COSC 1437
TAMAGOTCHI PET GAME PROJECT
MIDTERM – REPORT

Name: Ashmal Macknojia

PS ID#: 2171115

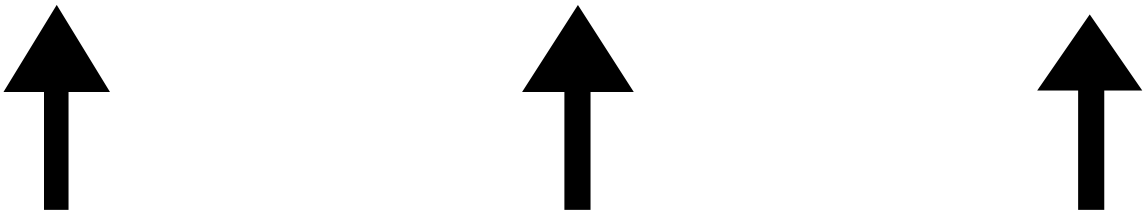
DESIGN OF THE PARENT AND INHERITED CLASSES

In this Tamagotchi Pet Game Project, there will be an implementation of a parent class and three inherited classes that inherit from the parent class. The parent class of this project will be called Tamagotchi which consists of the necessary public, private, and protected members that will be useful throughout the game and reuse some of those members in the inherited classes to function in the game. This game will focus more on the animal theme of Pokémon where the inherited classes will consist of three different animal- theme-based Pokémon, which will be Pichu and Sandshrew of their two different types which are ground and ice. The inherited classes will be Electric, Ground, and Ice. Each inherited class will have access to the protected members from the parent class and by default, the stats of the Pokémon will be set to 45 in order for the player to start playing without any issues and make modifications to its chosen Pokémon character with the exception that the protected member petName will be set to an empty string or no name by default since it's a string datatype.

THE UML DESIGN OF THE PARENT AND INHERITED CLASSES

#: Protected
+: Public

TAMAGOTCHI
#hungerness = 45: int #sleepyness = 45: int #boredness = 45: int #happiness = 45: int #petName = "": string
+ Tamagotchi() : + virtual saveTheData() : void + virtual loadTheData() : void + virtual feedTheAnimal() = 0: void + virtual treatTheAnimal() = 0: void + virtual playWithTheAnimal() = 0: void + virtual restTheAnimal() = 0: void + virtual gymTheAnimal() = 0: void + virtual gameRangeLimitation() : void + virtual nextHour() : void



Electric
+ Electric(string name); + saveTheData() : void + loadTheData() : void + feedTheAnimal() : void + treatTheAnimal() : void + playWithTheAnimal() : void + restTheAnimal() : void + gymTheAnimal() : void + gameRangeLimitation() : void + nextHour() : void

Ground
+ Ground(string name); + saveTheData() : void + loadTheData(): void + feedTheAnimal(): void + treatTheAnimal(): void + playWithTheAnimal() : void + restTheAnimal() : void + gymTheAnimal() : void + gameRangeLimitation() : void + nextHour() : void

Ice
+ Ice(string name); + saveTheData() : void + loadTheData() : void + feedTheAnimal() : void + treatTheAnimal() : void + playWithTheAnimal() : void + restTheAnimal() : void + gymTheAnimal() : void + gameRangeLimitation() : void + nextHour(): void

TAMAGOTCHI PARENT CLASS UML DESIGN DESCRIPTION:

• This parent class consists of 5 protected members with descriptions provided below:

1. hungerness = 45
2. sleepyness = 45
3. boredness = 45
4. happiness = 45
5. petName = “”

The protected members like hungerness, sleepyness, boredness, and happiness are set to the value 45 in this class because the players will be given a menu to pick a Pokémon by its class type and when they decide on which Pokémon to pick, they will have the ability to access various functions that adjusts their stats depending on the goal of that function. The reason they’re set to 45 is because the player should be given an initial value to play around with their character Pokémon to help them start making modifications properly.

