**1.PROGRAM CODE**

**STUDENT CLASS**

// Class representing a student with validation logic for age and name

public class Student {

// Instance variables to store student information  
 int roll\_no;  
 String name;  
 int age;  
 String course;  
 // Constructor to initialize student details  
 public Student(int roll\_no, String name, int age, String course) {  
 this.roll\_no = roll\_no;  
 this.name = name;  
 this.age = age;  
 this.course = course;  
 }  
 // Method to validate the student's age  
 // Throws custom exception if age is not within 15 to 21

public void validateAge(int age) throws AgeNotWithinRangeException {  
 if (age < 15 || age > 21) {  
 throw new AgeNotWithinRangeException("Age not valid");  
 } else {  
 System.out.println("Age is valid");  
 }  
 }  
  
 // Method to validate the student's name  
 // Throws custom exception if name contains anything other than letters and spaces  
 public void validateName(String name) throws NameNotValidException {  
 if (!name.matches("[a-zA-Z\\s]+")) {  
 throw new NameNotValidException("Name not valid");  
 } else {  
 System.out.println("Name is valid");  
 }  
 }  
  
 // Main method to demonstrate exception handling for student validation  
 public static void main(String[] args) {  
 // Creating a student object with invalid age and valid name  
 Student obj = new Student(100, "suhail", 12, "BCA");  
  
 try {  
 // Validating age and name  
 obj.validateAge(obj.age); // Expected to throw AgeNotWithinRangeException  
 obj.validateName(obj.name); // Will be validated only if age is valid  
 } catch (AgeNotWithinRangeException e) {  
 e.printStackTrace(); // Print stack trace if age is invalid  
 } catch (NameNotValidException e) {  
 e.printStackTrace(); // Print stack trace if name is invalid  
 }  
 }  
}

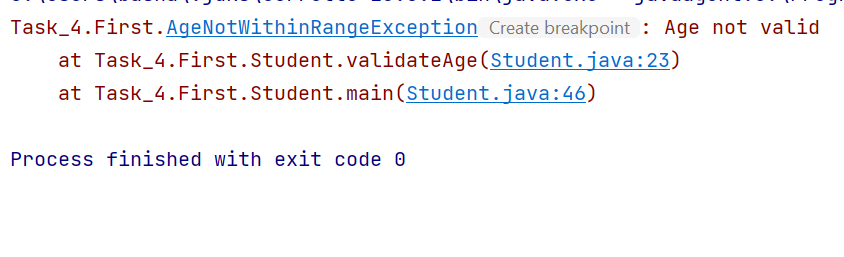
**NAMENOTVALIDEXCEPTION CLASS**

// Custom checked exception to handle invalid name input  
public class NameNotValidException extends Exception {  
  
 // Constructor that accepts a custom error message  
 public NameNotValidException(String message) {  
 super(message); // Pass the message to the parent Exception class  
 }  
}

**AGENOTWITHINRANGEEXCEPTION CLASS**

// Custom checked exception to handle age validation errors  
public class AgeNotWithinRangeException extends Exception {  
  
 // Constructor that accepts a custom error message  
 public AgeNotWithinRangeException(String message) {  
 super(message); // Pass the message to the parent Exception class  
 }  
}

**OUTPUT**

****

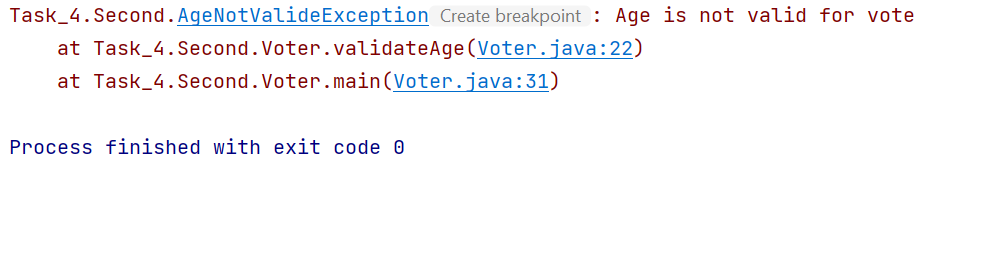
**2.PROGRAM CODE**

**CLASS VOTER**  
public class Voter {  
  
 int voterId;  
 String name;  
 int age;  
  
 // Constructor to initialize a Voter object  
 public Voter(int voterId, String name, int age) {  
 this.voterId = voterId;  
 this.name = name;  
 this.age = age;  
 }  
  
