*.println("Stack was empty");  
 }  
 }  
  
 //removing the last element form stack  
 private static void PopElement(Stack<Integer> stack)  
 {  
 if(stack.isEmpty())  
 {  
 System.*out*.println("Stack is empty! Nothing to pop.");  
 }  
 else {  
 System.*out*.println(stack.pop()+" was removed successfully");  
 }  
  
 }  
  
 //printing all number form stacks  
 private static void DisplayStack(Stack<Integer> stack)  
 {  
 if (stack.isEmpty()) {  
 System.*out*.println("Stack is empty!");  
 } else {  
 System.*out*.println("Stack Elements: " + stack);  
 }  
 }  
  
 // adding numbers to stack  
 private static void PushElement(Stack<Integer> stack) {  
 System.*out*.print("Enter number to push: ");  
 int num = *input*.nextInt();  
 stack.push(num);  
 System.*out*.println("Number was add to stack successfully");  
 }  
}

**Output screen Shot:**  
