Score Attack Game
Group 10
Part 5
D'Vreaux Fontaine, Qi Pe, Dongyao Wang

1. List the features that were implemented

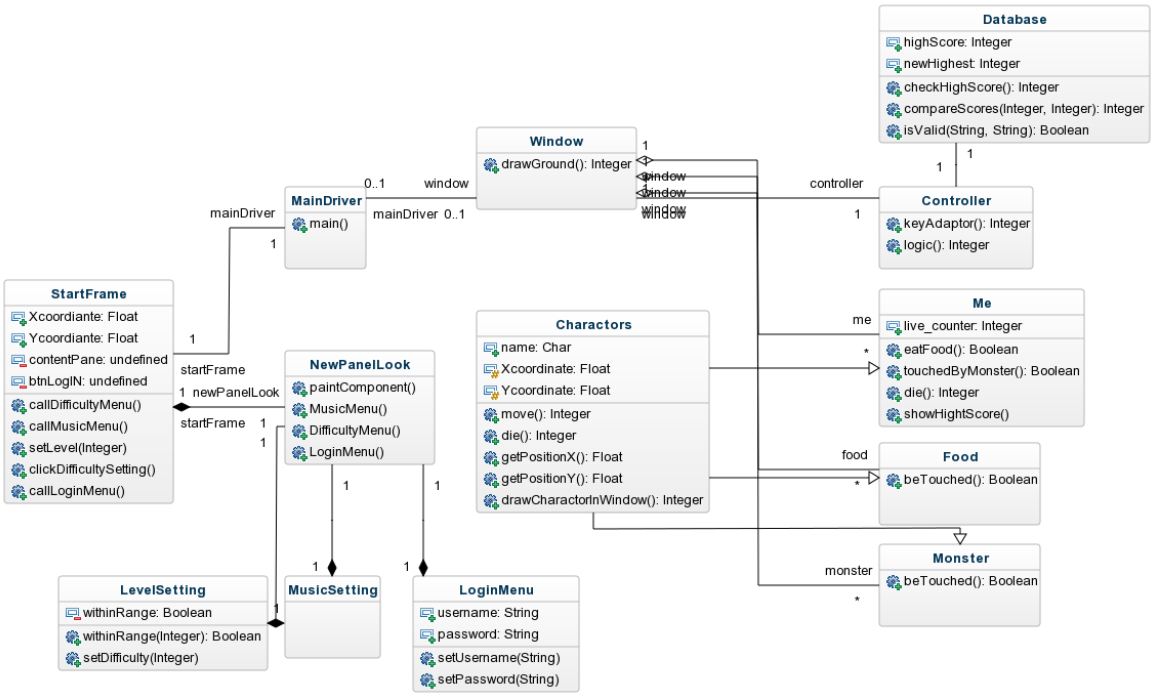
ID	Title
BR-001	Can be installed on any computer with JVM
FR-001	Click Start to Start Game
FR-004	Use 4 arrow keys to move player in that direction
FR-006	Lives counter increase when player eats food
FR-007	Player dies when live counter reaches zero
FR-008	Collision Detection between objects exists
FR-009	Store scores in database
NFR-001	Game installation time should be under 15 seconds
NFR-002	The key response time be under 0.02 seconds
NFR-003	Size of game under 1 Gb
UR-001	User should be able to sign up by typing in a name
UR-002	User should be able to login
UR-003	User should be able to check the instructions
UR-004	User should be able to check settings
UR-006	User should be able to check highest score by account name
UR-008	User should be able to start game by pressing start
UR-009	User should be able to move character with the arrow keys
UR-010	User should be able to save their highest scores automatically
UR-011	User should have their lives decreased by touching a monster

