Design Manual

Team Bixby

Introduction:

Within our program we wanted to stay true to the original Tamagotchi design while also adapting the interactions and interface to the PC environment. To do this, we broke down our design into several stages. For the first stage, we wanted to design a Pet object that would track all the necessary stats and interactions. The Pet object would include attributes such as name, happiness, sleepiness, etc and would control the interaction of putting the pet to sleep or feeding the pet. To control the stats of the pet we created functions that dictated how quickly the pet's stats would drain and how much they would increase or decrease depending on what you fed them, when you put them to sleep, and the age of the pet. Next, we designed the GUI that would include the images of the pet and the interactable interface and buttons. We followed the MVC model for designing the GUI with JavaFX. Finally, once we had both the Pet object and the GUI set up we had to tie the two parts together making the GUI read and represent all the attributes of the user's pet. The GUI would update depending on the condition of the pet, for example the stats bar would change to indicate the health and condition of the pet and the pet's image would change according to its age.

User Stories:

1.) As a user, I would like to be able to see a visual representation of my pet on the screen

2.) As a user, I would like to create, name a new pet

3.) As a user, I would like to pet my pet

4.) As a user, I would like to feed my pet

5.) As a user, I would like to access the stats for the pet

6.) As a user, I would like to save my progress

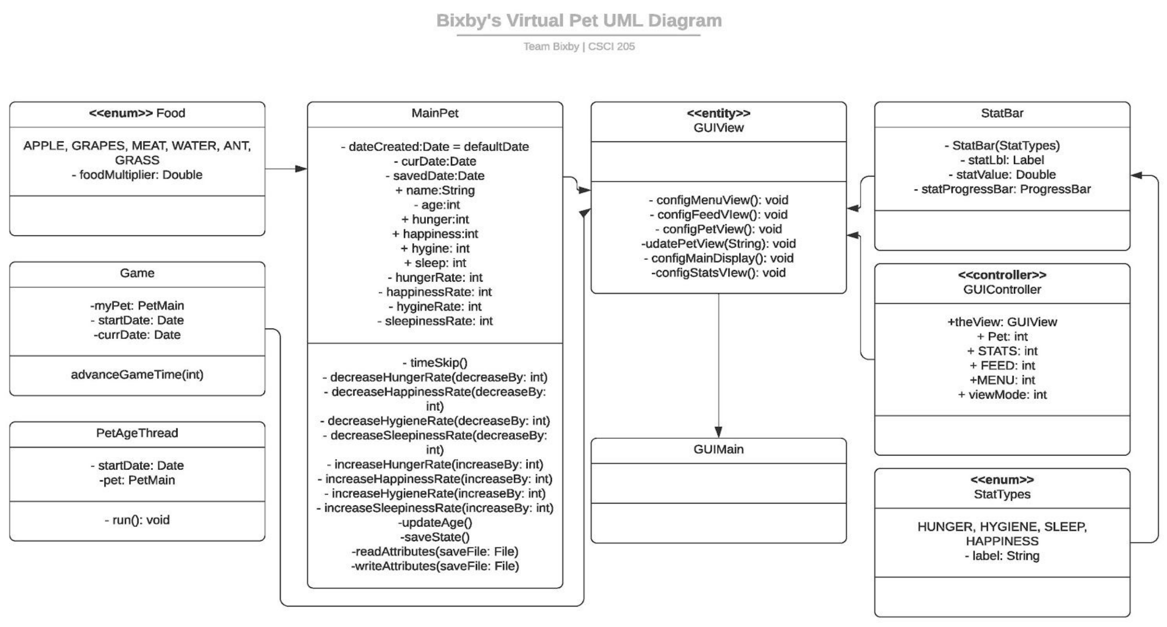
7.) As a user, I want my pet to age (have a life cycle)

