

Solution 1:

```
package task4;
```

```
// Custom exception for age not within the range
```

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

```
// Custom exception for invalid name
```

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

```
// Student class
```

```
class Student {  
    int rollNo;  
    String name;  
    int age;  
    String course;
```

```
// Parameterized constructor
```

```
public Student(int rollNo, String name, int age, String course)  
    throws AgeNotWithinRangeException, NameNotValidException {  
    if (age < 15 || age > 21) {  
        throw new AgeNotWithinRangeException("AgeNotWithinRangeException");  
    }
```

```
    else if (!name.matches("[a-zA-Z]+")) {  
        throw new NameNotValidException("NameNotValidException");  
    }
```

```
    this.rollNo = rollNo;  
    this.name = name;  
    this.age = age;  
    this.course = course;
```

```

    }

    // Method to display student details
    public void displayStudentDetails() {
        System.out.println("Roll No: " + rollNo + ", Name: " + name
            + ", Age: " + age + ", Course: " + course);
    }
}

// Main class to test the implementation
class Main {
    public static void main(String[] args) {
        try {
            // Valid student
            Student student1 = new Student(1, "Vamsi", 19, "JFSD");
            student1.displayStudentDetails();

        } catch (AgeNotWithinRangeException e) {
            System.out.println("Exception: " + e.getMessage());
        } catch (NameNotValidException e) {
            System.out.println("Exception: " + e.getMessage());
        }
    }
}

```

Output:

NameNotValidException	//output for-name given wrong
AgeNotWithinRangeException	//output for-age not given within range
Roll No: 1, Name: Vamsi, Age: 19, Course: JFSD	//if both are correct

Solution 2:

```

package task4;

class Voting {
    private int voterId;
    private String name;
    private int age;
}

```

```

// Parameterized constructor
public Voting(int voterId, String name, int age) throws Exception {
    if (age < 18) {
        throw new Exception("Invalid age for Voter");
    }
    this.voterId = voterId;
    this.name = name;
    this.age = age;
}

// Getter methods for the fields
public int getVoterId() {
    return voterId;
}

public String getName() {
    return name;
}

public int getAge() {
    return age;
}

public static void main(String[] args) {
    try {
        // Attempt to create a voter with invalid age
        Voting voter = new Voting(1, "Ramesh", 10);
        System.out.println("Voter is Eligible " + voter.getName());
    } catch (Exception e) {
        // This block executes if age is less than 18
        System.out.println("Exception: " + e.getMessage());
    }
}

```

Output:

Invalid age for Voter

Solution 3:

```

package task4;

public class Solution3 {
    static String []
weeknames={"Sunday","Monday","Thursday","Wednesday","Friday","Saturday"};
    public String getMessage(){
        return "Please Enter the Index range(0-6) only";
    }
    public static void main(String[] args) {
        Solution3 solution3 = new Solution3();
        try {
            System.out.println(Solution3.weeknames[7]);
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println(solution3.getMessage());
        }
    }
}

```

Output:

Please Enter the Index range(0-6) only

Solution 4:

```

package task4;

import java.util.HashMap;
import java.util.Scanner;

public class StudentGrades {

    private HashMap<String, Integer> studentGrades;
    public StudentGrades() {
        studentGrades = new HashMap<>();
    }

    public void addStudent(String name, int grade) {
        studentGrades.put(name, grade);
    }

    public void removeStudent(String name) {

```

```

        studentGrades.remove(name);
    }

    public int getGrade(String name) {
        return studentGrades.getOrDefault(name, -1);
    }

    public void displayMenu() {
        System.out.println("Menu:");
        System.out.println("1. Add a student");
        System.out.println("2. Remove a student");
        System.out.println("3. Display a student's grade");
        System.out.println("4. Exit");
    }

    public static void main(String[] args) {
        StudentGrades studentGrades = new StudentGrades();
        Scanner scanner = new Scanner(System.in);

        while (true) {
            studentGrades.displayMenu();
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();
            scanner.nextLine(); // Consume the newline character

            switch (choice) {
                case 1:
                    System.out.print("Enter student name: ");
                    String name = scanner.nextLine();
                    System.out.print("Enter student grade: ");
                    int grade = scanner.nextInt();
                    scanner.nextLine(); // Consume the newline character
                    studentGrades.addStudent(name, grade);
                    break;
                case 2:
                    System.out.print("Enter student name to remove: ");
                    String nameToRemove = scanner.nextLine();
                    studentGrades.removeStudent(nameToRemove);

