Jack Higgins, Joseph Sembower
Prof. Montgomery
CSCI 4448
5 May 2023

Project 7 - Slytherin

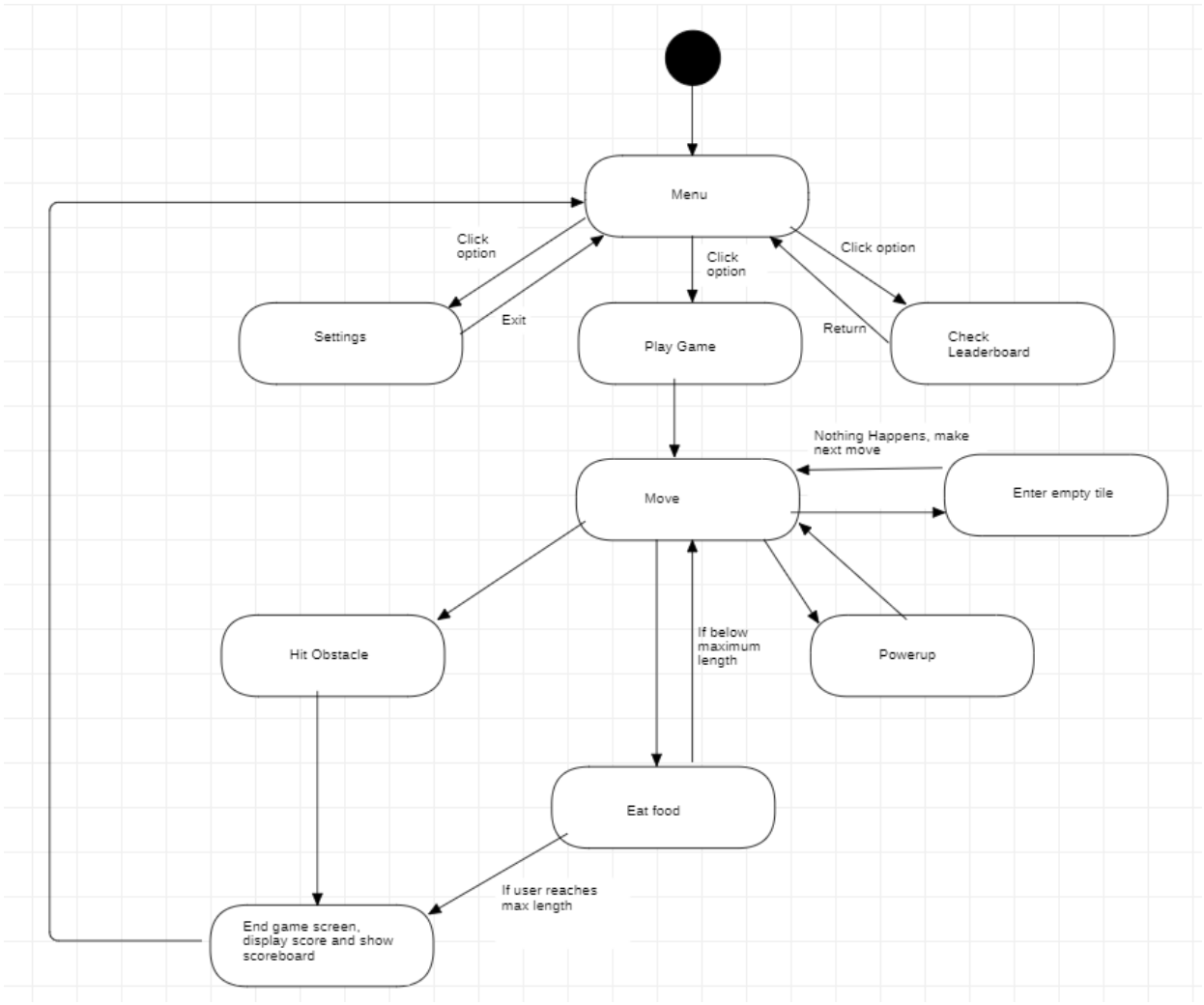
Final State of System Statement

Slytherin is a working snake game with working menus including a leaderboard and a directions page. When you start the program you are greeted with a menu. The options from the menu are to play the game, view the leaderboard, or look at the directions. The leaderboard is a csv file containing the name of the player, the score they got, and the date when they played. Snake game works by controlling a snake and trying to get as many pellets as you can. When the snake collides with a wall or its body, the game is over. We added 2 spells into the game. One decreases the speed of the snake for the player, allowing them to maneuver easier. The other moves the food pellet such that if the pellet is in a poor spot, the player can move it. After playing the user is asked to enter a name to store their score on the leaderboard. Once this is done, the program closes.

