Final Project- Battle Royale

Object-Oriented Programming 2024/2025

Overview:

A Battle Royale video game is a multiplayer game in which several players in teams or individually aim to fight until they are the last player or team standing eliminating all other opponents. Generally, players start with minimal equipment, and acquire new elements that can help them in their objectives. The game ends when only one player or team is left standing, and the game usually provides some sort of reward depending on how long they survived.

Specific Case Description:

Create a Battle Royale type game that allows you to select a number of players, allows them to choose characters and customize them by adding a tool that helps them in the fight. The game must initialize everything and simulate a battle between the players. In the end, there should be a winner and a reward for the winner. For proper operation, it must be complied with:

- The program asks for the total number of human players and the total number of players on the machine, as well as the difficulty level.

- The interpretation of the difficulty level is free for the student (e.g. the tools of the players of the machine have bonuses in their characteristics, or receive less damage, or whatever...)

- The program will create the characters either automatically, or through a single file the characters and tools available to human players.

- If there are enough characters and tools, the characters will be chosen. If there are not enough resources, it will notify it and the execution will be ended.

- Once the assignment has been made, and following a system of turns, either rotating or random, each player will make an attack on another until only one remains.

- The player will be offered the possibility to upload the result of the battle to a file, the system will have the ability to save the last 1000 operations performed (Operations Log type), as well as the final result.

- The game can be played on a console or through a graphical interface.

Minimum evaluation criteria

1. (2 points) Create an inheritance hierarchy (characters and tools) that models the different characters that can be chosen and the tools that can be used to modify their characteristics, deciding which attributes and methods each class should have and distributing them appropriately in the hierarchy.

    a. There should be at least three different types of characters, and each one should have abilities to character's own characteristics.

    b. Same with tools.

    c. Use abstraction (either through interfaces or abstract classes) when appropriate.

2. (0.5 point) Polymorphism will be used when printing the final assignations of each of the players participating in the game, reusing the methods where appropriate.

3. (0.5 point) Different constructors are used indicating default values where appropriate.

4. (0.5 point) Make correct use of the encapsulation, both in methods and in class attributes.

5. (0.5 point) The *toString*, *compareTo* and *equals* methods are implemented in the classes that require them.

6. (1 point) Create a split class that allows you to simulate battles, showing everything on console. One user interface must be organized per console to make the initial assignations and enjoy the battle.

Expanded evaluation criteria

7. (1.5 point) Implement the reading and writing of files. Methods and class variables (static) will be used for reading and writing. In case there is an error in the files (writing or missing characters) throw an exception created specifically for it. Capture possible exceptions corresponding to the unavailability of resources, for example not finding a read file.

8. (1.5) Make a configuration that allows interaction with the application.

    a. It must ask the user for information at the beginning, allow them to choose a character from among those available, show them the course and end of the battle and allow them to upload the result to a file.

9. (1 point) Classes that model tools that can be modified will implement a Modifiable interface. This will include the *checkModification()* methods that will verify that the necessary characteristics are met for the weapon to be modified and the *modification()* method that modifies it. This point can be replaced by another interface with another well-defined purpose.

10. (1 point) *Extraordinary point*. A point will be awarded for the effort, originality and level of difficulty of the game developed. Examples of meritorious efforts on this point are:

    a. Complex hierarchy structure, with more than two levels, or inclusion of new classes (e.g. potions...).

    b. Implementation of different forms of games, e.g. possibility of team play, map generation...

    c. Advanced graphical interface with visuals.

    d. …

Conditions for the realization, delivery and defense of the project

- The project must be developed **in groups of between 2 and 4 people** up to a **maximum of 8 groups per class**.
- The **eclipse project** must be delivered in a .zip exported from the IDE. The name of the .zip must follow the Nombre_Apellido_Proyecto_Final.zip format.
- **Each student** will upload the project to Blackboard.
- An explanatory report (5-10 pages max.) must be included in .pdf format. The name of each student who has been part of the project must be included. **THE MEMORY WILL BE INDIVIDUAL!,** the memories do not have to be the same, each member of the team can dedicate more detail to the part that they have had to develop by mentioning it in their individual report.
- Everything is delivered in Blackboard in the delivery entitled: **ProyectoFinal_Ordinaria.**
- The deadline for submission is Sunday, **December 15, 2024 at 11:59 p.m.**
- The following week will be reserved for the **defences of the projects** with an **approximate duration of 20 minutes** for each group to be carried out during two sessions.