```

```

        break;
    case 3:
        System.out.print("Enter student name to display grade: ");
        String nameToDisplay = scanner.nextLine();
        int gradeToDisplay = studentGrades.getGrade(nameToDisplay);
        if (gradeToDisplay != -1) {
            System.out.println("Grade for " + nameToDisplay + ": " + gradeToDisplay);
        } else {
            System.out.println("Student " + nameToDisplay + " not found.");
        }
        break;
    case 4:
        System.out.println("Exit");
        scanner.close();
        return;
    default:
        System.out.println("Invalid choice. Please try again.");
    }
}
}
}

```

Output:

Menu:

1. Add a student
2. Remove a student
3. Display a student's grade
4. Exit

Enter your choice: 1

Enter student name: vamsi

Enter student grade: 1

Menu:

1. Add a student
2. Remove a student
3. Display a student's grade
4. Exit

Enter your choice: 1

Enter student name: rajju

Enter student grade: 2

Menu:

1. Add a student
2. Remove a student
3. Display a student's grade
4. Exit

Enter your choice: 2

Enter student name to remove: raju

Menu:

1. Add a student
2. Remove a student
3. Display a student's grade
4. Exit

Enter your choice: 3

Enter student name to display grade: vamsi

Grade for vamsi: 1

Menu:

1. Add a student
2. Remove a student
3. Display a student's grade
4. Exit

Enter your choice: 4

Exit

Solution 5:

```
package task4;
```

```
import java.util.Stack;
```

```
import java.util.Scanner;
```

```
public class CollectionStack {
```

```
    // Stack to store integers
```

```
    private Stack<Integer> stack;
```

```
    // Constructor to initialize the stack
```

```
    public CollectionStack() {
```

```
        stack = new Stack<>();
```

```
    }
```

```
    // Method to push an element onto the stack
```

```

public void push(int value) {
    stack.push(value);
    System.out.println("Pushed " + value + " onto the stack.");
}

// Method to pop an element from the stack
public Integer pop() {
    if (!isEmpty()) {
        int value = stack.pop();
        System.out.println("Popped " + value + " from the stack.");
        return value;
    } else {
        System.out.println("Stack is empty. Cannot pop.");
        return null;
    }
}

// Method to check if the stack is empty
public boolean isEmpty() {
    return stack.isEmpty();
}

// Main method to demonstrate the stack functionalities
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    CollectionStack integerStack = new CollectionStack();

    while (true) {
        System.out.println("Stack Menu:");
        System.out.println("1. Push Element");
        System.out.println("2. Pop Element");
        System.out.println("3. Check if Stack is Empty");
        System.out.println("4. Exit");
        System.out.print("Choose an option: ");
        int choice = scanner.nextInt();

        switch (choice) {
            case 1: // Push Element

```



```

        System.out.print("Enter an integer to push: ");
        int value = scanner.nextInt();
        integerStack.push(value);
        break;
    case 2: // Pop Element
        integerStack.pop();
        break;
    case 3: // Check if Stack is Empty
        if (integerStack.isEmpty()) {
            System.out.println("The stack is empty.");
        } else {
            System.out.println("The stack is not empty.");
        }
        break;
    case 4: // Exit
        System.out.println("Exiting the program.");
        scanner.close();
        return;
    default:
        System.out.println("Invalid option. Please try again.");
    }
}
}
}
}

```

Output:

Stack Menu:

1. Push Element
2. Pop Element
3. Check if Stack is Empty
4. Exit

Choose an option: 1

Enter an integer to push: 10

Pushed 10 onto the stack.

Stack Menu:

1. Push Element
2. Pop Element
3. Check if Stack is Empty
4. Exit

Choose an option: 1

Enter an integer to push: 20

Pushed 20 onto the stack.

Stack Menu:

1. Push Element
2. Pop Element
3. Check if Stack is Empty
4. Exit

Choose an option: 2

Popped 20 from the stack.

Stack Menu:

1. Push Element
2. Pop Element
3. Check if Stack is Empty
4. Exit

Choose an option: 3

The stack is not empty.

Stack Menu:

1. Push Element
2. Pop Element
3. Check if Stack is Empty
4. Exit

Choose an option: 4

Exiting the program.