 // Method to validate if the voter's age is eligible  
 public void validateAge(int age) throws AgeNotValideException {  
 if (age >= 18) {  
 System.out.println("Age is valid for vote");  
 } else {  
 // Throw custom exception if age is below 18  
 throw new AgeNotValideException("Age is not valid for vote");  
 }  
 }  
  
  
 public static void main(String[] args) {  
 // Creating a Voter object with valid age  
 Voter obj = new Voter(1, "Suresh", 18);  
 try {  
 obj.validateAge(obj.age);  
 } catch (AgeNotValideException e) {  
  
 e.printStackTrace();  
 }  
 }  
}

**AGENOTVALIDEEXCEPTION CLASS**

public class AgeNotValideException extends Exception {  
  
 // Constructor that accepts a custom error message  
 AgeNotValideException(String message) {  
 super(message); // Pass the message to the parent Exception class  
 }  
}

**OUTPUT**

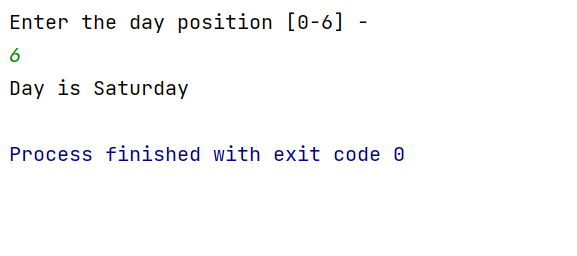
****

**3.PROGRAM CODE**

**DAYS CLASS**

public class Days {  
  
 public static void main(String[] args) {  
  
 // Array containing names of the days of the week  
 String[] weekdays = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"};  
  
 // Scanner object to take user input  
 Scanner obj = new Scanner(System.in);  
  
 System.out.println("Enter the day position [0-6] - ");  
 int day = obj.nextInt();  
  
 try {  
  
 System.out.println("Day is " + weekdays[day]);  
 } catch (ArrayIndexOutOfBoundsException e) {  
 e.printStackTrace();  
 }  
 obj.close();  
 }  
}

**OUTPUT**

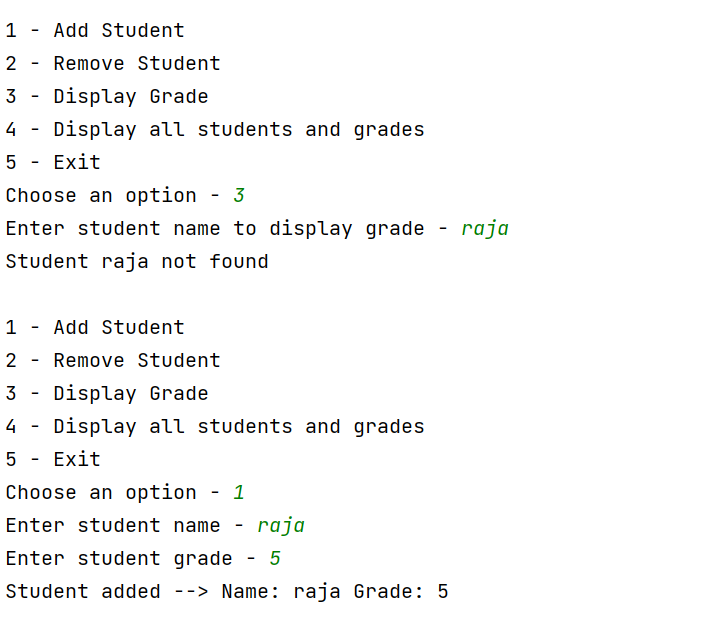
****

**4.PROGRAM CODE**

**STUDENTHASHMAP CLASS**

import java.util.HashMap;  
import java.util.Map;  
import java.util.Scanner;  
  
public class StudentHashMap {  
  
  
 static HashMap<String, Integer> studentmap = new HashMap<>();  
  
 // Method to add a student and their grade to the map  
 public static void add(String name, int grade) {  
 studentmap.put(name, grade);  
 System.out.println("Student added --> Name: " + name + " Grade: " + grade);  
 }  
  
 // Method to remove a student by name  
 public static void remove(String name) {  
 if (studentmap.containsKey(name)) {  
 studentmap.remove(name);  
 System.out.println("Student removed - " + name);  
 } else {  
 System.out.println("Student " + name + " not found");  
 }  
 }  
  
 // Method to display the grade of a specific student  
 public static void displayGrade(String name) {  
 if (studentmap.containsKey(name)) {  
 System.out.println("Grade of " + name + ": " + studentmap.get(name));  
 } else {  
 System.out.println("Student " + name + " not found");  
 }  
 }  
  
