AP Computer Science Final Project - README Template

Instructions:
The first step in creating an excellent APCS final project is to write up a README. At this stage, this README file acts as your **project proposal**. Once you've filled in all components, Shelby will read through it and suggest edits. Ultimately, you need a document that adequately describes your project idea and **we must agree on this plan**.

Have one member of your group **make a copy of this Google Doc**. Then, they should share it with all other members **and with Mr. Shelby** so that every group member has edit permissions, and Shelby can add comments on your ideas.

There's a lot of parts of this document that you might not have full answers for yet. Because you haven't written the program yet, it's difficult to think about the **instructions** or **which group members will do which parts**. Even though this is hard to think about, you must have something in these sections that acts as your current plan. However, during the course of the project, you'll **continuously update this document**. This means that you will not be *held* to exactly what you put here - components of this document can change (and it's pretty common!).

There is one exception: the **Features List** section. Once Shelby OKs your README, the Features List section **cannot be modified**. For this reason, it is most important that you get a solid idea of what you want to make and the primary features it will have *now*.

Talk with your group. Consider drawing some pictures of what you think your project might look like. Be precise. When you're ready, fill this out together. Each component in brackets below ([these things]) should be replaced with your ideas. Note that there are several sample READMEs posted on this assignment for you to use as guidance.

----------When README is finalized, remove everything above this line----------

Grinch Fringe

Authors: Dhruv Lohani, Ishwar Suriyaprakash, Arindam Kulkarni
Revision: 5/20/22

Introduction: (Ishwar)

To entertain yourself, imagine that you are an ordinary police officer in a world ravaged by hunger and disease. You are tasked with finding certain blueprints for a machine that will help accelerate food growth and production. To achieve this goal, you are sent to a maze, within which the blueprints are stored in rooms, and you have to navigate the maze to retrieve those blueprints. All blueprints must be collected. However, the maze is occupied by an elusive Grinch, who is very antagonistic to intruders and is also very possessive of its blueprints. The Grinch has rigged the maze with invisible traps that injure intruders, and the Grinch also has the power to kill intruders in close proximity, taking away from the Officer's health. While retrieving the blueprints, you have to evade the traps and the Grinch to exit the maze alive. For example, the Grinch happens to emit ionizing radiation, so a Geiger counter will help you determine your proximity to the Grinch.

Extra
Additionally, the Grinch may have evil collaborators in the maze who help it plant more traps and attack intruders, and, to communicate, they send encrypted messages between each other. You can intercept and decrypt those messages to find out more about their plans.

Instructions: (Arindam)

At the start of the game, we have a menu, which, for now, asks the user to click the center of the screen to start the game. Later on, the menu will have more and more options (such as options for different mazes, multiplayer, etc.). In gameplay, the user can press W, A, S, and/or D to translate the police officer and help it move. W key would make it go up, A would make it go left, S down, and D right. The Mouse changes the line of sight - it would change the direction in which the officer's flashlight points towards, and would allow you to look in different directions. To help the user pick up blueprints in the maze, the screen will tell the user that a blueprint is nearby. The user should press E to pick up blueprints.

Features List (THE ONLY SECTION THAT CANNOT CHANGE LATER): (Dhruv)

Features that exist

- The program displays a 2D maze with 2D graphics
- The maze has traps rigged around the maze that are not immediately visible to the officer
- The program displays blueprints in rooms around the maze that need to be collected by the officer
- The program has multiple options of maze layouts for the user to pick from

- The program only shows the parts of the maze that are illuminated by, for example, the officer's flashlight
- The maze has a *moving* Grinch that tries to attack the officer and kills the officer if in close proximity

Features that are nearly done

- To give the police officer an advantage, the officer can have access to teleportation technology but can only teleport between specific parts of the maze.
- The program has sound effects for events (eg. encountering a trap, Geiger counter beeping, reward sound for when a blueprint is found)

Must-have Features:

- The program displays a 2D maze with 2D graphics
- The maze has traps rigged around the maze that are not immediately visible to the officer
- The program displays blueprints in rooms around the maze that need to be collected by the officer
- The program has multiple options of maze layouts for the user to pick from
- The program only shows the parts of the maze that are illuminated by, for example, the officer's flashlight
- The maze has a *moving* Grinch that tries to attack the officer and kills the officer if in close proximity
- The game gives path recommendations for beginner players - a shortest Hamiltonian path (a path that visits all destinations - blueprint rooms) can be drawn out for the player before he/she starts the game.

Want-to-have Features:

- The maze has other antagonists that help the Grinch attack the intruders and plant more traps
- The program has sound effects for events (eg. encountering a trap, Geiger counter beeping, reward sound for when a blueprint is found)
- The game could include the ability for the officer to crack encrypted messages between Grinch and antagonists to find more information about their plans
- To give the police officer an advantage, the officer can have access to teleportation technology but can only teleport between specific parts of the maze.
- The officer can have a UV flashlight to identify the Grinch's footsteps (that is, if the Grinch is moving); these footsteps can help the officer locate the Grinch, which gives the officer an advantage
- Multiple players can play with each other (using networking); players can either play against each other in a free-for-all or play in a team to help each other

Stretch Features:

- The program can have 3D graphics that show the perspective of the officer
- The program can randomly generate mazes so that the user doesn't play in the same maze multiple times

- The program can generate a ‘maze’ from actual streets of a city such as Amsterdam or Paris. Sort of like a ‘chase’ scene which meanders through canals and small alleys of the street. It would really make it a fancy project.

