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 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.

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 14th March to 25th April

Friday the 25th April

Demonstration and presentation of the framework to the teacher between approx. 8 PM and 4 AM. Each student has around 10 min.
Besides you must hand in an URL to your GitHub (or gutbucket, …) 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 subset of 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.

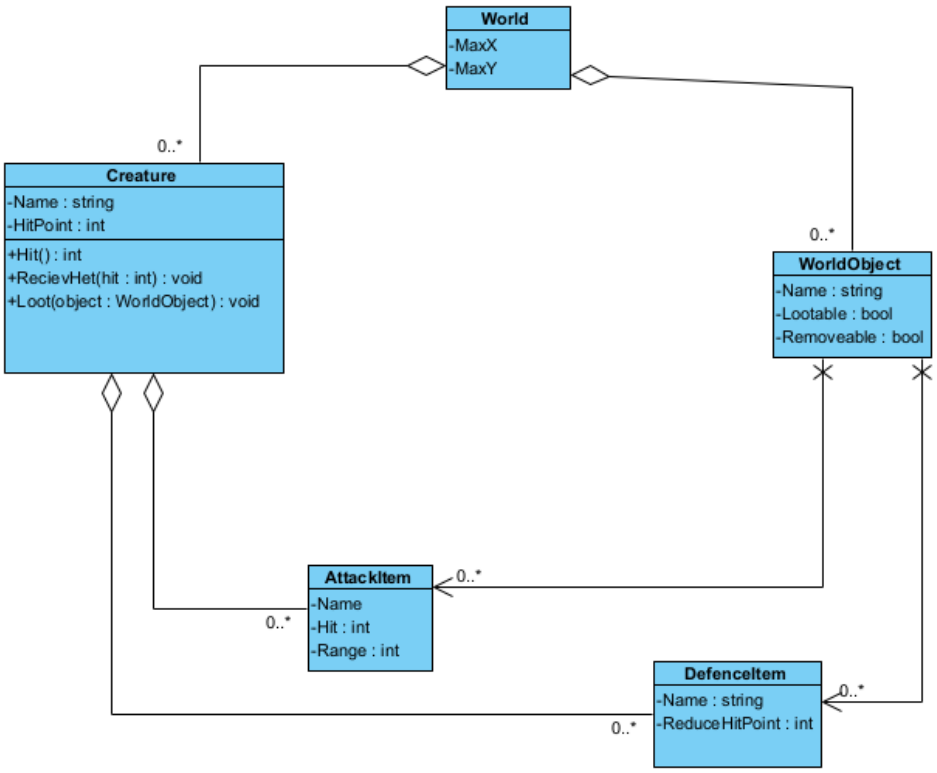
Therefore, you are to make **a Framework** i.e. a library to support other game-designers doing Turn-based 2D games.

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, name of the creature, hitpoints (or livepoints) and methods Hit, ReceiveHit, Pick/Loot)
- **WorldObject** (placed in world, can be removable)
- **Attack** Object (Placed at Creature, a possibility also in bonus boxes or just on the ground; ideas to properties hit point, range, description)
- **Defence** Object (Placed at Creature, on the ground or 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 or loot objects (if the object can be pickable or lootable), 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 step by step 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 7)
  Configurable parameters should at least be:
  world size i.e. max x and max y, **level of the game** (novice, normal, trained)
  but can include more parameters

- The framework must support **tracing/logging** messages (week 7)
  Throughout the framework appropriated messages must be logged (traced)
  Make sure the logging is flexible in use seen from the programmer of your framework. Meaning the user should have the possibility to adjust the logging of their own taste e.g. adding and removing TraceListeners (i.e. on file, format or ..).

- The framework must be **documented** i.e. ///-comments (week 6)
  Public methods in the framework must be documented using xml-comments (///-comments)

- The framework must follow the principles of **SOLID** (week 13)
  You must demonstrate at least one example of S,O,L,I and D.

- The framework makes use of iterations and **LINQ** (week 15)

- Apply **Design Pattern** (week 6, 11, 14) to your framework

  - The Creature must be a **Template**
  - The Creature must support **Observer** whenever have been hit
  - The AttackItem must have a **Decorator** for boosting the weapon hit
    (or perhaps weaken the attack force e.g. if hitpoints/livepoints are below 25%)
  - The AttackItems must be able to be contained in a **Composite** class
  - The Creature must include one function which is implemented as a **Strategy**
  - You can add other design patterns as well.

- Some nice features could be supported
  - **Reflection** (week 10)
  - **Operator overload** (week 15)
    e.g. creature = creature + worldObject (instead of /together with the Loot method)

Issues to consider:

- If any state can give general super power or special weakness. (Like Mario).
- 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 (inventory) if yes try to use flight weight design pattern.
- If a creature can switch between different weapons, it carries.
- If there could be different levels/categories of creatures.

Make a small console application to try your own Framework

- Besides creating the framework, you must create a new project (Console App) yourself where you try your own created framework by creating some example classes from your framework and instantiate objects of these.

- Make classes etc. 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 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.
  Path findings (week 12)