• The Virtual Overriding Functions:

1. virtual void saveTheData()

//saveTheData function will print out the saved data of the game if the player decides to save their stats rather than continuing to play with other options. This function will be reused by all inherited classes. This function will autosave the player’s progress and will then break out of the game.

2. virtual void loadTheData()

//loadTheData function will load up the data of the stats of the Pokémon class selected by the user in order to start playing and modifying the stats. This function will be reused by all inherited classes.

3. virtual void feedTheAnimal()

//feedTheAnimal function will allow the player to feed their selected Pokémon from the chosen function class and this function will be reused by all inherited classes.

4. virtual void treatTheAnimal()

//treatTheAnimal function will allow the player to treat their selected Pokémon from the chosen function class and this function will be reused by all inherited classes.

5. virtual void playWithTheAnimal()

//playWithTheAnimal function will allow the player to play with their selected Pokémon from the chosen function class and this function will be reused by all inherited classes.

6. virtual void restTheAnimal()

//restTheAnimal function will allow the player to rest their selected Pokémon from the chosen function class and this function will be reused by all inherited classes.

7. virtual void gymTheAnimal()

//gymTheAnimal function will allow the player to gym or exercise their selected Pokémon from the chosen function class and this function will be reused by all inherited classes.

8. virtual void gameRangeLimitation()

//gameRangeLimitation function will play a role in limiting the number assigned to all the protected members of type int to where it assigns 100 to those members if they're greater than 100 and 0 if they're less than 0. This function will be reused by all inherited classes.

9. Virtual void nextHour()

//nextHour function will play a role by printing out a warning message if their protected members of type int are going low or high so that the player can take actions in being able to decrease or increase the number assigned to that protected member by choosing another member function that could help it. This function will be reused by all inherited classes.

ELECTRIC INHERITED CLASS UML DESIGN DESCRIPTION:

• This inherited class consists of 10 public members that consist of overloaded constructors and various overriding functions that are functions of the game where players have the ability to select from the menu which particular function they want to access and some of those functions would make adjustments to the character's stats while some may print an alert message when stats are too low or high, and even just save or load the stats.

• The Overloaded Constructor:

1. Electric(string name)

#During the start of the game, the user will be given an input statement to enter the name of their selected class Pokémon. Once they enter the name, the name they enter is assigned to the variable name in the string name, so this overloaded

constructor would be implemented in the Electric.cpp file which will assign the protected member petName to name.

• The Overriding Functions:

1. void saveTheData()

   //saveTheData function will print out the saved data of the game if the player decides to save their stats rather than continuing to play with other options. This function will autosave the player's progress and will then break out of the game.

2. void loadTheData()

   //loadTheData function will load up the data of the stats of this Pokémon class selected by the user in order to start playing and modifying the stats.

3. void feedTheAnimal()

   //feedTheAnimal function will be an option given to the user to feed this chosen character which will increase happiness by 5, increase sleepyness by 5, decrease hungerness by 10, and decrease boredness by 5. This function will also print a unique message telling the user they fed and how much the stats fluctuated.

4. void treatTheAnimal()

   //treatTheAnimal function will be an option given to the user to give this chosen character a treat which will increase happiness by 10, increase sleepyness by 5, decrease hungerness by 5, and decrease boredness by 10. This function will also print a unique message telling the user they treated the character and how much the stats fluctuated.

5. void playWithTheAnimal()

   //playWithTheAnimal function will be an option given to the user to play with this chosen character which will increase happiness by 15, increase hungerness by 15, decrease hungerness and boredness by 10. This

function will also print a unique message telling the user they played with the character and how much the stats fluctuated.

6. void restTheAnimal()

   //restTheAnimal function will be an option given to the user to rest this chosen character which will increase happiness by 5, increase hungerness by 10, decrease sleepyness and boredness by 15. This function will also print a unique message telling the user they rested the character and how much the stats fluctuated.

7. void gymTheAnimal()

   //gymTheAnimal function will be an option given to the user to pretty much exercise or train this chosen character which will increase happiness, hungerness, and sleepyness by 10, and decrease boredness by 10. This function will also print a unique message telling the user they gym the character and how much the stats fluctuated.

