COMP10062: Assignment 6

The Assignment

This assignment is about effectively using inheritance, association, overriding, and constructors. You're going to work alone to translate a description of the inhabitants of a game world into a set of classes that could serve as part of a **model** for a game or gaming utility, draw the UML class diagram for these classes, and then implement them in Java.

Read the descriptions below carefully and try to take every detail into account in your design. There might be multiple ways to implement each attribute and ability. Just because your idea is different, doesn't mean it's wrong.

The Inhabitants of Old Ireland

In a small town known as Ballybeg, north of Dublin, there once lived mystical creatures. In this town, there are three kinds of Inhabitants: Werewolves, Witches, and Humans.

Attributes

All Inhabitants have names that cannot change.

Inhabitants also have numeric ratings for three basic attributes that describe their abilities: Strength, Agility, and Health Rating. These attributes can be raised or lowered by one point at a time, within the range 1 to 10.

The name and basic attributes of an inhabitant are obvious and can be easily retrieved by anyone.

Werewolves cannot be created with attribute values less than 5 for all 3 attributes. Witches cannot be created with attribute values less than 5 for agility and health rating. If a value less than 5 is provided or randomly generated, a value of 5 is assigned in these cases.

An Inhabitant is alive if their heath rating is greater than 0, otherwise they are dead. An inhabitant cannot be brought back from the dead.

Every Werewolf belongs to a pack. They are proud of their allegiance and would never change it. There will be many werewolves in a single pack, but only one werewolf can be the Alpha (leader).

A witch may belong to a werewolf pack or may be a lone witch. A pack may not have more than one witch in the pack. Also, werewolf packs are not loyal to witches, and they may change which witch is a part of their pack.

A Human can be a Healer or a Hunter. A healer starts with a magic rating that is randomly set between 1 and 10.

Abilities

Attack and Defense

Unfortunately, not everything in Ballybeg is peaceful and the inhabitants often attack one another.

When an Inhabitant other than a Hunter attacks another Inhabitant, it produces a damage score equal to the average of its strength, agility, and health. When a Hunter attacks, it does twice the normal damage.

When an Inhabitant defends itself against an attack, it will lose health points equal to the damage score of the attack divided by a random defense value between 1 and 10. If the damage score divided by the defense value rating is less than 1, than it will lose 1 health point.

Dead inhabitants cannot attack or defend themselves.

Werewolves or witches of the same pack cannot attack one another, they are too loyal.

Healing

When a Healer uses their magic, they can produce a healing score to help heal another Inhabitant (increase their health rating). The healing score is equal to their magic rating divided by 2.5, but the healer loses 1 point from their magic rating. If its magic rating is 0 or less, it cannot heal. Dead Healers can't heal.

A witch can help a healer by strengthening them. When a witch strengthens a healer, the witch gives the healer 1 point for their magic rating for every 3 points the witch has in strength.

The Design

Draw a UML class diagram for a set of classes to represent the inhabitants of the word described above. Make sure to represent all instance variables, parameters, return values, and relationships between classes. You can create your UML diagram on paper, using UMLet or draw.io. If you want to use a different piece of software, check with your instructor first.

Use **inheritance** to eliminate code duplication and **overriding** to avoid if statements (e.g. don't have an if statement to check if the Inhabitant is a Hunter before reporting damage).

Use **encapsulation** to restrict access to the instance variables.

Include the minimum set of **methods** to implement the world as described above.

Include two **constructors** for each class: one that sets all instance variables, and one that sets just the name and fills in random parameters for everything else.

Include a `toString()` method for every class that prints the class name, the name of the humanoid, whether they are alive or dead, and all their ratings (health, strength, etc.). Make sure the output is easy to read.

The Implementation

Implement the classes you designed in Java.

It would be a good idea to test each class by creating at least two instances (one for each constructor), calling each method, and printing the results of each method to standard output. But this is not required. For this assignment, you are developing the **model** but not the view.

Use automatic code generation as much as you can (“Insert Code” to create constructors, getters, setters and `toString()` methods).

Use the `super` and `this` keywords to eliminate code duplication wherever possible.

Check for errors in all methods (e.g. parameters out of range, null parameters, actions can’t be taken when the Inhabitant is dead, etc.). If an error happens, print a meaningful error message to standard output and return immediately from the method. Make sure you test the error cases.

Optional Extras

This is purely a design exercise, but maybe it could be turned into some sort of a game. For example, can you simulate turn-based combat between a Werewolf and a Hunter? Can you make it graphical? Is there anything else you can do to make the town of Ballybeg come to life?