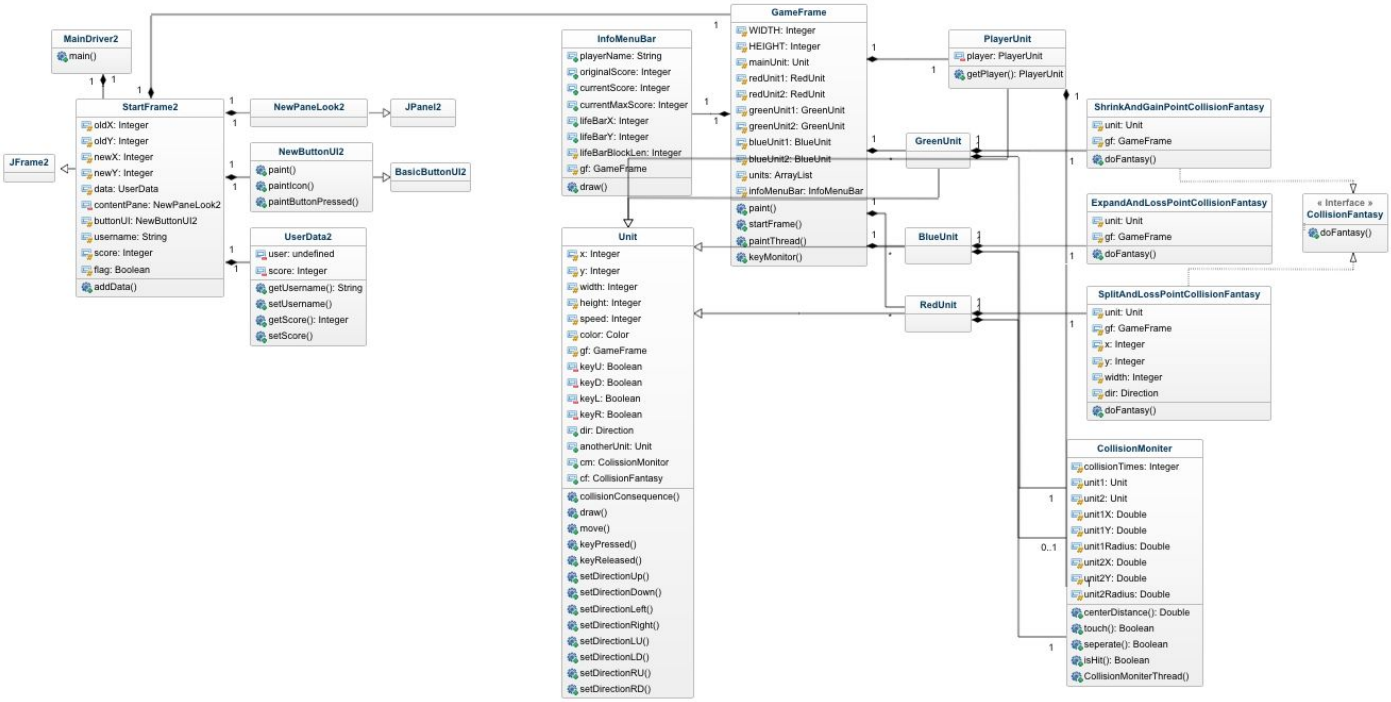
UR-012	User should have their lives increase by eating food
UR-013	User should be able to check team information

2. List the features that were not implemented

ID	TItle
BR-002	One email address per account
FR-002	Click settings to set up background music
FR-003	Choose different level for difficulty
FR-005	Lives counter is full when game starts
UR-005	User should be able to set difficulty level in settings
UR-007	User should be able to turn on/off music

3. Show your Part 2 class diagram and your final class diagram





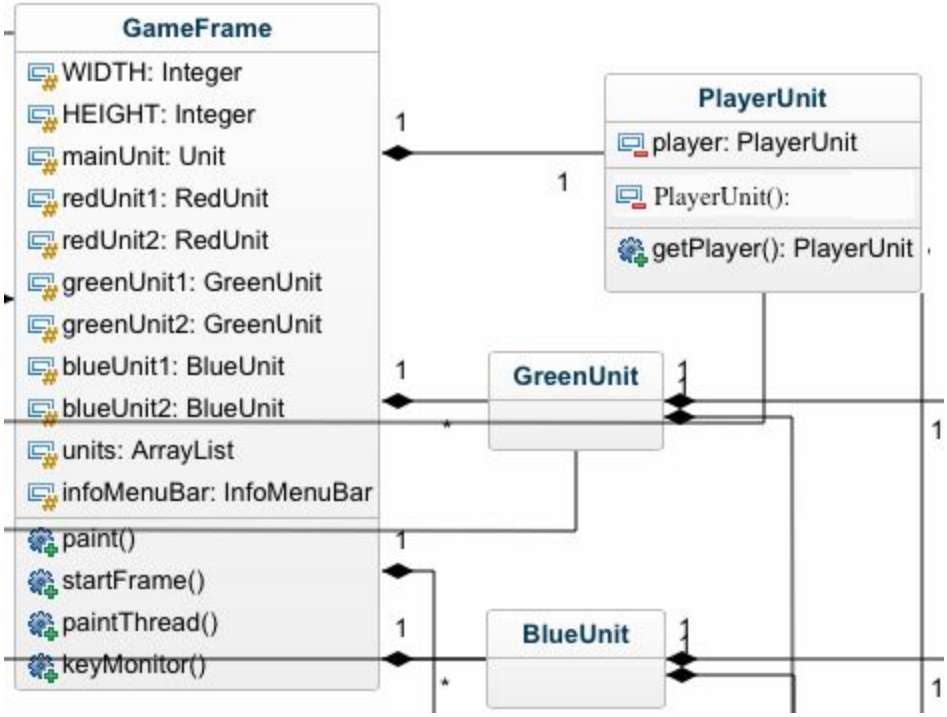
What changed? Why?

