Object Oriented Programming

Project - description of tasks

The main goal of the project is the implementation of a 2D virtual world simulator. The virtual world should have the structure of a two-dimensional NxM grid. In this world, simple life forms will exist, each with different behavior depending on its species. Every organism should occupy exactly one cell of the world's 2D grid array. Each cell can contain no more than a single organism at a time (in case of a collision, one of the organisms should be removed from this cell).

All action in the simulator should be performed in turns. During each turn, every living organism in the world should perform an action appropriate to their kind. Some of them will move (animals), while others will remain static (plants). In case of a collision (one of the organisms enters a cell occupied by another organism) one of them wins, either by killing its opponent (i.e. wolf) or by reflecting the opponent's attack (i.e. turtle). The order of all actions during a single turn depends on the initiative of each living organism. The animals with the highest initiative move first. In case of animals with the same initiative, the order is determined by the animal's age (the oldest animal will move first). In case of a collision, the victory depends on the strength of both organisms - the stronger organism wins (with some exceptions, eg. turtle). In case of equal strength, the encounter is won by the attacker. The game should also include a human player, who is a specific kind of animal. Unlike regular animals, his movement is not random. Instead, the direction of the human's movement is determined by the player by pressing the appropriate arrow key before the start of every round. Human also possesses a special ability (see Appendix A. Human special abilities), which can be activated with a separate key. Once activated, the ability works for 5 turns, after which it is automatically deactivated. After deactivation, the ability cannot be activated for the next 5 turns. The simulation should be initiated with several instances of every kind of organism already placed in the game world. The program window should include a text box for displaying messages about the results of fights between animals, consumption of plants and other events occurring inside the simulated world.

Student's first name, last name and index number should be displayed in the application user interface.

Tips for project implementation:

Create a class named **World** which will manage the gameplay and life cycle of organisms. It should contain methods such as:

- makeTurn()
- drawWorld()

and fields such as:

- organisms

Create an abstract class named **Organism** which will be a base for Animals and Plants:

basic fields:

- strength,
- initiative,
- position (x,y),
- world - reference (pointer) to the cell of the world grid in which the organism is placed

basic methods:

- action() - basic behavior of organism should be implemented in this method,
- collision() - behavior of organism in case of collision with other organism,
- draw() - method which draws or returns symbolic or graphical representation of the organism.

Class **Organism** should be abstract, and should be the base class for two other abstract classes: **Plant** and **Animal.**

Common behaviors for all animals should be implemented in the class Animal. Behavior such as:

- base movement in method action() - every typical animal moves to a randomly selected neighboring field,
- multiplication in method collision() - when two animals of the same species collide, instead of fighting with each other, both animals remain in their original positions, and next to them a new animal of their species is created.

Class **Human** should extend the class **Animal**. Human does not implement random movement nor multiplication, but instead is controlled by the player. There should be only one instance of **Human** on the map.

Table 1 Human class properties

strength	initiative	action()	collision()
5	4	Human moves in same way as animals, but the direction of his movement corresponds to the arrow keys pressed by the player. For example, if the player presses the left arrow key, the human (in his turn) will move to the cell to the left of his initial position.	Human possesses a special ability (see Appendix A. Human special abilities), which can be activated by a separate key on the keyboard. After activation, this ability affects the behavior of his collision() method for next 5 rounds. After that, the special ability is turned off and cannot be activated for the next 5 rounds.

Implement 5-6 subclasses of **Animal.** The available types of animals are defined below:

Table 2 Description of animal classes

Id	animal	strength	initiative	action()	collision()
1	wolf	9	5	default for Animal	default for Animal
2	sheep	4	4	default for Animal	default for Animal
3	fox	3	7	Has good sense of smell: fox will never move to a cell occupied by a stronger organism.	default for Animal
4	turtle	2	1	Has 75% chance to stay in the same place.	Reflects attacks of animal with strength less than 5. Attacker will return to the previous cell.

Id	animal	strength	initiative	action()	collision()
5	antelope	4	4	Has wider range of movement - 2 fields instead of 1.	Has 50% chance to escape from fight. In such case it moves to a free neighboring cell.
6	cyber-sheep[1]	11	4	Its main goal is the extermination of Sosnowsky's hogweed. It always moves towards the closes hogweed and tries to eat it. If there are no Sosnowsky's hogweeds, it behaves like a normal sheep.	Eats Sosnowsky's hogweed.

