Code forQ1

public class AgeNotWithRangeException extends Exception {  
 String error;  
 AgeNotWithRangeException(String error)  
 {  
  
 super(error);  
 }  
}

public class NameNotValidException extends Exception {  
 String error;  
 NameNotValidException(String error)  
 {  
 super(error);  
 }  
}

public class Student {  
  
 int rollno;  
 String name;  
 int age;  
 String course;  
  
 public Student(int rollno, String name, int age, String course)  
  
 throws AgeNotWithRangeException, NameNotValidException {  
  
 if (age < 15 || age > 21) {  
 throw new AgeNotWithRangeException("Age of the student must be between 15 to 21");  
 }  
 if (!name.matches("[a-zA-z\\s]+")) {  
 throw new NameNotValidException("Name shouldn't contain numbers or special characters");  
  
 }  
  
  
 this.rollno = rollno;  
 this.name = name;  
 this.age = age;  
 this.course = course;  
 }  
  
 public void display() {  
  
 System.*out*.println("Rollno:"+rollno);  
 System.*out*.println("Name of the student:"+name);  
 System.*out*.println("Age:"+age);  
 System.*out*.println("Course:"+course);  
 }  
  
  
 public static void main(String[] args) {  
  
  
 try {  
 Student s1 = new Student(1, "Jyothi", 18, "CSE");  
 s1.display();  
 Student s2 = new Student(2, "Aadhya@11", 18, "ECE");  
 } catch (AgeNotWithRangeException | NameNotValidException e) {  
 System.*out*.println(e.getMessage());  
 }  
 try {  
 Student s3 = new Student(3, "Dharani", 10, "EEE");  
  
 } catch (AgeNotWithRangeException | NameNotValidException e1) {  
 System.*out*.println(e1.getMessage());  
 }  
 }  
}

Output:

Rollno:1

Name of the student:Jyothi

Age:18

Course:CSE

Name shouldn't contain numbers or special characters

Age of the student must be between 15 to 21

Code for Q2:

public class InvalidvoterAgeException  
 extends Exception {  
  
 String message;  
 InvalidvoterAgeException(String message) {  
 super(message);  
 }  
}

public class VoterAge {  
  
 private String name;  
 private int age;  
  
 // Parameterized Constructor  
 public VoterAge(String name, int age ) throws InvalidvoterAgeException  
 {  
  
 if(age<18)  
 throw new InvalidvoterAgeException("invalid age for voter");  
  
  
 this.name=name;  
 this.age=age;  
  
 }  
  
 public void displayDetails(){  
  
 System.*out*.println("voter name:"+this.name);  
 System.*out*.println("voter age:"+this.age);  
 }  
  
  
  
 public static void main(String[] args) {  
  
  
 try {  
 VoterAge v1 = new VoterAge("Jyothi", 20);  
 v1.displayDetails();  
 }catch (InvalidvoterAgeException e) {  
 System.*out*.println(e.getMessage());  
 }try {  
 VoterAge v2 = new VoterAge("Dharani", 15);  
 v2.displayDetails();  
 }catch (InvalidvoterAgeException e) {  
 System.*out*.println(e.getMessage());  
 }try{  
 VoterAge v3=new VoterAge("Dharshi",21);  
 v3.displayDetails();  
 } catch (InvalidvoterAgeException e) {  
 System.*out*.println(e.getMessage());  
 }  
  
  
 }  
}

Output:

voter name:Jyothi

voter age:20

invalid age for voter

voter name:Dharshi

voter age:21

Code for Q3:

import java.util.Scanner;  
  
public class Weekday {  
  
  
  
 public static void main(String[] args) {  
 // Array of weekdays  
 String[] weekdays = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"};  
  
 Scanner scanner = new Scanner(System.*in*);  
  
 try {  
 // input for the day position  
 System.*out*.print("Enter the day index (0-6): ");  
 int index = scanner.nextInt();  
  
