**Java Advance**

**Exceptions & Collections**

**Q1) Student Management System**

**A.Student Class Creation with Attributes and Parameterized Constructor**

**// Student Class Creation with Attributes**

**public class Student{**

**int rollNo;**

**String name;**

**int age;**

**String course;**

**// Parameterized Constructor for Student Class**

**public Student(int rollNo, String name, int age, String course)throws AgeNotWithinRangeException,NameNotValidException {**

**if(age<15 || age>21){**

**throw new AgeNotWithinRangeException("AgeNotWithinRangeException!");**

**}**

**if (!name.matches("[a-zA-z ]+")){**

**throw new NameNotValidException("NameNotValidException!");**

**}**

**this.rollNo = rollNo;**

**this.name = name;**

**this.age = age;**

**this.course = course;**

**}**

**B.Age Restriction between 15 to 21 and Exception Handling**

**if(age<15 || age>21){**

**throw new AgeNotWithinRangeException("AgeNotWithinRangeException!");**

**}**

**Exception For Array index out of Bound Exception**

**// Generating Age not in Range Exception**

**class AgeNotWithinRangeException extends Exception{**

**public AgeNotWithinRangeException(String message){**

**super(message);**

**}**

**}**

**C. Name Restriction only Contains Strings(No numbers and Symbols)**

**if (!name.matches("[a-zA-z ]+")){**

**throw new NameNotValidException("NameNotValidException!");**

**}**

**Exception For Name Not valid Exception**

**// Generating Name not valid Exception**

**class NameNotValidException extends Exception{**

**public NameNotValidException(String message){**

**super(message);**

**}**

**}**

**Catching the Exceptions**

**// Catching the Age not in Range Exception**

**catch (AgeNotWithinRangeException | NameNotValidException e){**

**System.*out*.println("Exception: "+e.getMessage());**

**}**

**// Catching the Name Not Valid Exception**

**catch (Exception e){**

**System.*out*.println("Invalid Input!!");**

**}**

**OUT PUT :**

**Execution WithOut Exception:**

Enter Student RollNo:

1

Enter Student Name:

Stark

Enter Student age:

18

Enter Student Course:

CSE

**RollNo: 1**

**Name: Stark**

**Age: 18**

**Course: CSE**

**Execution with AgeNotInRange Exception :**

Enter Student RollNo:

1

Enter Student Name:

Stark

Enter Student age:

12

Enter Student Course:

CSE

**Exception: AgeNotWithinRangeException!**

**Execution with NameNotValid Exception :**

Enter Student RollNo:

1

Enter Student Name:

Star24#

Enter Student age:

18

Enter Student Course:

CSE

**Exception: NameNotValidException!**

**Q2.**

**A.Voter class Creation with Attributes and Parameterized Constructor**

**// Voter Class Creation with Attributes**

**public class Voter {**

**int voterId;**

**String name;**

**int age;**

**// Parameterized Constructor for Voter Classs**

**public Voter(int voterId, String name, int age) throws VoterAge {**

**if (age<18){**

**throw new VoterAge("Invalid Age of Voter!");**

**}**

**this.voterId = voterId;**

**this.name = name;**

**this.age = age;**

**}**

**B.Age Restriction Exception, Age must be above 18**

**Exception Handling**

**if (age<18){**

**throw new VoterAge("Invalid Age of Voter!");**

**}**

**Checked Exception**

**// Checked Exception**

**class VoterAge extends Exception{**

**public VoterAge (String message){**

**super(message);**

**}**

**}**

**OUT PUT :**

**Input Without Exception:**

Enter voter Id:

1

Enter voter Name:

john

Enter voter Age:

18

Voter Id: 1

Voter Name: john

Voter Age: 18

**Input With Exception:**

Enter voter Id:

1

Enter voter Name:

john

Enter voter Age:

11

Exception: Invalid Age of Voter!

**Q3.**

**A.Storing Weekdays in Array Array Range is (0-6)**

**// Array for Storing WeekDays**

**String[]weekdays={"Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday"};**

**B.Get Day Position from User For Print Day**

**try {**

**// Getting Day position for printing Day**

**System.*out*.println("Enter the Day Position: ");**

**int position = scanner.nextInt();**

**System.*out*.println("The Day is: "+weekdays[position]);**

**}**

**C.Handling Array Index Out Of Bound Exception**

**// Handling Array out of Bound Exception**

**catch (ArrayIndexOutOfBoundsException e){**

**System.*out*.println("Error: Enter Valid Position!");**

**}**

**OUT PUT :**

**Input Without Exception:**

Enter the Day Position:

1

The Day is: Monday

**Input With Exception:**

Enter the Day Position:

7

Error: Enter Valid Position!