 // Method to display all students and their grades  
 public static void display() {  
 if (studentmap.isEmpty()) {  
 System.out.println("No Students Available");  
 } else {  
 for (Map.Entry<String, Integer> m : studentmap.entrySet()) {  
 System.out.println(m.getKey() + ": " + m.getValue());  
 }  
 }  
 }  
  
 public static void main(String[] args) {  
 Scanner obj = new Scanner(System.in);  
 int choice;  
  
 do {  
 System.out.println("\n1 - Add Student");  
 System.out.println("2 - Remove Student");  
 System.out.println("3 - Display Grade");  
 System.out.println("4 - Display all students and grades");  
 System.out.println("5 - Exit");  
 System.out.print("Choose an option - ");  
 choice = obj.nextInt();  
 obj.nextLine();  
  
 switch (choice) {  
 case 1:  
 System.out.print("Enter student name - ");  
 String name = obj.nextLine();  
 System.out.print("Enter student grade - ");  
 int grade = obj.nextInt();  
 StudentHashMap.add(name, grade);  
 break;  
  
 case 2:  
 System.out.print("Enter student name to remove - ");  
 String remove = obj.nextLine();  
 StudentHashMap.remove(remove);  
 break;  
  
 case 3:  
 System.out.print("Enter student name to display grade - ");  
 String namegrade = obj.nextLine();  
 StudentHashMap.displayGrade(namegrade);  
 break;  
  
 case 4:  
 StudentHashMap.display();  
 break;  
  
 case 5:  
 System.out.println("Program exited.");  
 break;  
  
 default:  
 System.out.println("Enter a valid choice");  
 }  
 } while (choice != 5);  
 obj.close();  
 }  
}

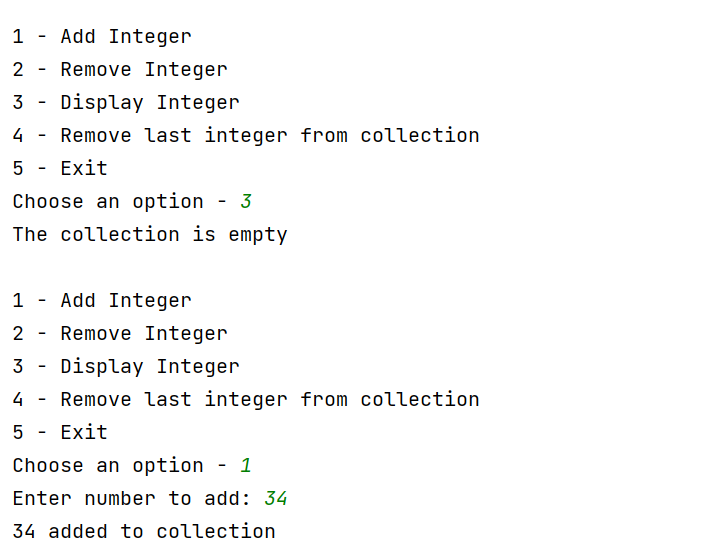
**OUTPUT**

****

**5.PROGRAM CODE**

import java.util.Scanner;  
import java.util.Stack;  
  
  
public class StackCollection {  
  
 static Stack<Integer> stack = new Stack<>();  
  
 // Method to add an integer to the stack  
 public static void add(int num) {  
 stack.push(num); // Pushes the number onto the stack  
 System.out.println(num + " added to collection");  
 }  
  
 // Method to remove a specific integer  
 public static void remove(int num) {  
 // Check if the stack contains the specified number  
 if (stack.contains(num)) {  
 stack.remove(stack.indexOf(num));  
 System.out.println("removed - " + num);  
 } else {  
 System.out.println(num + " not found");  
 }  
 }  
  
 // Method to remove the last on the stack  
 public static void removeLast() {  
 if (stack.isEmpty()) {  
 System.out.println("The collection is empty");  
 } else {  
 // Pop removes and returns the top element of the stack  
 System.out.println("Removed last element - " + stack.pop());  
 }  
 }  
  
 // Method to display the current elements of the stack  
 public static void display() {  
 if (stack.isEmpty()) {  
 System.out.println("The collection is empty");  
 } else {  
 // Directly prints the elements of the stack  
 System.out.println("The collection is - " + stack);  
 }  
 }  
  
 public static void main(String[] args) {  
 Scanner obj = new Scanner(System.in);  
 int choice, num;  
  
 do {  
  
 System.out.println("\n1 - Add Integer");  
 System.out.println("2 - Remove Integer");  
 System.out.println("3 - Display Integer");  
 System.out.println("4 - Remove last integer from collection");  
 System.out.println("5 - Exit");  
 System.out.print("Choose an option - ");  
  
 // Reading user input  
 choice = obj.nextInt();  
 obj.nextLine();  
  
 // Switch case to handle user choice  
 switch (choice) {  
 case 1:  
 System.out.print("Enter number to add: ");  
 num = obj.nextInt();  
 StackCollection.add(num);  
 break;  
 case 2:  
 System.out.print("Enter number to remove: ");  
 num = obj.nextInt();  
 StackCollection.remove(num);  
 break;  
 case 3:  
 StackCollection.display();  
 break;  
 case 4:  
 StackCollection.removeLast();  
 break;  
 case 5:  
 System.out.println("Exiting");  
 break;  
 default:  
 System.out.println("Enter the correct choice");  
 }  
  
 } while (choice != 5);  
  
 obj.close(); // Close the scanner  
 }  
}

**OUTPUT**

****