SE2007 Object Oriented Programming Project Assignment

Smart Earthquake Emergency Bag Assistant

1. Introduction

In this project, you will develop a console-based Java application called the "Smart Earthquake Emergency Bag Assistant." The goal of the project is to apply core Object-Oriented Programming (OOP) concepts such as inheritance, polymorphism, abstract classes, interfaces, exception handling, and file I/O.

The application helps users prepare an earthquake emergency bag by taking into account the number of family members, their categories (Adult, Elderly, Baby, Chronic Patient), the earthquake risk level of the region, and the maximum carrying capacity of the bag. Based on this information, the system automatically loads base items and personalized items from external text (.txt) files and generates a detailed summary of the emergency bag.

2. Learning Objectives

By completing this assignment, you are expected to:

- Design and implement an OOP-based system using abstract classes and inheritance.
- Use interfaces to define common behaviors (e.g., weight handling via Weighable).
- Apply polymorphism to handle different person categories uniformly.
- Work with file input/output (I/O) to read and append configuration data.
- Implement custom exceptions (OverweightBagException) for special error cases.
- Structure your code in a modular and maintainable way.

3. System Description

The Smart Earthquake Emergency Bag Assistant simulates a real-world decision support system that helps a family prepare for an earthquake. The workflow is as follows:

1. The user specifies how many family members will be included.
2. For each person, the user enters a name and selects a category:

-Adult
-Elderly
-Baby
-Chronic Patient

3. The user selects the earthquake risk level (HIGH, MEDIUM, or LOW).
4. The user enters the maximum carrying capacity of the emergency bag in kilograms.

5. For each person, the system loads:

- Base items depending on the risk level (from base_items_*.txt files)
- Personalized items depending on the category (from category-specific .txt files)

6. Each item is added to the EmergencyBag if it does not exceed the maximum capacity. If an item would exceed the limit, a custom OverweightBagException is thrown and handled.

7. A summary is displayed, including family members, item list, total weight, maximum capacity, and whether the bag is within the allowed limit.

4. Class Overview

The following classes and interfaces are required for the assignment:

Class / Interface	Responsibility
Person (abstract)	Base class for all person types; stores name and provides base item handling.
Adult	Represents an adult family member.
Baby	Represents a baby; has additional baby-specific items.
Elderly	Represents an elderly; may have special medical needs.
ChronicPatient	Represents a chronic patient; requires medicine-related items.
Item	Represents a bag item with a name and a weight (implements Weighable).
Weighable (interface)	Defines a common method getWeight() for all objects that have a weight.
EmergencyBag	Stores items, enforces maximum capacity, and reports the total weight.
OverweightBagException	Custom checked exception thrown when the bag exceeds its capacity.

ItemLoader	Loads items from external .txt files.
RiskLevel (enum)	Enumerates HIGH, MEDIUM, and LOW risk levels.
Main	Provides the user interface via the console and coordinates the system.

5. File-Based Configuration

All items are loaded from external text files. This allows the configuration to be changed without recompiling the program.

Required files:
- base_items_high.txt
- base_items_medium.txt
- base_items_low.txt
- adult_items.txt
- baby_items.txt
- elderly_items.txt
- chronic_items.txt

Each line in these files should follow the format:
ItemName;weight

Example:

```
Water 1.5L;1.5
Canned food;0.5
Flashlight;0.3
```

6. Exception Handling

To handle situations where the bag becomes too heavy, you must implement a custom checked exception named OverweightBagException. This exception should be thrown from the EmergencyBag.addItem(...) method whenever adding a new item would exceed the maximum capacity. The Main class should catch this exception, display a warning message, and continue processing the remaining items.

7. Assignment Tasks

1. Implement all classes and interfaces as described in the Class Overview section.
2. Implement file-based configuration using ItemLoader to read items from .txt files.
3. Implement the EmergencyBag class to enforce the maximum capacity and calculate total weight.
4. Implement the OverweightBagException and use in EmergencyBag.addItem(...) method.
5. Implement the Main class to:
   - Interact with the user via the console.
   - Create Person objects based on user input.
   - Load base and personalized items.
   - Display a detailed summary of the final emergency bag.
6. Implement the feature to add new items to category-specific files during runtime.

8. Sample Console Screens (Simulated)

The following examples are simulated console screenshots that illustrate how your program might interact with the user. Your exact output does not have to match character-by-character, but the overall structure and information should be similar.

Example 1: Creating a bag for one Adult and one Baby (HIGH risk):

```
=== SMART EARTHQUAKE EMERGENCY BAG ASSISTANT ===
1 - Create New Earthquake Bag
2 - Add New Item to Category
0 - Exit
Your choice: 1

How many family members are there? 2

1. person's name: Alice
Select category: 1-Adult 2-Elderly 3-Baby 4-Chronic patient 2

2. person's name: Bob
Select category: 1-Adult 2-Elderly 3-Baby 4-Chronic patient 3

Risk level: 1-HIGH 2-MEDIUM 3-LOW 1

Bag capacity (kg): 10
```

```
==== BAG SUMMARY ====
Family members:
- Alice (Elderly)
- Bob (Baby)

Items:

1. Water 1.5L                1,50 kg
2. Water 1.5L                1,50 kg
3. Water 1.5L                1,50 kg
4. Tinned food               0,50 kg
5. Tinned food               0,50 kg
6. Torch                     0,30 kg
7. Powerbank                 0,40 kg
8. First aid kit             0,80 kg
9. Thermal blanket           1,00 kg
10. Medicine kit             0,60 kg
11. Spare glasses            0,20 kg
12. Walking stick (foldable) 0,70 kg
13. Blood pressure monitor   0,50 kg


Total Weight: 10,00 kg

Maximum Capacity: 10,00 kg

Status: ✓ Within limit


Press Enter to return to the menu...
```

Example 2: Adding a new item to the Baby category:

```
=== SMART EARTHQUAKE EMERGENCY BAG ASSISTANT ===
1 - Create New Earthquake Bag
2 - Add New Item to Category
0 - Exit
Your choice: 2

Select category:
1- Adult
```

```
2- Baby
3- Elderly
4- Chronic patient
Your choice: 2

Item name: Extra milk powder
Weight (e.g., 0.5): 0.4

Item appended to file.
```

9. Grading Criteria

- Correct class structure and OOP design
- Inheritance, polymorphism, and interface usage
- File I/O implementation and correctness
- Exception handling (OverweightBagException)
- Code readability, comments, and style
- Program behavior and output correctness

10. Notes and Recommendations

- Make sure your code compiles and runs without errors before submission.
- Test your program with different risk levels and different combinations of family members.
- Do not hard-code item lists in Java code. All item definitions must come from external .txt files.
- Use meaningful class, method, and variable names.
- Comment your code where necessary to explain non-trivial logic.

11. Bonus Features (Optional)

You may improve your grade by adding:

- Support for multiple bags

- Exporting the final bag summary to a .txt file