**Q4.**

**A.HashMap Creation with Key:Name, Values:grades**

**// Hashmap Creation**

**HashMap<String, Integer> studentGrades = new HashMap<>();**

**Method for Add New Student:**

**// Method to add a student**

**public void addStudent(String name, int grade) {**

**studentGrades.put(name, grade);**

**System.*out*.println(name + " added with grade: " + grade);**

**}**

**Method for Remove Student:**

**// Method to remove a student**

**public void removeStudent(String name) {**

**if (studentGrades.containsKey(name)) {**

**studentGrades.remove(name);**

**System.*out*.println(name + " removed from the system.");**

**} else {**

**System.*out*.println(name + " not found.");**

**}**

**}**

**Method for Display Student’s name by grade:**

**// Method to display a student's grade by name**

**public void displayStudentGrade(String name) {**

**if (studentGrades.containsKey(name)) {**

**System.*out*.println(name + "'s grade: " + studentGrades.get(name));**

**} else {**

**System.*out*.println(name + " not found.");**

**}**

**}**

**OUT PUT :**

Student Management System:

1. Add Student

2. Remove Student

3. Display Student Grade

4. Exit

Enter your choice: 1

Enter student name: John

Enter student grade: 1

John added with grade: 1

Student Management System:

1. Add Student

2. Remove Student

3. Display Student Grade

4. Exit

Enter your choice: 1

Enter student name: John

Enter student grade: 2

John added with grade: 2

Student Management System:

1. Add Student

2. Remove Student

3. Display Student Grade

4. Exit

Enter your choice: 1

Enter student name: Stark

Enter student grade: 3

Stark added with grade: 3

Student Management System:

1. Add Student

2. Remove Student

3. Display Student Grade

4. Exit

Enter your choice: 3

Enter student name to display grade: john

john not found.

Student Management System:

1. Add Student

2. Remove Student

3. Display Student Grade

4. Exit

Enter your choice: 3

Enter student name to display grade: John

John's grade: 2

Student Management System:

1. Add Student

2. Remove Student

3. Display Student Grade

4. Exit

Enter your choice: 2

Enter student name to remove: 1

1 not found.

Student Management System:

1. Add Student

2. Remove Student

3. Display Student Grade

4. Exit

Enter your choice: 4

Exiting...

**Q5.Stack Class Creation**

**// Stack class Creation**

**Stack<Integer>stack=new Stack<>();**

**A.Method for Pushing Elements on the Stack**

**// Method For Push Elements in the Stack**

**public void pushElements(int number){**

**stack.push(number);**

**System.*out*.println(number+" Pushed into the Stack");**

**}**

**B.Method for Popping Elements on the Stack**

**// Method For Pop Elements in the Stack**

**public void PopElements(){**

**if (!stack.isEmpty()){**

**int value=stack.pop();**

**System.*out*.println(value+" Popped from the Stack");**

**}else{**

**System.*out*.println("Stack is empty");**

**}**

**}**

**C.Method for Checking Stack Empty or not**

**// Method For checking stack empty or not**

**public void isEmpty(){**

**if (stack.isEmpty()){**

**System.*out*.println("The Stack is Empty!");**

**}else {**

**System.*out*.println("The Stack is not Empty!");**

**}**

**}**

**OUTPUT :**

1. Push Elements

2. Pop Elements

3. Display Stack

4. isEmpty

5. Exit

Enter your choice: 1

Enter the Number for Push:

10

10 Pushed into the Stack

1. Push Elements

2. Pop Elements

3. Display Stack

4. isEmpty

5. Exit

Enter your choice: 1

Enter the Number for Push:

20

20 Pushed into the Stack

1. Push Elements

2. Pop Elements

3. Display Stack

4. isEmpty

5. Exit

Enter your choice: 1

Enter the Number for Push:

30

30 Pushed into the Stack

1. Push Elements

2. Pop Elements

3. Display Stack

4. isEmpty

5. Exit

Enter your choice: 1

Enter the Number for Push:

40

40 Pushed into the Stack

1. Push Elements

2. Pop Elements

3. Display Stack

4. isEmpty

5. Exit

Enter your choice: 1

Enter the Number for Push:

50

50 Pushed into the Stack

1. Push Elements

2. Pop Elements

3. Display Stack

4. isEmpty

5. Exit

Enter your choice: 3

[10, 20, 30, 40, 50]

1. Push Elements

2. Pop Elements

3. Display Stack

4. isEmpty

5. Exit

Enter your choice: 4

The Stack is not Empty!

1. Push Elements

2. Pop Elements

3. Display Stack

4. isEmpty

5. Exit

Enter your choice: 2

50 Popped from the Stack

1. Push Elements

2. Pop Elements

3. Display Stack

4. isEmpty

5. Exit

Enter your choice: 2

40 Popped from the Stack

1. Push Elements

2. Pop Elements

3. Display Stack

4. isEmpty

5. Exit

Enter your choice: 2

30 Popped from the Stack

1. Push Elements

2. Pop Elements

3. Display Stack

4. isEmpty

5. Exit

Enter your choice: 2

20 Popped from the Stack

1. Push Elements

2. Pop Elements

3. Display Stack

4. isEmpty

5. Exit

Enter your choice: 2

10 Popped from the Stack

1. Push Elements

2. Pop Elements

3. Display Stack

4. isEmpty

5. Exit

Enter your choice: 2

Stack is empty

1. Push Elements

2. Pop Elements

3. Display Stack

4. isEmpty

5. Exit

Enter your choice: 5

Exiting....