Homework: Encapsulation and Polymorphism

This document defines the homework assignments from the "OOP" Course @ Software University. Please submit as homework a single **zip** / **rar** / **7z** archive holding the solutions (source code) of all below described problems. The solutions should be written in C#.

Problem 1. Shapes

- Define an interface **IShape** with two abstract methods: **CalculateArea()** and **CalculatePerimeter()**.
- Define an abstract class **BasicShape** implementing **IShape** and holding **width** and **height**. Leave the methods **CalculateArea()** and **CalculatePerimeter()** abstract.
- Define two new **BasicShape** subclasses: **Rhombus** and **Rectangle** that implement the abstract methods **CalculateArea()** and **CalculatePerimeter()**.
- Define a class **Circle** implementing **IShape** with a suitable constructor.
- Create an array of different shapes (**Circle**, **Rectangle**, **Rhombus**) and test the behavior of the **CalculateSurface()** and **CalculatePerimeter()** methods.

Problem 2. Bank of Kurtovo Konare

A bank holds different types of accounts for its customers: **deposit accounts**, **loan accounts** and **mortgage accounts**.

- Customers could be **individuals** or **companies**.
- All accounts have **customer**, **balance** and **interest rate** (monthly based).
- **Deposit** accounts are allowed to deposit and withdraw money. **Loan** and **mortgage** accounts can only deposit money.
- All accounts can calculate their interest for a given period (in months) using the formula
  A = money * (1 + interest rate * months)
- **Loan** accounts have no interest for the first **3 months** if held by **individuals** and for the first **2 months** if held by a **company**.
- **Deposit** accounts have no interest if their balance is positive and less than 1000.
- **Mortgage** accounts have ½ interest for the first 12 months for **companies** and no interest for the first 6 months for **individuals**.

Write a program to model the bank system with classes and interfaces. You should identify the classes, interfaces, base classes and abstract actions and implement the calculation of the interest functionality through overridden methods. Write a program to demonstrate that your classes work correctly.

Problem 3. Game Engine

You are a given a **partially-implemented game** (see homework archive). The game consists of turns and characters. Every turn each character performs 1 **interaction** with another **character** within his **range**. The game consists of the following classes:

- **`Engine`** – contains several methods for parsing the input (read from the console) and executing commands. Currently, the engine only supports the **`status`** command (prints information for each character in the game).
- **`GameObject`** – base class for objects in the game. Contains field **Id**.
  - **`Character`** – base class for all active character objects. Contains fields **X** and **Y** coordinates, **`Health points`**, **`Defense points`**, **`Team`** (Blue or Red), **`Inventory`** (list of items), **`Range`** (for interacting with other champions) and **`IsAlive`** (for tracking if the character is dead or alive). Holds

the methods **AddToInventory/RemoveFromInventory** (adds/removes an item to the character's inventory), **ApplyItemEffects/RemoveItemEffects** (applies/removes the item's effect on the character) and **GetTarget** (gets the most suitable target to interact with).

- **Item** – base class for all items in the game. Contains fields **HealthEffect**, **DefenseEffect**, **AttackEffect** and affects the fields of the character who uses the item.
- **IHeal**, **IAttack**, **ITimeout** interfaces

The following **Characters** should be implemeted:

- **Warrior** – implements **IAttack** and interacts by attacking alive characters from the other team. Always picks the first target. Has default **Health points** of **200**, **Defense points** of **100**, **Attack points** of **150** and **Range** of **2**.
- **Mage** – implements **IAttack** and interacts by attacking alive characters from the other team. Always picks the last target. Has default **Health points** of **150**, **Defense points** of **50**, **Attack points** of **300** and **Range** of **5**.
- **Healer** – implements **IHeal** and interacts by healing alive characters from his/her own team. Always picks the target with the least **Health points** (cannot target self). Has default **Health points** of **75**, **Defense points** of **50**, **Healing points** of **60** and **Range** of **6**.

All characters are created via the command **create *characterClass id x y team***

The following **Items** should be implemented:

- **Axe** – **Item** with **HealthEffect** of **0**, **DefenseEffect** of **0** and **AttackEffect** of **75.**
- **Shield** – **Item** with **HealthEffect** of **0**, **DefenseEffect** of **50** and **AttackEffect** of **0**.
- **Bonus** – base class for an item with a **temporary effect**. Implements **ITimeout**. Items of type **Bonus** are removed from the character's inventory after a few turns (depending on the **timeout** value). **Their effects on the player are also removed.**
  - **Injection** – **Bonus** item with **HealthEffect** of **200** for **3** turns. If a character's health points fall under 1 after the bonus times out (and is removed), his/her **health points** become **1**.
  - **Pill** – **Bonus** item with **AttackEffect** of **100** for **1** turn.

All items are added via the command **add *character itemClass itemId***

The engine currently supports all game logic (input parsing, interactions, attacking and healing calculations, etc.). Your task is to study the engine and **implement** the necessary classes and their **functionality** so that the game engine may use them. After all turns, the engine **prints** the winning team (the team with most characters alive) and **information** about the characters. An **empty input line** denotes the input's end.

You are **NOT** allowed to edit directly any of the given classes / interfaces. You may edit the **Main()** method only.

Input	Output
create mage Nakov 3 4 Red add Nakov axe Axe add Nakov pill IronPill add Nakov injection AnalInjection create warrior Vlado 5 4 Blue add Vlado shield HeavyShield create healer Alex 7 8 Red create warrior BateArni 2 3 Blue add BateArni axe TurboMegaAxe add BateArni shield TurtleShield	Red team wins the game! -- Name: Nakov, Team: Red, Health: 290, Defense: 50, Attack: 375 -- Name: Alex, Team: Red, Health: 75, Defense: 50, Healing: 60