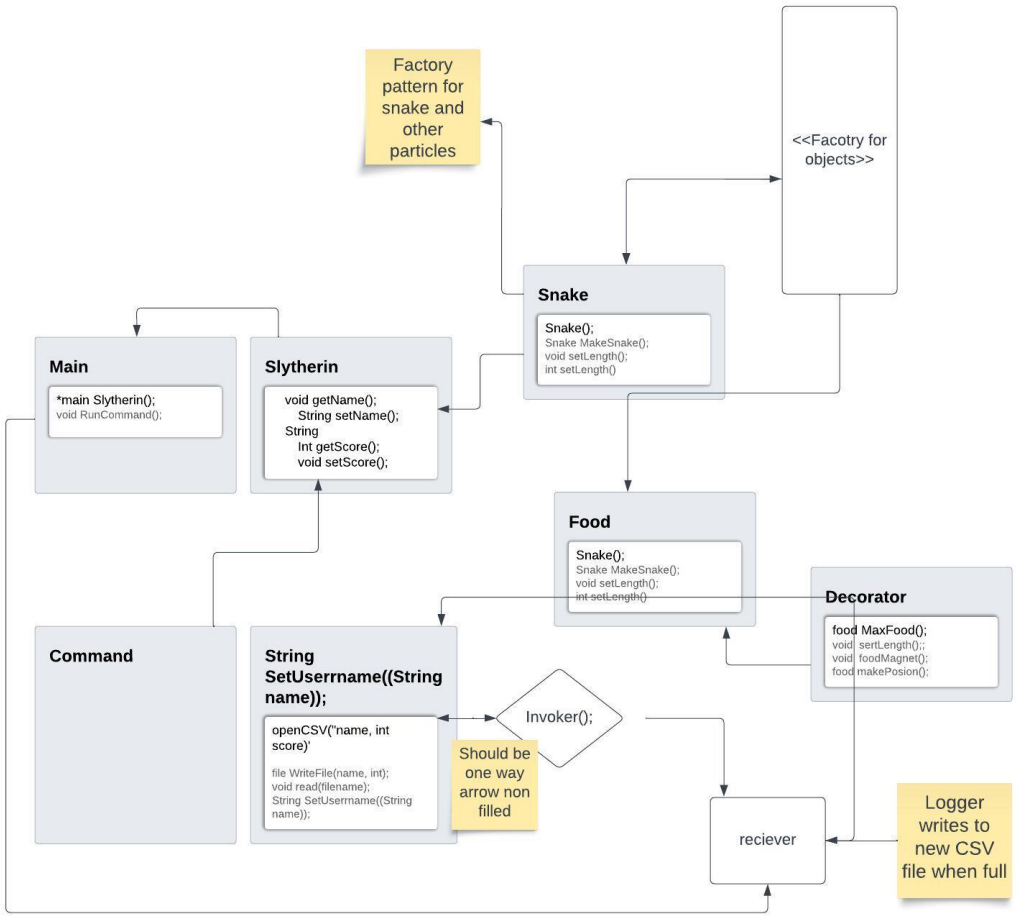
Due to time constraints and UI issues some things were changed during development. The spells were simplified and balanced. The original idea of infinite food was too game breaking. The speed was decreased instead of increased to make it easier for the player. The windows are fixed size instead of scaled to the user.