A lot changed from our first class diagram. The StartFrame, PanelLook, MainDriver are called the same but now are updated with the functions and variables that allow them to work with everything. Character became Unit, Food became Green Unit, Monsters became Red Unit, and Me became Player Unit. Each of those has been updated to included (or not include) new functionality based on the design patterns we learned and what needed to change to actually make it work. Things that were added were the InfoBar, the entire collision system, and a simplistic way to show settings using the buttons. I feel as if we knew about the design patterns before making these, then we would have changed a lot less. However just looking at both of them, it is clear that we were simplifying the project as a whole. The structure is pretty much the same, but how much we needed was far greater than predicted.

4. Did you make use of any design patterns in the implementation of your final prototype?

Yes! We used the simpleton and strategy design patterns.

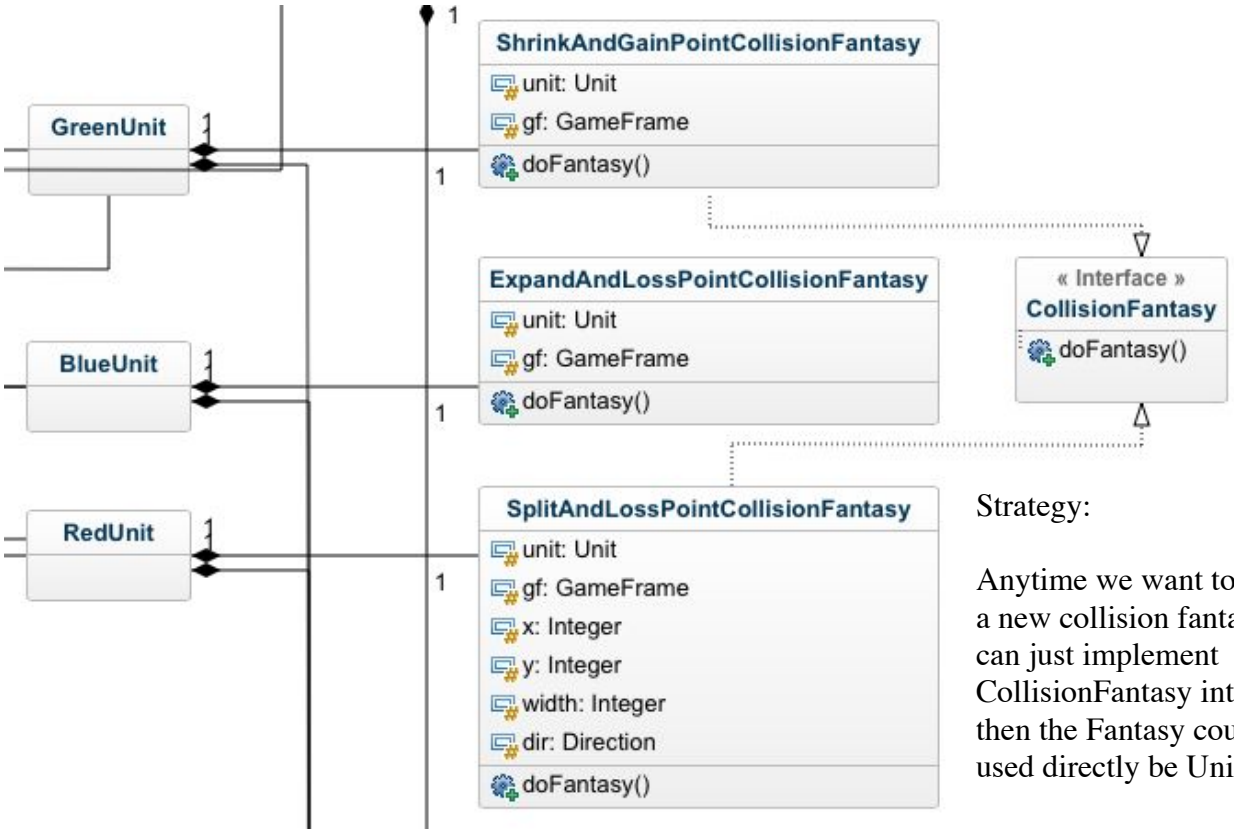
Simpleton:



Singleton:

First declaring all constructors to be private, then provide a static method that return the reference of the instance.

Strategy:



Strategy:

Anytime we want to create a new collision fantasy, we can just implement CollisionFantasy interface, then the Fantasy could be used directly be Unit.

5. Knowing all of the things we can use would be helpful at the start of the design and analysis of a system. Not knowing about the design patterns made us change a lot of our initial class diagram. However even without those, having the diagrams helped develop the individual parts of the system without slaving over how to do it. Implementing design patterns with knowledge that you are going to use a pattern is incredibly helpful long term and not even that difficult. However, putting design patterns into a system can be hard to do, especially if you don't need to implement. Of course, this may be due to our relatively new exposure to everything. I think that this information will help in future development of systems. In fact, two of our group members have already used the things we have learned in development of other systems.