CS 2340 — Milestone 2: Project Iteration 1

New Game Configuration, Use Case Diagram, and Domain Model

BACKGROUND: Over the next three milestones, you will design and build a Dungeon Crawler adventure game. You'll be responsible for implementing some of the functionalities to play the game. The player will be able to explore a dungeon consisting of several rooms containing monsters. The player will be able to fight the monsters. The ultimate goal of the game will be for the player to reach the last room of the dungeon, defeat the monsters there, and escape the dungeon. Future milestones will build upon your work and detail which new features to add.

PURPOSE: There are two primary goals for this project. The first goal is to provide you with experience collaborating with a team to develop a product with specific requirements. The second goal is to increase your aptitude with several key software engineering principles and technologies, namely SOLID, GRASP, Design Patterns, TDD, and Git. Over the course of the project, these concepts will be introduced to you so that they can be incorporated into your future milestones. With each milestone, testing will occur alongside development, and you will be responsible for writing unit tests to verify your implementation's functionality.

TASK: For this milestone, you are asked to create and submit to GitHub two design deliverables in addition to the first portion of your app and its accompanying tests. For the implementation portion of this milestone, you will create three screens: a welcome screen, a player configuration screen, and an initial game screen. Other than the requirements outlined below, the details of your implementation are up to you. Your app implementation and functionality will be graded during a demo which will occur the week after milestones are due.

Design Deliverable 1: Use Case Diagram

1. Categorize the following actors as *Primary*, *Supporting*, or *Off-Stage*
    - Player
    - Game Admin that uses the game's Admin screen to manage Players
    - Third-party database service that the game uses to store its state
    - Shareholder of the software company that develops the game
    - Contractor company that localizes the game to other languages
    - International Game Developers Association
2. Brainstorm one additional actor and categorize it like above
3. Draw a Use Case Diagram for the game application
    - Include the player and at least three other actors
    - Place Primary Actors on the left of the system boundary
    - Place Supporting Actors on the right of the system boundary
    - Place Off-Stage Actors near but unconnected to the system boundary
4. Include six or more *Functional Requirements* within the system boundary
    - *Functional Requirements* should define the ways in which *Primary* and *Supporting* Actors may interact
    - Example: player attacks monster, admins add additional shop items

Design Deliverable 2: Domain Model

1. Identify and list at least ten potential nouns which could be used in your project
    - Example: Player, Monster, Room
2. Sort the ten nouns into two categories (a minimum of 5 nouns must be classes)
    - Game Objects (classes): require their own methods and attributes
        - Example: Player, Monster
    - Attributes: do not require a whole class
        - Example: Price, Difficulty
3. Draw a Domain Model for the nouns you brainstormed
4. Connect each class within your Domain Model to at least one other class using associations
    - Example: Player "attacks" Monster
5. Include multiplicities for each association - one on each side of the association

Project Setup Requirements

1. The application should be implemented as a JavaFX desktop application.
2. Establish your project with version control using the Gatech GitHub.
   - Create a new repository for this application. All the remaining iterative milestones will continue to use this repository. Do not use the repository you set up during M1.
   - All group members must be contributors on the GitHub repository.
   - Include a .gitignore file specifying which files shouldn't be tracked. If you are unsure what to put in this file, look at the Java gitignore from GitHub. (Here is a helpful link: https://www.toptal.com/developers/gitignore)

Implementation Requirements

1. Display a **welcome screen** for the application
   - Must include a way to start the game (i.e. a start button)
2. Starting the game should take the player to an **initial configuration screen** which has the following requirements:
   - Allows the player to enter their name.
     - Players should not be allowed to pass in a null, empty, or whitespace-only name.
   - Allows the player to select a game difficulty from at least three options.
   - Allows the player to select a starting weapon from at least three options.
   - Allows the player to proceed to the next screen.
3. Implement the **initial game screen**, which should display textually or graphically the first room the player will see. You will be adding functionality in later milestones. For now, it must do the following:
   - Display the player
   - Display starting money
     - Starting money should vary based on the difficulty chosen.
   - Display exits
     - Up to four exits

Testing Requirements

1. Write **unit tests** to verify the functionality of your implementation.
    - There is no code coverage requirement, but you should make sure that your unit tests cover meaningful functionality.
    - *You should have at least 2 unit tests per team member.*
2. **Testing Deliverable**: Include with your submission a brief writeup describing your tests and testing process for the milestone. Explain which implementation components were chosen for testing and why. Additionally, explain how your tests verify that the code functions as expected.

Checkstyle

During your demo, your team will be required to run the **checkstyle.py** script (located on Canvas under **Files>checkstyle>2340 Checkstyle Guide.pdf**). This script will give your project a score out of 10 and will account for 10 points of your final M2 grade. Be sure to run the checkstyle script prior to submission to avoid unforeseen deductions.

Milestone Tagging

Tags are a way of marking a specific commit and are typically used to mark new versions of software. To do this, use "**git tag**" to list tags and "**git tag –a tag_name –m description**" to create a tag on the current commit. ***Important:*** To push tags, use "**git push origin –tags".** Pushing changes normally will not push tags to GitHub. *You will be required to checkout this tagged commit during demo.* This is done with **"git fetch --all --tags"** and **"git checkout tag_name".**

Submission Requirements

Be sure to include a link to your Gatech GitHub repository in your submission. This can be done via a submission comment on Canvas. Also, ensure that you have added your grading TA(s) and the professor as collaborators so that they may view your private repository. **Repositories must be located on the Georgia Tech GitHub and must be set to private!** Points may be deducted if these guidelines are not followed!

Criteria

You will be graded according to the successful and correct completion of the design deliverables, implementation requirements, and testing requirements above. Groups are required to demo in order to receive credit for the features they have implemented.