CS5004, Object-Oriented Design and Analysis

Lab4: Inheritance and Composition. Interfaces.

Due date: Friday, 18th February 11:59pm

Therapon Skoteiniotis, Tamara Bonaci and Abi Evans

skotthe@ccs.neu.edu, tbonacin@northeastern.edu, ab.evans@northeastern.edu

1. Summary

In today's lab, we will continue our conversation about inheritance, composition, interfaces, and abstract classes. We will focus on:

- Inheritance and abstract classes
- Public behavior, contracts and interfaces
- Relationships between abstract classes and interfaces
- Exceptions: throwing and properly handling (catching) exceptions, as well as writing our own exceptions
- Testing methods that can throw exceptions

Note 1: Labs are intended to help you get started, and give you some practice while the course staff is present and able to provide assistance. You are not required to finish all the questions during the lab, but you are expected to push your lab work to a designated repo on the Khoury GitHub.

GitHub at the end of the lab.

2. Inheritance, Abstract Classes and Interfaces

Problem 1

You are a part of a team developing a smart **food ordering system** for a culinary institute. The smart food ordering system will eventually be expected to automatically generate a daily list of needed food items, and to send it to a grocery retailer, who will then fulfill the order, and deliver the ordered items next day.

The food ordering system orders **food items**, and `food items` can be one of:

- `Perishable food item`
- `Non-perishable food item`

`Perishable food` item can be one of:

- `Meat`
- `Fruit`
- `Vegetable`

`Non-perishable food` item can be one of:

- `Rice`
- `Pasta`

For every `food item`, the smart food ordering system keeps track of its:

- **Name**, represented as a String
- **Current price per unit**, represented as a Double
- **Currently available quantity**, represented as an Integer
- **Maximum allowed quantity**, which is a constant represented as an Integer. For all perishable food items, it equals 100, while for all non-perishable food items, it equals 250.

Additionally, for every `perishable item`, we also keep track of:

- **Order date**, encoded as a `LocalDateTime` object.
- **Expiration date**, also encoded as a `LocalDateTime` object.

Your tasks:

1. Write the `FoodOrderingSystem` class, as well as any other classes that help you to meet the above specification.
2. Write tests for your `FoodOrderingSysteem` class and all dependent classes.
3. Provide the UML Class diagram for your design of the smart food processing system.

3. Inheritance, Abstract Classes, Interfaces and Exceptions

Problem 2

You were tasked with building a prototype of a video game. The game designers have an idea, and they would like you to build a program to test their idea out.

The game consists of `Pieces`. A `Piece` can be:

- `Civilian`
- `Soldier`

A `Civilian` is one of:

- `Farmer`
- `Engineer`

A `Soldier` is one of:

- `Sniper`
- `Marine`

The designers provided the following properties:

1. All `Pieces` contain information about their:
    - `Name`, containing information about a `Piece's` first and last name
    - `Age`, which is an Integer in the range [0, 128], containing information a Piece's age
2. `Civilians` generate wealth. Each `Civilian` must keep track of their wealth, and wealth is a positive real number.
    - We should be able to increase a `Civilian's` wealth by passing a number to add to the current wealth of a `Civilian`.
    - We should be also able to decrease a `Civilian's` wealth by passing a number to remove from the current wealth of a `Civilian`.
3. `Soldiers` keep track of their stamina. Each `Soldier` must keep track of their stamina, and stamina is a real number in the range [0, 100].
    - We should be able to increase a `Soldier's` stamina by passing a number to add to the current stamina of a `Soldier`.
    - We should be able to decrease a `Soldier's` stamina by passing a number to remove from the current stamina of a `Soldier`.