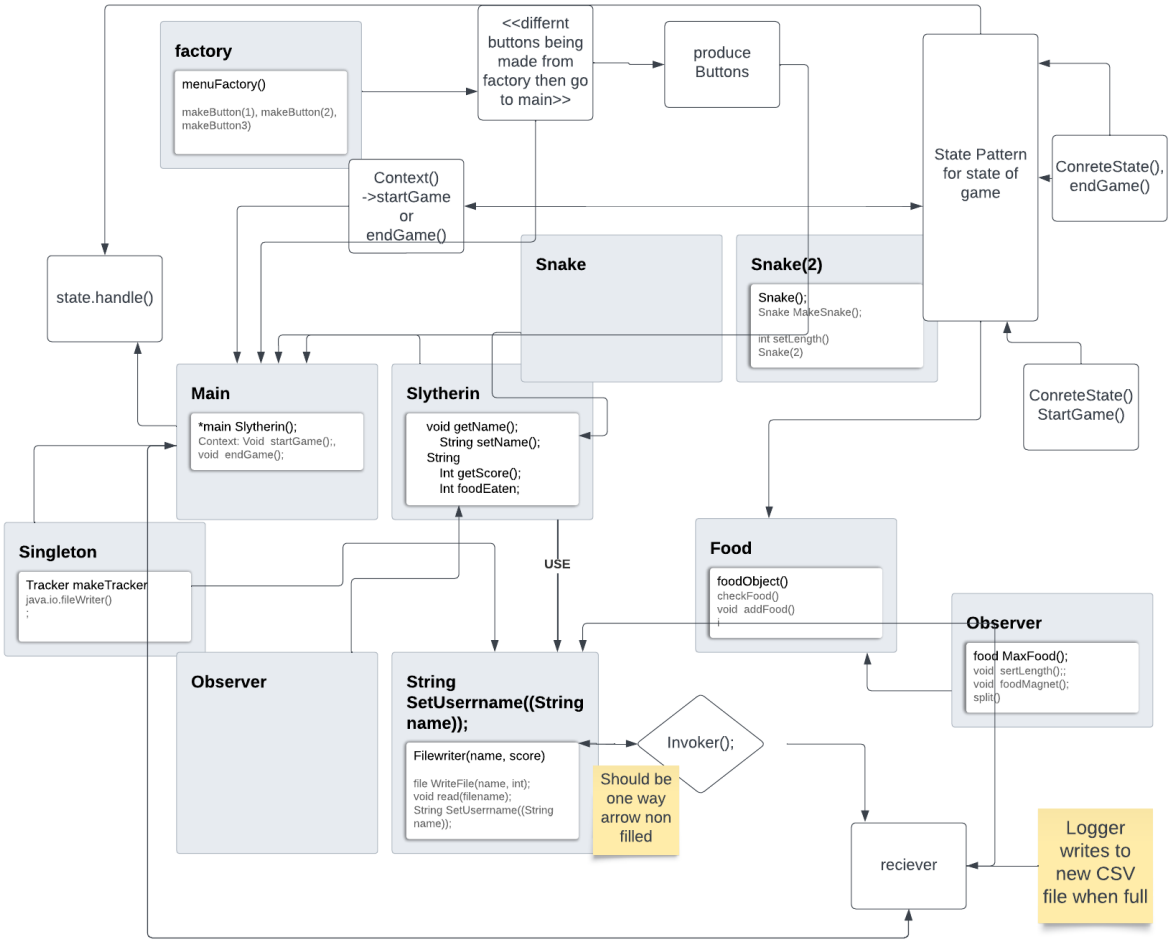
Final Class Diagram and Comparison Statement

Previous UML Diagrams:

