COSC 1437 Tamagotchi Pet Game Final Report

Name: Ashmal Macknojia
PS ID#: 2171115

THE PROGRAM & CLASSES DESCRIPTION

In this Tamagotchi Pet Game Project, the program was designed and developed with the idea of starting off by preparing a midterm report prior to doing any coding. During the start of the project, I noticed that coding things in the start wasn't really a useful start because it allowed me to get my thoughts into code; However, it seemed like I was missing a lot of small details that make up for an entire project which is why I decided to first think about preparing my midterm report which could assist me with the process of coding everything later and making changes then as suggested by Professor Dan and his TA's. As I was planning out my midterm report, I kept the requirements stated in the rubric into consideration to ensure that my project meets the requirements. Based on the requirements, I decided to think about what the function of the project would be to start preparing and doing what's needed to meet the requirements in which I was able to come up with an idea of how my game would consist of a base class that will consist of all the protected members along with public functions that will be virtual as they'll be reused by the derived classes. In this case, my derived classes were three different Pokémon types with their name set by default which are Electric (Pichu), Ground (Sandshrew), and Ice (Sandshrew).

Within the base class called Tamagotchi.h, I had 9 public member functions that were reused by the derived classes as those functions dealt with modifying the stats of the Pokémon selected by the player and also saving and loading the game progress made by the player if the player wishes to save and exit the game or load up a game with saved progress from previous attempt. The protected members were Hungerness, Sleepyness, Boredness, Happiness, and petName where the first 4 served as the stats of the Pokémon on a scale of 1-100 while petName was set by default to Pichu and Sandshrew, which was later updated to what the player wanted to name their pet. Along with the fact that all of the public functions were reused by the derived classes, each derived class also consisted of a setter's function which played a role in updating petName to what the player inputted for the name during the game loop so that the print statements use the name specified by the player rather than the names set by default.

The main idea or purpose of this game was to have 3 derived classes of different Pokémon that consist of functions that allow the player to save their game progress, feed, treat, play, rest, and gym their selected Pokémon which they named as that would fluctuate the levels of Hungerness,

Sleepyness, Boredness, and Happiness uniquely as every Pokémon's stats of protected members fluctuated differently. Along with the stats fluctuating based on the input from the user of which function they would like to choose or what they would like to do with their Pokémon, there was a nextHour() function that printed unique warning messages to the player if stats were too low or high for a certain protected member which possibly suggests the player to do certain things in order to get rid of the warning message. Because the protected member's stats were on a scale of 1-100, there was a gameRangeLimitation() function that adjusted the stats level to 0 if they went below 0 and 100 if they went above 100. This overall game didn't have a win-or-lose situation for the player, but rather a chance to select a Pokémon of three different types, name it, and modify it by either feeding, treating, playing, resting, or gymming it unlimited times to see the stats fluctuate and deal with warning messages coming up until they decide to select the option of saving and exiting the game, in which the stats of all the protected members get saved into a saveTheData.txt file which keeps the game progress until the player decides to either load up that progress and continue playing or just start a new game to where the progress of that game gets saved and updated into the same file.

Special Instructions to Operate The Program

The special instructions to operate the program of this game were as follows throughout the main.cpp in making up the menu for the user: The player is greeted with a starting message that basically welcomes them into the game and then asks them if they'd like to continue playing with a game that they saved from a previous attempt or start a new game. These two options are dealt with in the game loop by giving the player an option to type in 1 if they would like to continue with a saved game or type in 2 if they would like to start a new game, which involves input validation through the use of a while loop that ensured player isn't typing in anything besides 1 or 2 and if they do then they'll be asked to type in the option again till they type either 1 or 2. If they selected option 1, then they're asked which class Pokémon they selected that was saved, allowing them to type in 1 for Electric(Pichu), 2 for Ground(Sandshrew), and 3 for Ice(Sandshrew). This input process also goes through input validation the same way it went with previously selecting options 1 or 2 with a while loop. Based on their input, there are 3 if-statement blocks that have a class object assigned to the class Pokémon where that object calls the loadTheData() function which loads up the game progress. After this, there's a while true loop that consists of pictures from ASCII art of the Pokémon along with other creative art that was coded, and it tells the player about different ways that they can modify their selected Pokémon and what to type in for the certain modifications or functions to do their job. These functions or modifications consist of typing 1 to save the game progress and exit from the game, 2 to feed, 3 to treat, 4 to play, 5 to rest, or 6 to gym which involves input validation through the use of a while loop that asks the user to enter either 1, 2, 3, 4, 5, or 6. When the user types in their input, the function is called in that class by the class object which prints out the print statements and updates the stats of the protected member until the player decides to type in 1 to save their game progress and exit from the game. As part of the stats updating, all the protected members except petName are set to 45 by default to give the player a start with that number. Once they save and exit from the game, their progress or stats of the protected members will be

saved into the saveTheData.txt file for them to continue with if they'd like when they decide to play the game again.