In class **Plant** implement common behaviors for all plants:

- spreading of plant in method action() - with a certain probability the plant can "sow" a new plant on a random free neighboring field.

Initiative for all plants is 0.

Implement 5 classes of plants.

Table 3 Description of plant classes

Id	plant	strength	action()	collision()
1	grass	0	default for Plant	default for Plant
2	sow thistle	0	Performs 3 attempts at spreading in each turn	default for Plant
3	guarana	0	default for Plant	Strength of the animal which ate guarana is permanently increased by 3.
4	belladonna	99	default for Plant	Kills any animal which eats it.
5	Sosnowsky's hogweed	10	Kills every animal in its immediate neighborhood except cyber-sheep.	Kills any animal which eats it, apart from cyber-sheep.

Create the class **World** which will contain objects of class **Organism**. Implement turns by calling the action() method for every organism on the map and collision() for organisms on the same cell. Remember that **the order of calling the action() method depends on the initiative (and age) of the organisms**.

Organisms can affect the world state (eg. by spawning new organisms). Therefore it is necessary to pass to methods action() and collision() a reference to the instance of the **World** class. Remember: class **World** should define as public only such methods and fields that are required by other objects. The rest of its methods and fields should be private or protected.

[1] mandatory only in the third project - Python

Project 1. C++

Visualization of the virtual world should be performed in console. Every organism should be presented as a different ASCII symbol. Pressing one key should cause a transition to the next turn, clearing the console and printing all symbols in their new states. At least one line of text in console should be used to report results of events such as fights, spawning new animals/plants and eating.

Suggested marks:

- 3 points:
  - Implementation of the world and its visualization,
  - Implementation of all required animals without breeding,
  - Implementation of all plants without sawing,
  - Implementation of a Human which can controlled by arrow keys.
- 4 points, everything above, and:
  - breeding of animals and sawing of plants,
  - implementation of Human's special ability.
- 5 points:
  - Implementation of saving and loading state of the world to file and from file.

Project 2. Java

Create an application similar to that written in C++. This time UI should be created using Swing (or other GUI library for Java). Application functionality (transition to the next turn, saving and loading game state) should be realized by components from the GUI library.

Suggested marks:

- 3 points:
  - Implementation of the world and its visualization,
  - Implementation of all required animals,
  - Implementation of all plants,
  - Implementation of a Human which can controlled by arrow keys.
  - Implementation of Human's special ability,
  - Implementation of saving and loading state of the world to file and from file.
- 4 points:
  - Implementation of adding a new organism to the word by clicking on a free map cell. It should be possible to add organism of any kind.
- 5 points:
  - Implementation of abstraction for world, creating two different versions. One implementation should use grid and the other one should use hexagonal fields (with all consequences that impact neighborhood relations between cells).

Project 3. Python

Implement similar application to that written in Java using Python language and any GUI library.

Suggested marks:

- 3 points:
  - Implementation of the world and its visualization,
  - Implementation of all required animals (including cyber-sheep),
  - Implementation of all plants,
  - Implementation of a Human which can controlled by arrow keys.
  - Implementation of Human's special ability,
  - Implementation of saving and loading state of the world to file and from file.
- 4 points:
  - Implementation of adding a new organism to the word by clicking on free map cell. It should be possible to add organism of any kind.
- 5 points:
  - Implementation of abstraction for world, creating two different versions. One implementation should use grid and the other one should use hexagonal fields (with all consequences that impact neighborhood relations between cells).

Appendix A. Human special abilities

Choose one ability and implement it.

Table 4 Human special abilities

Id	Ability	Description
0	immortality	Human cannot be killed. In case of confrontation with stronger opponent he is moved to a random free neighboring cell.
1	magical potion	Human strength rises to 10 in first turn and decreases by "1" per round until it returns to original value.
2	antelope's speed	Human's movement range is 2 for 3 rounds. In next 2 rounds probability that he moves by 2 cells is 50%.
3	Alzur's shield	Human deters all animals. Animal which collides with Human is moved to a random free neighboring cell.
4	purification	Human destroys all animals and plants that are adjacent to his position.