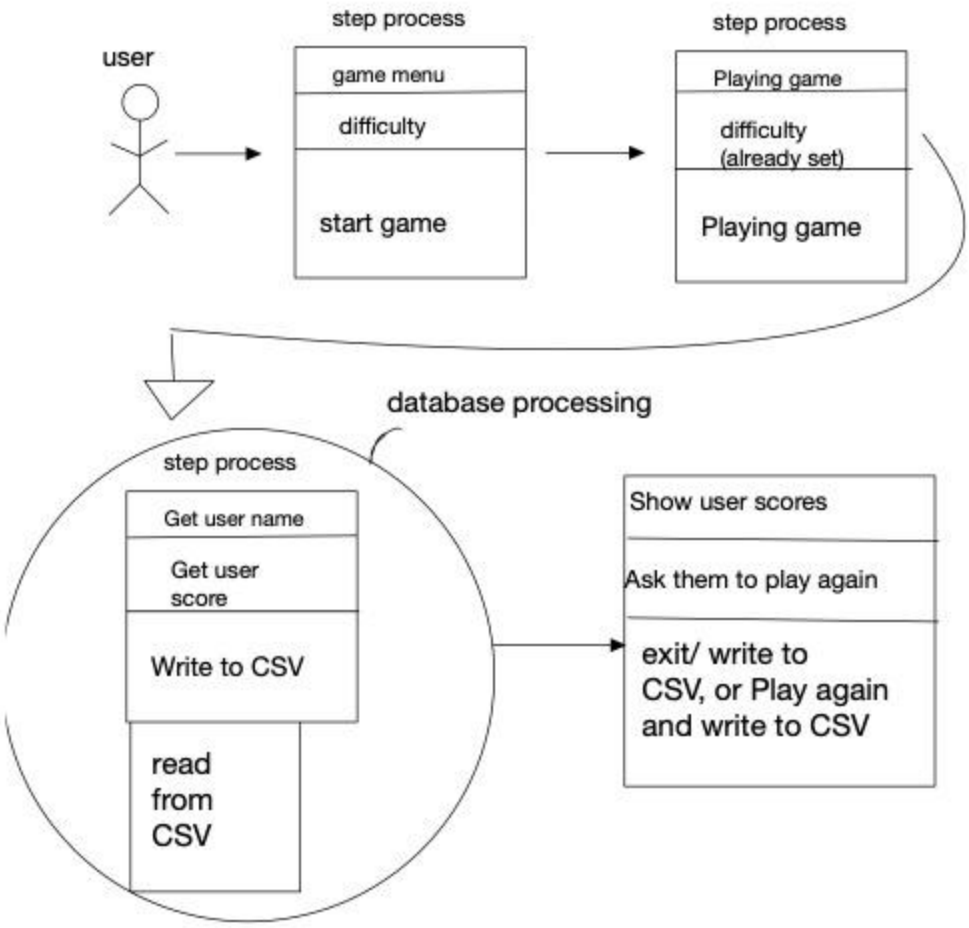


Project 5 UML followed by project 7 UML:


Factory pattern for snake and other particles
<<Facotry for objects>>
Snake
Snake();
Snake MakeSnake();
void setLength();
int setLength()
Main
*main Slytherin();
void RunCommand();
Slytherin
void getName();
String setName();
String
Int getScore();
void setScore();
Food
Snake();
Snake MakeSnake();
void setLength();
int setLength()
Decorator
food MaxFood();
void sertLength();;
void foodMagnet();
food makePosion();
Command
String SetUserrname((String name));
openCSV("name, int score)'
file WriteFile(name, int);
void read(filename);
String SetUserrname((String name));
Invoker();
Should be one way arrow non filled
reciever
Logger writes to new CSV file when full



Some key changes:
As you can see from the project 5 UML and the project 7 UML was the change of design patterns and functionality that is used in the program. The snake is not presented as really an object in the project 7 UML unlike the 5 UML, this goes as the same for the food object. We did not use the command pattern or the decorator pattern as we initially thought we would.



data is stored locally

Pattern usage:

- Observer

The observer pattern was utilized to write to a CSV for the leaderboard. This reduced the number of times the csv had to be scanned. A tracker object was created which would store the database in String form until the time came to write to the csv.

- Singleton

The Tracker object follows the singleton pattern. We used lazy instantiation to ensure that the object could only have one instance.

- Factory

A factory pattern was used to create the different menus used in the game. When MenuFactory is called with a text of what menu is desired, that menu is created

- State

The state pattern was initially introduced to be able to tell if the game was itself in a state or running or if the game was over. This was used to really test to see that the game was being run in the background from creating the game panel.
JOSEPH

Key Changes:
The four patterns we initially were going to use: Factory, Decorator, Command and Logger were not all used. Instead we used Factory, Logger, State, and Singleton.

For project 5 we never actually created a class UML. In terms of functionality, very little changes were made. The architecture diagram we created is essentially the same. The only difference is that we do not ask the player to play again. After entering the name, the program is closed. The activity diagram we created is the exact same as what was created in project 7.

Third-Party code vs. Original code Statement
• A clear statement of what code in the project is original vs. what code you used from other sources
– whether tools, frameworks, tutorials, or examples – this section must be present even if you used
NO third-party code - include the sources (URLs) for your third-party elements
https://docs.oracle.com/javase/tutorial/uiswing/components/button.html
This was a tutorial for JFrames and JButtons. I used their demo to start my menus. No code ended up being used to the best of my knowledge.

https://www.baeldung.com/java-csv
This was used to figure out how to write to a csv without OpenCSV.

https://www.youtube.com/watch?v=S4lPjokjHWs
This video was used to help understand how to implement the snake game itself using swing in java.

https://www.geeksforgeeks.org/state-design-pattern/
GeeksforGeeks was used to help understand how the use of the state pattern other swing examples for creating text objects.

Statement on the OOAD process for your overall Semester Project

- We had no understanding of Swing UI or Java UI. We never used UI in class and Java is not a fun language to create UI in. It took a lot more time than previously thought to get a handle of Java UI in order to start getting code done. The UI still does not look perfect upon submission but the functionality works.
- The UML activity and architecture diagrams ended up being very helpful in giving us a vision of the final goal. We split tasks so having an end goal to cite was useful. We followed these diagrams closely during development.
- Having a line of communication through discord was also very helpful over the course of the semester. Our group could tell the others about issues we were facing and help each other. We could also hold each other accountable on deadlines we set for ourselves. Our communication throughout the course of the semester was very productive.