8.) As a user, I want to be able to skip time

9.) As a user, I want the pet to evolve

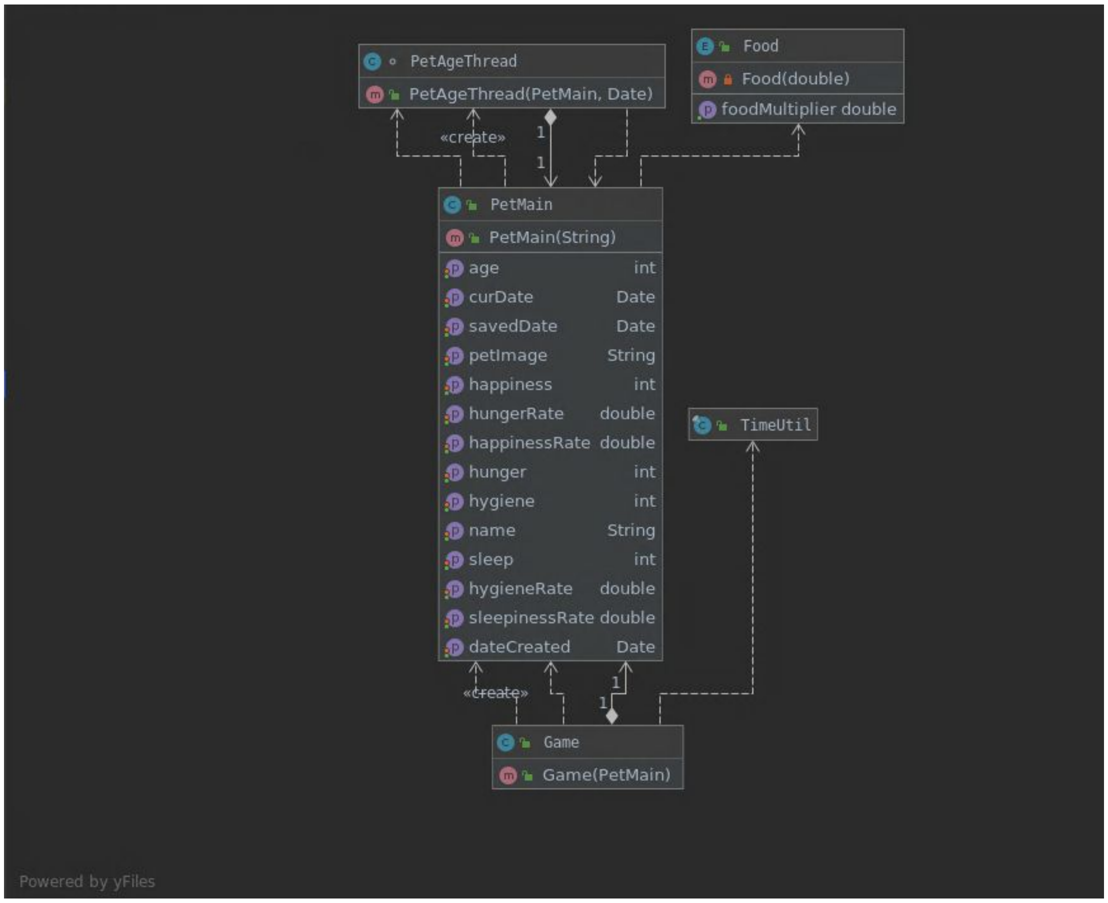
Object-Oriented Design:

Our design was broken down into two separate sections. One section with the classes that controlled the GUI and the other section with the classes that controlled and managed all the attributes and interactions of the Pet. These two segments of our program can be better illustrated in the UML diagram below.