  
 System.*out*.println("The day is: " + weekdays[index]);  
 } catch (ArrayIndexOutOfBoundsException e) {  
 // array index out-of-bounds exception  
 System.*out*.println("Error: Please enter a number between 0 and 6.");  
 }  
 scanner.close();  
 }  
  
  
}

Output:

Enter the day index (0-6): 5

The day is: Friday

Process finished with exit code 0

Code for Q4:

import java.util.\*;  
  
public class StudentGrade {  
 // HashMap to store student names and grades  
 private HashMap<String, Integer> studentGrades;  
  
 // Constructor  
 public StudentGrade() {  
 studentGrades = new HashMap<>();  
 }  
  
 // Method to add a new student  
 public void addStudent(String name, int grade) {  
 studentGrades.put(name, grade);  
 System.*out*.println("Student added: " + name + " with grade " + grade);  
 }  
  
 // Method to remove a student  
 public void removeStudent(String name) {  
 if (studentGrades.containsKey(name)) {  
 studentGrades.remove(name);  
 System.*out*.println("Student removed: " + name);  
 } else {  
 System.*out*.println("Student not found: " + name);  
 }  
 }  
  
 // Method to display students sorted by grade  
 public void displayStudents() {  
 studentGrades.entrySet().stream()  
 .sorted(Map.Entry.*comparingByValue*(Comparator.*reverseOrder*())) // Sort by grade descending  
 .forEach(entry -> System.*out*.println(entry.getKey() + ": " + entry.getValue()));  
 }  
  
 public static void main(String[] args) {  
 StudentGrade manager = new StudentGrade();  
 manager.addStudent("Jyothi", 90);  
 manager.addStudent("Dharani", 91);  
 manager.addStudent("Dharshi", 72);  
 System.*out*.println("Students sorted by grades:");  
 manager.displayStudents();  
 manager.removeStudent("Dharani");  
 System.*out*.println("Students");  
 manager.displayStudents();  
 }  
}

Output:

Student added: Jyothi with grade 90

Student added: Dharani with grade 91

Student added: Dharshi with grade 72

Students sorted by grades:

Dharani: 91

Jyothi: 90

Dharshi: 72

Student removed: Dharani

Students

Jyothi: 90

Dharshi: 72

Process finished with exit code 0

Code for Q5:

import java.util.\*;  
  
public class StackManager {  
 private Stack<Integer> stack;  
  
 // Constructor to initialize the stack  
 public StackManager() {  
 stack = new Stack<>();  
 }  
  
 // Method to push an element into the stack  
 public void pushElement(int element) {  
 stack.push(element);  
 System.*out*.println("Pushed: " + element);  
 }  
  
 // Method to pop an element from the stack  
 public void popElement() {  
 if (stack.isEmpty()) {  
 System.*out*.println("Stack is empty. Cannot pop.");  
 } else {  
 int poppedElement = stack.pop();  
 System.*out*.println("Popped: " + poppedElement);  
 }  
 }  
  
 // Method to display all elements in the stack  
 public void isStackEmpty() {  
 if (stack.isEmpty()) {  
 System.*out*.println("Stack is empty.");  
 } else {  
 System.*out*.println("Stack elements: " + stack);  
 }  
 }  
  
 // Method to peek at the top element without removing it  
  
  
 public static void main(String[] args) {  
 StackManager stackManager = new StackManager();  
  
 // Push elements  
 stackManager.pushElement(1);  
 stackManager.pushElement(2);  
 stackManager.pushElement(3);  
 stackManager.pushElement(4);  
 stackManager.isStackEmpty();  
 stackManager.popElement();  
 stackManager.popElement();  
 stackManager.popElement();  
 stackManager.popElement();  
 stackManager.isStackEmpty();  
  
  
 }  
}

Output:

Pushed: 1

Pushed: 2

Pushed: 3

Pushed: 4

Stack elements: [1, 2, 3, 4]

Popped: 4

Popped: 3

Popped: 2

Popped: 1

Stack is empty.

Process finished with exit code 0