Homework 2-3

Zork Art Command Line

Zork is one of the earliest interactive fiction computer games, with roots drawn from the original genre game, Colossal Cave Adventure. You can find more information about Zork on the Internet. In this assignment, you will be implement a much simpler version of the game.

Game specifications:

1. Game is command line based Java application.
2. This is a single player game.
3. Once the game is started, player will be interacting with the game through command line.
4. Game must support at least all of the commands presented in lecture 2.
5. Game must support at least 3 levels and each level must have an objective in order to win the level.
6. Game must support concepts of items and monsters.
7. Read other game details in the lecture note 2.

Submission tasks:

1. Create a git repository for your project which you will be submitting. The repository should contain gitignore file that excludes all unrelated files and folders from committing. Create a file called `readme-git.txt`, and explain your reasons for excluding each of the rules in gitignore. 5 marks will be given for this task:
    a. 1 mark if you create a git repository as instructed.
    b. 1 mark if it contains at least 3 rules.
    c. 1 mark for correct explanations in the `readme-git.txt`.
    d. 1 mark if the submitted repository does not contain the files/folders as specified in the gitignore.
    e. 1 mark for pushing it to an online repository
2. Create a maven project in this git repository. You should configure the pom.xml to use only the dependencies you need. After compilation with maven, it should produce a runnable jar file (fat jar) in the target folder. Add `readme.txt` file in the root of the project to describe the compilation method and how to execute the game. 5 marks will be given for this task:
    a. 1 mark if project is mavenized.
    b. 2 mark if project is successfully compiled.
    c. 1 mark if jar file is runnable.
    d. 1 mark if it contains `readme.txt` with correct descriptions as instructed.
3. Write down your game story and objectives in the `readme-story.txt` file in the root. It must also contain the story and objectives for each of the levels you have. 5 marks will be given for this task so be creative.
4. Draw flow diagram of your game which represents how class/object/concept interact to each other. You must clearly represent the flow of the game from start to end, inheritance of classes, interaction of entities, etc. You are not expected to correctly use standard symbol in UML for this task. This task is to help you understand the flow of your own application so use any symbol that make sense to you. You will know if it makes sense or not if you present it to your friends, and they can understand it. This is an important practice for you before working on a bigger project. The diagram will be submitted as `diagram.png` in project root. You will be given 10 marks for this task. Use software tool to draw the diagram, e.g.

Google diagram, Power Point. You may draw the diagram by hand if it looks decent. Otherwise, I reserve the full right to deduct marks as I see fit. In this project, I will be your customer and will judge it as if I am a customer, not as your instructor.

5. Explain your diagram and give your reason for the design you make in `readme-design.txt`, e.g. why do you use inheritance here and there, or why do you use this particular design pattern. You will be given 10 marks for this task.
6. Write down all instruction manual for all commands, items, characters/monsters, play's properties, e.g. HP, mana (if you have it), etc in `readme-instruction.txt`. Explain rules you define for all interaction with the game, e.g. rules for attacking and retaliating, rules for decreasing items, HP, etc. You will be given 10 marks for this task.
7. Implement the game according to your design and specifications. You will be given 50 marks for this task.
8. Write unit testing for the components in your source code. You will be given 20 marks for this task.

Check list:

1. **`readme-git.txt`**
2. **`readme.txt`**
3. **`readme-story.txt`**
4. **`diagram.png`**
5. **`readme-design.txt`**
6. **`readme-instruction.txt`**
7. source codes
8. unit testing codes
9. You will be submitting a text file, named **`hw2-{student-id}-{name}.txt`**, which contains link to your git repository on canvas.
10. The task from 1-5 will be submitted as homework 2 on Canvas by Friday 23:59 ICT. You are not expected to be able to fully done with these tasks. It only serves as a check point and will be graded when you submit the homework 3.
11. The task from 1-8 will be submitted as homework 3 by the deadline given on Canvas. You are allowed to change everything that was previously submitted as homework 2. There is nothing to submit on Canvas because I will automatically pull your source codes from your repository at the specified date/time so make sure it is publicly accessible.