8. void gameRangeLimitation()

   //gameRangeLimitation function will be an included function that will ensure each of the protected members never reaches less than 0 or greater than 100. If they do end up reaching that number based on the player's actions, then the number will instantly become 0 if less than 0 and 100 if greater than 100.

9. void nextHour()

   //nextHour function will be an included function for this class where it would print a warning message to the player telling them the stats of certain protected members are too low. It will print a unique message when happiness is less than 15, hungerness is greater than 80, sleepyness is greater than 80, and boredness is greater than 75.

GROUND INHERITED CLASS UML DESIGN DESCRIPTION:

• This inherited class consists of 10 public members that consist of overloaded constructors and various overriding functions that are functions of the game where players have the ability to select from the menu which particular function they want to access and some of those functions would make adjustments to the character's stats while some may print an alert message when stats are too low or high, and even just save or load the stats.

• The Overloaded Constructor:

2. Ground(string name)

#During the start of the game, the user will be given an input statement to enter the name of their selected class Pokémon. Once they enter the name, the name they enter is assigned to the variable name in the string name, so this overloaded constructor would be implemented in the Ground.cpp file which will assign the protected member petName to name.

• The Overriding Functions:

10. void saveTheData()

//saveTheData function will print out the saved data of the game if the player decides to save their stats rather than continuing to play with other options. This function will autosave the player's progress and will then break out of the game.

11. void loadTheData()

    //loadTheData function will load up the data of the stats of this Pokémon class selected by the user in order to start playing and modifying the stats.

12. void feedTheAnimal()

//feedTheAnimal function will be an option given to the user to feed this chosen character which will increase happiness by 10, increase sleepyness by 10, decrease hungerness by 10, and decrease boredness by 5. This function will also print a unique message telling the user they fed and how much the stats fluctuated.

13. void treatTheAnimal()

    //treatTheAnimal function will be an option given to the user to give this chosen character a treat which will increase happiness by 10, increase sleepyness by 10, decrease hungerness by 5, and decrease boredness by 10. This function will also print a unique message telling the user they treated the character and how much the stats fluctuated.

14. void playWithTheAnimal()

    //playWithTheAnimal function will be an option given to the user to play with this chosen character which will increase happiness by 15, increase hungerness by 15, decrease hungerness and boredness by 10. This function will also print a unique message telling the user they played with the character and how much the stats fluctuated.

15. void restTheAnimal()

    //restTheAnimal function will be an option given to the user to rest this chosen character which will increase happiness by 5, increase hungerness by 10, decrease sleepyness and boredness by 15. This function will also print a unique message telling the user they rested the character and how much the stats fluctuated.

16. void gymTheAnimal()

    //gymTheAnimal function will be an option given to the user to pretty much exercise or train this chosen character which will increase happiness, hungerness, and sleepyness by 10, and decrease boredness by 10. This function will also print a unique message telling the user they gym the character and how much the stats fluctuated.

17. void gameRangeLimitation()

//gameRangeLimitation function will be an included function that will ensure each of the protected members never reaches less than 0 or greater than 100. If they do end up reaching that number based on the player's actions, then the number will instantly become 0 if less than 0 and 100 if greater than 100.

18. void nextHour()

//nextHour function will be an included function for this class where it would print a warning message to the player telling them the stats if a certain protected members is too low. It will print a unique message when happiness is less than 20, hungerness is greater than 80, sleepyness is greater than 80, and boredness is greater than 75.

ICE INHERITED CLASS UML DESIGN DESCRIPTION:

• This inherited class consists of 10 public members that consist of overloaded constructors and various overriding functions that are functions of the game where players have the ability to select from the menu which particular function they want to access and some of those functions would make adjustments to the character's stats while some may print an alert message when stats are too low or high, and even just save or load the stats.

• The Overloaded Constructor:

3. Ice(string name)

#During the start of the game, the user will be given an input statement to enter the name of their selected class Pokémon. Once they enter the name, the name they enter is assigned to the variable name in the string name, so this overloaded constructor would be implemented in the Ice.cpp file which will assign the protected member petName to name.