The idea of the player typing in 2 for their option of starting a new game, the game loop is exactly the same as it is if they were to type in 1 with the exception that there would be no load function getting called as there's no game progress to load from previously played game attempt, the menu would this time ask them what they'd like to name their Pokémon as the player hasn't been given the opportunity to name it yet since they've started a new game. After that, the menu goes the same way with the exception that they would be given pictures of three different Pokémon types to choose from rather than getting a print statement to select which one they had chosen in the previous attempt because they should be given a picture of three to choose from, especially when they're playing the game for the first time. So, this wraps up on the idea of special instructions to operate the program or the game in order for the player to select a Pokémon, name it, and modify it unlimited times till they save and exit from the game.

IMPLEMENTATIONS DIFFERED FROM THE PLAN

After getting wrapped up with the midterm report and starting on the actual coding by following the midterm report on what functions will be used in each class and what implementations would be like along with the main.cpp, I noticed that my code was compiling but not producing the proper output. This was caused by not having the curly braces exactly where they needed to be, causing an output error for the function I was able to learn how the idea of curly braces being located where they need to be plays a crucial role in the idea of functioning a program properly and can be one of the things to look out for in the future as part of the debugging process as there might always be an error telling us something about the curly braces. Along with the curly brace mistake, I had noticed that when I was receiving the outputs from the functions when called based on the player's input, the name of the Pokémon was being printed out as the default name set to the Pokémon rather than the name set by the player to where I had to remove the className::className(string name) constructor that was being called and meant to update petName to whatever name is as it didn't seem to do its job as supposed by the object calling it. The solution to this situation was the change it into a setter function implemented as className::setName(string name) which was being properly called by the object of the Pokémon class selected by the user and was then printing out the actual name set by the player. Without this change, the purpose of the player getting to name their Pokémon would hold no value. Along with that, I also came to notice that my pseudocode was quite off but needed a minor fix to work as it needed when the player decides to select the option to load a previously saved game progress as it wouldn't print anything past selecting the Pokémon chosen in the game since I forgot to add the if-else statements that were added in the else statement when user types in 2 for starting a new game as this part slipped out my thought although I had an idea of the game loop is similar for both with minor exceptions. This change was easy to fix as I just had to type up the loop of the game menu that I had beneath my else block if the user were to type in 2. From there, I was able to clean it up a little by fixing the indentations and such. After everything worked

properly, I was suggested by Professor Dan and the TA's to make the project more creative in the sense that I shouldn't just have my print statements just printing, but to also add like more ASCII art pictures, space out my code, and add like some lines to make my menu look nicer when it prints to the player just for the creative look it can bring to the game and the player which was something I took into consideration by adding a SAVED! Picture from ASCII Art, adding a good amount of spacing between every print statement, adding a picture of selected Pokémon each time the loop iterates to allow the player to keep modifying it, and also adding lines created with ^, =, or @ to make the print statements or menu look more creative.

Despite such mistakes made throughout the pseudocode, I still felt like the idea of a midterm report was wonderful as it helped make the project easier to tackle in the sense that we were given the required opportunity to plan everything out and then code rather coding entirely and missing out on the small or big details as the midterm report itself created an opportunity to think and jot down our ideas and organize them into a UML-design for our classes and also have pseudocode on how the game loop would look like. This idea had really played a strong emphasis when it came to asking the professor or TA's any questions we had with our idea of the game and coding things up without too many worries about things functioning properly or not as I was able to get feedback from Professor Dan and his TA regarding my midterm report to ensure things are properly mentioned as implemented in the planning process to avoid having to go too much trouble later with things not working as needed.

This overall project had played a great impact in helping me improve my programming abilities by thinking in a systematic way as this was my first creative project where I actually got to plan things out for the first time and execute them as I would always dive straight into coding in the past as I didn't really learn the actual way to solving greater problems in programming. From this project experience, I plan to hopefully create and work on more creative projects, plan things out before actually executing the code, keep getting feedback, take the programming mistakes I got to learn from this project into consideration for future debugging purposes and try making any project bigger after completion from new things I learn in programming over time. Without the idea of creativity of such a project and the way this project was done in a systematic way, I would definitely say with no doubt that it made a huge difference in being able to plan and execute things as a programmer, otherwise I probably would've continued making the same mistakes I made in the past to go straight into the code and go through the troubles of things not working as needed in a harder way to debug and doing this project in such a way has helped me increase my confidence to be able to do more creative projects and be able to actually plan things out in a organized way and then execute them with code.