Mandatory Assignment in Advanced Software Construction

Idea:

To define a Library with classes that together provide a mini framework for turn-based 2D game by using different tools and techniques obtained throughout the course.
However, **you should not support any GUI**.

In this library you are to focused on creating a world with different creatures (of your flavour), which could have different protection (shield, magic etc.) and different attack possibilities (weapon, magic, army etc.).

The reason for having a 2D game is, that it is much easier to maintain, design and implement. In addition, for the same reason you are to make a turn based game.

Due to the topic during this course 'Advanced Software Construction' you are not to implement nor consider any User Interface in the Framework.

For inspiration:
*To be inspired look at Greenfoot see main page or the API references they are working with a **world** and with **actors**, but also GUI (which in this assignment is out of scope).*

Deadlines and Delivery:

You must work individual and hand in individual - but it is allowed to talk and be inspired by each other.

Working period for the assignment is in the period Date 11th March to 22nd April

Friday the 22nd April

Demonstration of the framework to the teacher between 9:10 PM and three PM.
Hand in a URL to your GitHub repository in Wiseflow.

You should prepare and make a demo/presentation. A common plan for the day of demos is coming mid-April.

Detailed description:

(Changes and precessions may occur during working period)

You must create a **Library** with classes, which together form a mini framework for a turn based 2D game where you have a World, Creatures and Objects (e.g. obstacles).

You **must not** make any written report. Comments in the code are welcome but no text-document like a report.

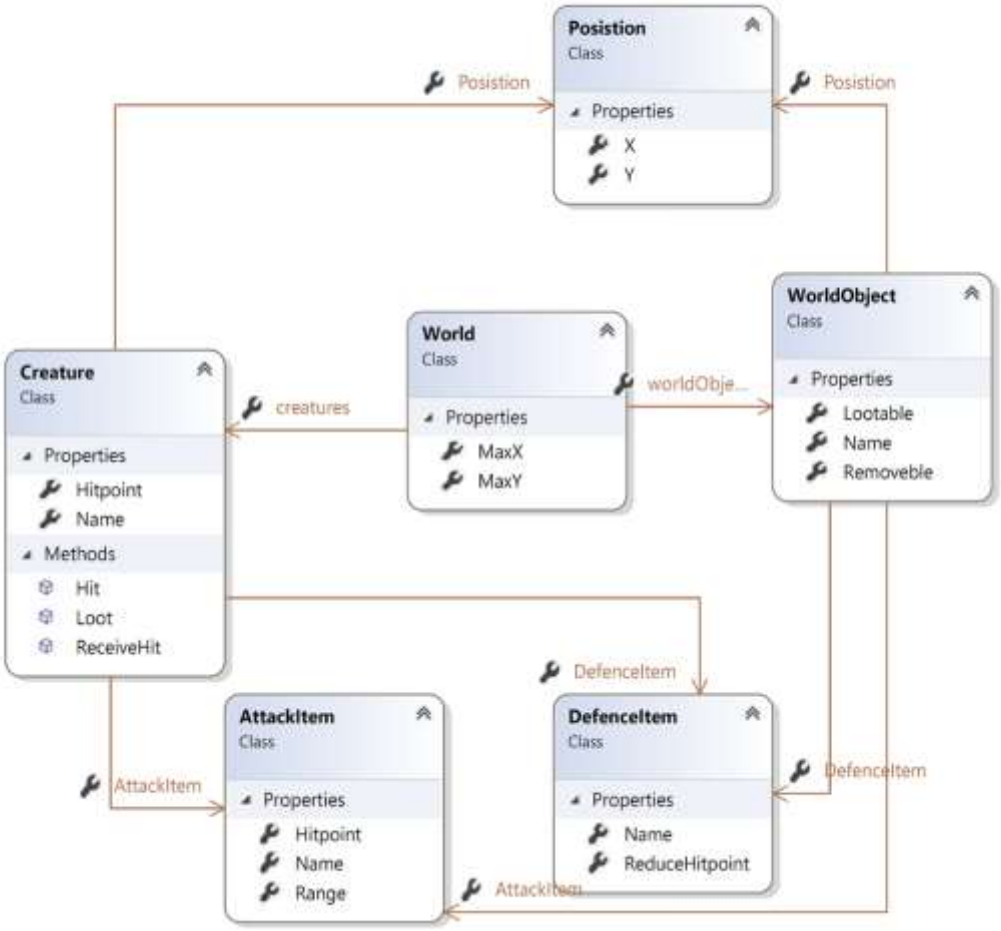
Therefore, you are to make **a Framework** i.e. a library to support other game-designers doing Turn-based 2D games.