• The Overriding Functions:

19. void saveTheData()

//saveTheData function will print out the saved data of the game if the player decides to save their stats than continuing to play with other options. This function will autosave the player's progress and will then break out of the game.

20. void loadTheData()

    //loadTheData function will load up the data of the stats of this Pokémon class selected by the user in order to start playing and modifying the stats.

21. void feedTheAnimal()

    //feedTheAnimal function will be an option given to the user which will increase happiness by 10, increase sleepyness by 10, decrease hungerness by 15, and decrease boredness by 10. This function will also print a unique message telling the user they fed and how much the stats fluctuated.

22. void treatTheAnimal()

    //treatTheAnimal function will be an option given to the user to give this chosen character a treat which will increase happiness by 10, increase sleepyness by 10, decrease hungerness by 5, and decrease boredness by 10. This function will also print a unique message telling the user they treated the character and how much the stats fluctuated.

23. void playWithTheAnimal()

    //playWithTheAnimal function will be an option given to the user to play with this chosen character which will increase happiness by 15, increase hungerness by 15, decrease hungerness and boredness by 10. This function will also print a unique message telling the user they played with the character and how much the stats fluctuated.

24. void restTheAnimal()

//restTheAnimal function will be an option given to the user to rest this chosen character which will increase happiness by 5, increase hungerness by 10, decrease sleepyness and boredness by 15. This function will also print a unique message telling the user they rested the character and how much the stats fluctuated.

25. void gymTheAnimal()

    //gymTheAnimal function will be an option given to the user to pretty much exercise or train this chosen character which will increase happiness, hungerness, and sleepyness by 10, and decrease boredness by 10. This function will also print a unique message telling the user they gym the character and how much the stats fluctuated.

26. void gameRangeLimitation()

    //gameRangeLimitation function will be an included function that will ensure each of the protected members never reaches less than 0 or greater than 100. If they do end up reaching that number based on the player's actions, then the number will instantly become 0 if less than 0 and 100 if greater than 100.

27. void nextHour()

    //nextHour function will be an included function for this class where it would print a warning message to the player telling them the stats for certain protected members are too low. It will print a unique message when happiness is less than 10, hungerness is greater than 80, sleepyness is greater than 70, and boredness is greater than 75.

PROJECT USE-CASES DESCRIPTION:

Use Case I: Loading Game Progress Saved –

During the start of the game, the user will have the option to choose between loading game progress already saved or starting a new game. In the aspect of loading, it will take input of what's stored inside the saveTheData.txt file.

Use Case II: Player Chooses Pichu –

During the game, a menu will be displayed giving the player an option to first pick which character they would like to select. If they decide on picking Pichu as an option, which would be the first option where they would input the number 1 and will then be prompted to choose the 6-member functions that can allow them to modify Pichu's protected members of type int or save and break out of the game. These functions are the following:

1. saveTheData()

// If the player chooses this function, then the data of the game will be outputted into the saveTheData.txt file immediately. This function will autosave the player's progress and will then break out of the game.

2. feedTheAnimal()

// If the player chooses this function, then the following will happen to Pichu: Increase happiness by 5, increase sleepyness by 5, decrease hungerness by 10, and decrease boredness by 5.

3. treatTheAnimal()

// If the player chooses this function, then the following will happen to Pichu: Increase happiness by 10, increase sleepyness by 10, decrease hungerness by 5, and decrease boredness by 10.

4. playWithTheAnimal()

// If the player chooses this function, then the following will happen to Pichu: Increase happiness by 15, increase hungerness by 15, decrease hungerness and boredness by 10.

5. restTheAnimal()

// If the player chooses this function, then the following will happen to Pichu: Increase happiness by 5, increase hungerness by 10, decrease sleepyness and boredness by 15.

6. gymTheAnimal()

// If the player chooses this function, then the following will happen to Pichu: Increase happiness, hungerness, and sleepyness by 10, and decrease boredness by 10.