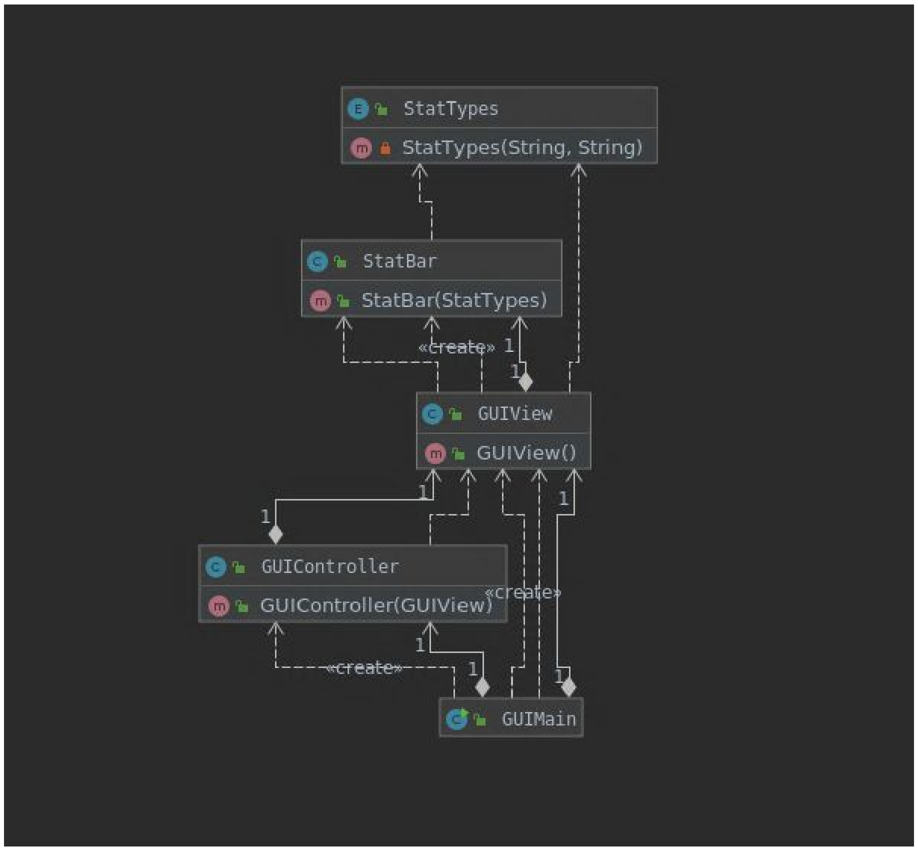
In the UML diagram, the PetMain, Food, and PetAgeThread classes control all the attributes and actions pertaining to the pet. The GUIView, StatBar, GUIController, StatTypes, Game, and GUIMain control the actions and representations of the GUI.

Controlling the pet:



- **PetMain:**
  - The PetMain class was implemented with the basic functionality of what a 'pet' would be like, and how the player could interact with it. This includes the actual implementation of the Pet class, which consists of the pets name, age, all its statistics and the rates at which the stats decrease.
  - All the interacts with the pet and the GUI are also through functions in the pet class.
- **Food:**
  - Food is an enumerated type that keeps track of the different types of food and how each of them affect the hunger attribute of the pet.
- **PetAgeThread:**
  - The PetAgeThread class creates a thread by implementing the Runnable interface. This thread is started in the constructor of the PetMain object and tracks the age of the pet and sets the corresponding image of the pet depending on the age.
- **TimeUtil:**
  - Utility class to keep track of the current time of the game instance. This class creates a correlation between the actual time and the age of the pet.

Controlling the GUI:



- **StatBar:**
  - The class StatBar represent the statistics of the pet (hygiene, sleep, hunger, happiness). It includes the GUI elements of these statistics which are: a label for the type of pet statistic and the progress bar that represents the statistics bar in the GUI. This class also includes a method to update the value of the progress bar (that appears on the GUI)
- **GUIView:**
  - GUIView includes all the visual elements of the GUI. The GUI root is an AnchorPane and is not re-sizable. There are various different “view” elements included in this class. These elements are drawn on the screen when required. For example, the stat view appears on the screen when the stat button is pressed. The stat view element is stored in this class.
- **StatTypes:**
  - StatTypes is an enumeration that contains all the types of statistics that a pet has. These include hygiene, sleep, hunger, and happiness. The enumeration stores a unique color for each type of statistic (used for its respective progress bar in the GUI) and a label for the type of the statistic.
- **Game:**
  - This class simply helps start the game and also generate the age of the pet at the time of the start.

- **GUIController:**
    - GUIController class makes the elements from GUIView interactive by using the PetMain class as a model. The GUIController class makes all the buttons work and switches the “views” as required.
- **GUIMain:**
    - GUIMain class basically runs the GUI by combining the functionality of PetMain, GUIView, and GUIController.