Your solution must be accessible from a GitHub-repository (or similar repositories).

The first step is to implement following elements (see also next page):

- A World (some 2D playground)
- Creatures (which have a position in the world)
- Objects (with a fixed position)
  Some should be removable from the world and some could hold bonus or drawbacks (new weapon, new shield, hit points, some anti-protection or ...)
- Attack objects (weapon, magic, ...)
- Defence objects (Shield, boots, ...)

For a beginning, implement the class diagram below:



- **World** (ideas to properties size in x, y + list of creatures/objects)
- **Creature** (ideas to properties list of attack/defence, methods Hit, ReceiveHit, Pick)
- **GameObject** (placed in world, can be removable)
- **Attack** Object (Placed at Creature, a possibility also in bonus boxes; ideas to properties hit point, range, description)
- **Defence** Object (Placed at Creature, a possibility also in bonus boxes; ideas to properties reduce hit point, description)

Some functions

- A Creature can hit other creatures

- A Creature can pick objects (if the object can be picked), whereby the creature can get weapon, shield, magic, hit points, or like.

- A Creature can receive hit and possible die if life-point are zero or below

Improve the framework

You have done the initial work, where you have implemented the class diagram above i.e. have implemented the classes simple and concrete.

Now you should refactor the framework to be more and more flexible for modifications and make use of the different techniques you have faced and will face during this course.

- The framework must be **configurable** from a configuration-file (week 5)
- The framework must support **tracing/logging** messages (week 6)
- The framework must be **documented** i.e. ///-comments (week 5)
- The framework must follow the principles of **SOLID** (week 12)
- The framework make use of iterations and **LINQ** (week 14)
- At least three **Design Pattern** (week 5, 10, 13) e.g. among these:
  - Template
  - State
  - Composite
  - Observer
  - Decorator
  - Strategy
  - Factory (Abstract Factory)
- Some nice features could be
  - **Reflection** (week 11)
  - **Operator overload** (week 14)

Issues to consider:

- If the playground is a matrix of x and y coordinates or do all creatures and world-objects have an x and y coordinate.
- if hit is the sum of hitpoint in attack objects or only an individual object
- Alike for defence just the sum of reduceHitPoint.
- If any state can give general super power or special weakness. (Like Mario).
- If any AbstractClasses with template method.
- If all objects and creatures should have an id.
- If there is a limit of number of attack weapon or defence weapon, a creature can carry.
- If a creature can switch between different weapons, it carry.
- If there could be different levels/categories of creatures.

Try Your Framework

- Besides creating the framework, you must create a new project yourself where you try your own created framework by creating some example classes from your framework.

- Make a small game based on your framework i.e. make some concreate classes, that inherit from your framework interfaces and classes.
  Thereby, you can demonstrate the strength and easy use of your framework.

- **Do not** make a full game. The focus is on the framework.

For those who run fast - more investigation

- A game loop, where in each loop all creatures do a move and possibly a hit or pick a an object

- Creatures can move, the movement in **one** turn is always shortest way e.g. X=3 steps Y= -10 steps, it is **possible** to check if any objects / obstacles are in a way.