Use Case III: Player Chooses Sandshrew(Ground) –

During the game, a menu will be displayed giving the player an option to first pick which character they would like to select. If they decide on picking Sandshrew(Ground) as an option, which would be the second option where they would input the number 2 and will then be prompted to choose the 6-member functions that can allow them to modify Sandshrew(Ground) protected members of type int or save and break out of the game. These functions are the following:

1. saveTheData()

// If the player chooses this function, then the data of the game will be outputted into the saveTheData.txt file immediately. This function will autosave the player's progress and will then break out of the game.

2. feedTheAnimal()

// If the player chooses this function, then the following will happen to Sandshrew(Ground): Increase happiness by 10, increase sleepyness by 10, decrease hungerness by 10, and decrease boredness by 5.

3. treatTheAnimal()

// If the player chooses this function, then the following will happen to Sandshrew(Ground): Increase happiness by 10, increase sleepyness by 10, decrease hungerness by 5, and decrease boredness by 10.

4. playWithTheAnimal()

// If the player chooses this function, then the following will happen to Sandshrew(Ground): Increase happiness by 15, increase hungerness by 15, decrease hungerness and boredness by 10.

5. restTheAnimal()

// If the player chooses this function, then the following will happen to Sandshrew(Ground): Increase happiness by 5, increase hungerness by 10, decrease sleepyness and boredness by 15.

6. gymTheAnimal()

// If the player chooses this function, then the following will happen to Sandshrew(Ground): Increase happiness, hungerness, and sleepyness by 10, and decrease boredness by 10.

Use Case IV: Player Chooses Sandshrew(Ice) –

During the game, a menu will be displayed giving the player an option to first pick which character they would like to select. If they decide on picking Sandshrew(Ice) as an option, which would be the second option where they would input the number 3 and will then be prompted to choose the 6-member functions that can allow them to modify Sandshrew(Ice) protected members of type int or save and break out of the game. These functions are the following:

1. saveTheData()

// If the player chooses this function, then the data of the game will be outputted into the saveTheData.txt file immediately. This function will autosave the player's progress and will then break out of the game.

2. feedTheAnimal()

// If the player chooses this function, then the following will happen to Sandshrew(Ice): Increase happiness by 10, increase sleepyness by 10, decrease hungerness by 15, and decrease boredness by 10.

3. treatTheAnimal()

// If the player chooses this function, then the following will happen to Sandshrew(Ice): Increase happiness by 10, increase sleepyness by 10, decrease hungerness by 5, and decrease boredness by 10.

4. playWithTheAnimal()

// If the player chooses this function, then the following will happen to Sandshrew(Ice): Increase happiness by 15, increase hungerness by 15, decrease hungerness and boredness by 10.

5. restTheAnimal()

// If the player chooses this function, then the following will happen to Sandshrew(Ice): Increase happiness by 5, increase hungerness by 10, decrease sleepyness and boredness by 15.

6. gymTheAnimal()

// If the player chooses this function, then the following will happen to Sandshrew(Ice): Increase happiness, hungerness, and sleepyness by 10, and decrease boredness by 10.