Class List: (Dhruv)
[This section lists the Java classes that make up the program and very briefly describes what each represents. It’s totally fine to put this section in list format and not to use full sentences.]

Main
DrawingSurface
abstract ScreenObject - superclass of all objects displayed on the screen
Actor/Character - main characters/players in the game
- Officer *extends* Actor
- Grinch *extends* Actor
- (want-to-have) Minion *extends* Grinch

Screen - for displaying different things at different times
- GameScreen
- OptionScreen
- EndScreen

abstract Item - for traps and or resources that the player can use
- Trap *extends Item*
- Blueprint *extends Item*
- GeigerCounter *extends Item*

MazeData - stores data on the structure of the maze
HauntedMaze - has characters, items, maze walls
would be implemented in our game
gridP - represents grid points in the maze

Not there / functional:
- Networking class (want-to-have class) - would allow for multiplayer features
- Encryption Class (want-to-have class) - would facilitate how the encryption codes works, and how it

Credits: (all)

Ishwar:

I contributed ideas in our brainstorm for the capstone project, and we eventually decided to implement one of them in combination with Dhruv’s idea (the maze game). I also gave the UML diagram its initial structure, which included almost all of the classes, relationships, and other. I wrote the introduction for the game, which has the context of the game, the main goal that the user should achieve, and the challenges that the user has to overcome in the game. In addition, I helped organize the initial code structure for all the classes, with methods and fields.

I completed the GameScreen, HauntedMaze, and Officer classes, and I have also contributed to other classes like Screen and Actor. I implemented the whole HauntedMaze method, which contains an Officer, a Grinch, and a group of items (either Blueprints or Traps). In HauntedMaze, I implemented the draw() method and also the lighting feature, which only reveals parts of the maze that are illuminated by the officer's flashlight. I made the officer's flashlight point towards the direction of the mouse's location on the screen. In the GameScreen and Officer classes, I added support for user interaction with the WASD keys and programmed the Officer's movements in response to these inputs.

In GameScreen, I added a time-limit feature which restricts the time in which the user can complete the game's goals. In the Grinch class, I coded the Grinch's ability to move in the direction of the Officer. Also, I made image handling in the project more organized, and I helped add functionality for collisions between actors in the maze and maze walls.

At each checkpoint of the project, I built the javadocs and the jar for our game.

Arindam:

For the first week, I worked on creating our final idea with the group, developing the idea of an antagonist in the grinch, and officer. I helped develop the idea of a flashlight as well. I worked on creating and designing our final idea based on the ideas that we as a group collectively came up with. I completed a significant amount of the readme, such as the full instructions section, classes section, and stretch features. Finally, I worked on the UML diagram with Ishwar and Dhruv.

For Week 2, I completed the options screen class from start to finish, ensuring that a clickable button was made that took the user from the options screen to the game screen. I coded everything that involved switching screens and such. This included drawing methods such as MousePressed. I also completed the info bar class. This involved setting up its constructors, fields, and then also drawing it on the grid and printing appropriate info such as the health, status, and blueprints collected of the officer. Finally, I coded the item and blueprint classes, making them pop up on the screen waiting to be collected.

For Week 3, I set up blueprint pins that showed up in the maze that the officer had to pick up. This involved setting it up such that when the officer came close, it would be able to click E to pick it up. The blueprint would then show up in the Infobar, with an icon and identifier that represented it. In addition, I created hearts! These hearts represent the health of the officer, and the infobar would show the health of the officer as a number which was printed, and perhaps more aesthetically presented as hearts (full/half/empty). I then also coded resize methods in DrawingSurface that allowed the whole program to resize. I added an end screen, which differed based on a winning/losing result. If the user won the game, it showed a winning screen, and if the user lost, it showed a losing screen. Both screens included a replay button that took it back to the option screen. While initially, the option screen would once again lead to the end screen, I tweaked with game screen and option screen classes, and the timer methods, such that timer and the whole game screen was always reset, 'restarting' the game. Finally, I vastly improved the option screen and end screen aesthetics, adding images instead of rectangles for buttons, and background images, as well as a custom made logo for our game!

Dhruv:

I worked on the original idea stage, and came up with a few final ideas. We incorporated part of my idea, and Ishwar's idea, into our final game. I worked a bit on the UML Diagram, mostly completing what was not done such as class relations and some class variables. I did the features list(the must-have, want to have, and stretch features) and the class list on the Readme. I also helped organize the initial code structure for all the classes yesterday on Zoom with Ishwar and Arindam.

I completed the entire MazeData class. This included working on creating the logic to random perfect mazes, as well as reading in mazes from files. I also created a grid point class which represents points in the maze. I also completed the Grinch class, which represents the Grinch in the game. I wrote up Javadocs comments for most of the classes. I also contributed to the screen and actor classes.

<u>Responsibility</u>
All: Screen, ScreenObject, Main
Ishwar: DrawingSurface, HauntedMaze, Officer, GameScreen
Arindam: OptionScreen, Antagonist, Item, Blueprint, Infobar, EndScreen
Dhruv: Trap, MazeData, Grinch, Encryption, gridP

Sources (links here)

-