Overview

The final project challenges you to incorporate as many Java topics covered throughout the school year (CSCI 21 and CSCI 22). You may work ***individually*** or you may work ***with a partner***. The task is to design and implement a multiplayer network game.

Except for ***Bricks***, ***Pong***, ***Tic Tac Toe***, ***Space Invaders***, and ***Memory Cards***, the game may be based on existing games such as board games, card games, or other graphic multiplayer games, but they **cannot** be copies of the existing game. They **must** somehow be **modified**. You are **not** allowed to find a game development tutorial and submit the output from following that tutorial. It can be an original game as well.

You are allowed to search the Internet for resources regarding this project, and you **MUST CITE ALL** of your sources. ***Failure to cite any source will be interpreted as academic dishonesty and will be reported to the LS Discipline Committee for appropriate action.***

Required Classes

1. **GameServer** — This class contains the code that manages the game server's functionality. It also contains the **`main`** method that instantiates and starts the server.
2. **GameFrame** — This class contains the code that sets up the main **`JFrame`** for the player.
3. **GameCanvas** — This class extends **`JComponent`** and overrides the **`paintComponent`** method in order to create the custom drawing.
4. **Player** — This class contains the code that manages the player's appearance and functionality.
5. **GameStarter** — This class contains the **`main`** method that will start the game from the player's side.

You may decide what other classes to have and what to name them,
but make sure that the class names are descriptive of what they are for/what they do.

Project Requirements

The program **must** contain, but is not limited to, the following:

- graphical user interface
- inheritance (you must define a class that will be extended by at least two of the other classes that you also defined)
- inner classes
- animation using Threads or Timers
- networking to send and receive game data

Networking Requirements

The game **must** accommodate at least two (2) human players who will play against, or in cooperation with, each other. The game **must** run on two (2) separate machines: one server-player and one client-player. Alternatively, the game may run on three (3) or more separate machines: one server, and ***n***-1 client players (where ***n*** is the number of machines).

You may add a chat system, but it will not be considered as a fulfillment of the networking requirement. Examples of what are considered to fulfill the network requirement are: sending and receiving player coordinates, generating enemies/obstacles via information sent by the server, etc.

You may receive a higher grade for more complex networking functionality. Generally, a game where each player's movements are simultaneously mirrored on the other player's screen is more complex to implement than a turn-based game.

Documentation

1. Create a game manual in the form of a PDF document named Manual-GameName.pdf. (replace GameName with the name of your game). The manual should include:
    a. the title of the game
    b. the name(s) of the author(s)
    c. a description of the game
    d. an explanation of the mechanics of the game
    e. detailed instructions on how to play the game
2. Each of your Java files should contain **header information** enclosed in ***javadoc-formatted comments***, containing a description of the class, your full name(s), your ID number(s), and the date you created your program. This should be followed by a code block containing the appropriate **certification of authorship**. Each **class description** must be written ***in your own words***, with a **minimum of two sentences**. Your inner classes must have class descriptions as well. Additionally, your **methods**, including the constructors, must also have ***javadoc method comments***.

Submission Instructions

1. Create a folder named **Final-LastName1IDNumber1-LastName2IDNumber2** (alphabetically by last name). (For example, Final-Medalla999999-Sugay777777)
   If you are working individually, **Final-LastName-IDNumber**. (For example, Final-Medalla-999999)
2. Properly accomplish a Certificate of Authorship document (PDF Form, Group COA for those working in pairs or Individual COA for those working individually) with the following details:
    - Title of Submission: CSCI 22 Final Project
    - Type of Submission: Program
    - Cite your sources in the source citation text input area. This area automatically adjusts to accommodate the length of the text provided.
    - Fill in the Student and Course Information section.
    - File name: **Final-COA-IDNumber1-IDNumber2** or **Final-COA-IDNumber**.
3. Place the following in the submission folder:
    - Game Manual (PDF)
    - Certificate of Authorship (PDF)
    - source files (.java)
    - media resources (.jpg, .gif, .wav, etc…)

   DO NOT include ***.class*** files. If you are using an IDE, **DO NOT** include the IDE files and folders as well. Your program must work properly without any IDE packages.
4. Archive your project folder, which should produce a file called **Final-LastName-IDNumber.zip** or **Final-LastName1IDNumber1-LastName2IDNumber2.zip**
5. Submit your project folder through to the Moodle module by **Tuesday, 01 June 2021, 23:59**.
6. Upload your server application onto an AWS EC2 Instance. (See section on Server Testing)

Local Testing

Your application will be compiled via the command line: **`javac *.java`**
Your server application will be executed via the command line: **`java GameServer`**
Your player/client application will be executed via the command line: **`java GameStarter`**

Server Testing

Upload your server application onto an AWS EC2 Instance. Refer to the video guide on Moodle.
Your player/client application should be able to connect to your server and your game should run as expected.

Grading and Defenses

Provided that there are no errors, projects will receive a maximum grade of **B** (85) for fulfilling the requirements stated above. For the possibility of earning high marks, create a project that incorporates more of the things you have learned in a manner that is creative, appropriate, and challenging.

Quality of code shall also be taken into consideration. This includes, but is not limited to:

- formatting for clarity and readability
- using appropriate identifiers (names for fields, methods, and classes)
- maintaining consistency with naming scheme
- eliminating redundancy

Additional points may be awarded for creativity. This includes visual creativity (art and design in game graphics) and the game's story and theme.

Project defenses start on **Monday, 31 May 2021**, which is the day before the deadline. Sign-ups will be announced on Moodle as the deadline approaches.

Students who submit and defend before the deadline will receive bonus points. There will be limited slots for this.

Grades for projects that are submitted beyond the deadline and defended will be **capped at 75**.

Grades for projects that are submitted beyond the last day of defenses, **Friday, 4 June 2021**, will be **capped at 65**, with defenses being forfeited.

A student who fails to appear within the ***first five minutes*** of their selected time slot forfeits their project defense. There is no make-up defense, except in extenuating circumstances.

Important Notes

- Carefully and attentively read all specifications.
- Follow file naming conventions.
- Follow submission procedures.
- Do not submit any excess files.