PSUEDOCODE

```
cout << Welcome message for the player << endl;

cout << Ask players if they want to start a new game or continue with a saved
file. << endl;

cout << Enter 1 if saved, Enter 2 if new game << endl;

cin << selectOption;

while(selectOption != 1 && selectOption != 2){
      cout << “Incorrect option, please enter 1 or 2” << endl;
      cin >> selectOption;
}


if(selectOption == 1){
      cout << Ask player which class Pokémon they had a saved file with and
would like to continue the game with. << endl;

      cout << “Please type in 1 to select Electric(Pichu) if that was saved”
<< endl;

      cout << “Please type in 2 to select Ground(Sandshrew) if that was
saved” << endl;

      cout << “Please type in 3 to select Ice(Sandshrew) if that was saved”
<< endl;

      cin >> selectPokemon;


      while(selectPokemon != 1 && selectPokemon != 2 && selectPokemon != 3){
            cout << “Incorrect option, please enter 1, 2, or 3” << endl;
            cin >> selectPokemon;
      }
}


else{
      cout << Welcome message of starting a new game << endl;

      cout << picture of Electric from Electric.txt file using ASCII Art <<
endl;
```

```
	cout << picture of Ground from Ground.txt file using ASCII Art << endl;

	cout << picture of Ice from Ice.txt file using ASCII Art << endl;

	cout << Tell player to enter 1-3 for Electric(Pichu),
Ground(Sandshrew), or Ice(Sandshrew) << endl;

	cin >> selectPokemon;


	while(selectPokemon != 1 && selectPokemon != 2 && selectPokemon !=3){
		cout << Incorrect option, please enter 1, 2, or 3 << endl;
		cin >> selectPokemon;
	}

	bool  theTruth = true;

	if(selectPokemon == 1){
		Electric classObj1
		//Make an if statement for loading a saved file.


		if(selectOption == 1){
			classObj1.loadTheData();
		}

		else{

			cout << “Please enter what you would like to name this
		Electric(Pichu) Pokémon” << endl;

			cin >> name;

			Electric classObj1(name);
}



		while(theTruth){
			cout << “Select options 1-6 to modify the Pokémon following
ways” << endl;

			cout << Type in 1 to save and exit from the game << endl;


			cout << Type in 2 to feed the Pokémon << endl;

			cout << Type in 3 to treat the Pokémon << endl;
```

```
      cout << Type in 4 to play with the Pokémon << endl;

      cout << Type in 5 to rest the Pokémon << endl;

      cout << Type in 6 to gym the Pokémon << endl;


      cin >> selectModifications;


      while(selectModifications != 1 && selectModifications !=2
&& selectModifications !=3 && selectModifications != 4 &&
selectModifications != 5 && selectModifications != 6){

            cout << “Incorrect option, please type in 1 to save
and exit, type in 2 to treat the Pokémon, type in 3 to treat the
Pokémon, type in 4 to play with the Pokémon, type in 5 to rest
the Pokémon, or type in 6 to gym the Pokémon << endl;

            cin >> selectModifications;
}



if(selectModifications == 1){
      classObj1.saveTheData();
      cout << “Successfully saved and exited game << endl;
      break;
}

else if(selectModifications == 2){
      classObj1.feedTheAnimal();
      classObj1.gameRangeLimitation();
      classObj1.nextHour();
}

else if(selectModifications == 3){
      classObj1.treatTheAnimal();
      classObj1.gameRangeLimitation();
      classObj1.nextHour();
}

else if(selectModifications == 4){
      classObj1.playWithTheAnimal();
      classObj1.gameRangeLimitation();
      classObj1.nextHour();
}

else if(selectModifications == 5){
      classObj1.restTheAnimal();
```

```
                classObj1.gameRangeLimitation();
                classObj1.nextHour();
        }

        else if(selectModifications == 6){
                classObj1.gymTheAnimal();
                classObj1.gameRangeLimitation();
                classObj1.nextHour();
        }




    if(selectPokemon == 2){
        Ground classObj2;
        //Make an if statement for loading a saved file.


        if(selectOption == 1){
                classObj2.loadTheData();
        }

        else{
                cout << “Please enter what you would like to name this
Electric Pokémon” << endl;

                cin >> name;

                Ground classObj2(name);
}


        while(theTruth){
                cout << “Select options 1-6 to modify the Pokémon following
ways” << endl;

                cout << Type in 1 to save and exit from the game << endl;


                cout << Type in 2 to feed the Pokémon << endl;

                cout << Type in 3 to treat the Pokémon << endl;

                cout << Type in 4 to play with the Pokémon << endl;

                cout << Type in 5 to rest the Pokémon << endl;

                cout << Type in 6 to gym the Pokémon << endl;
```

```
      cin >> selectModifications;


      while(selectModifications != 1 && selectModifications !=2
&& selectModifications !=3 && selectModifications != 4 &&
selectModifications != 5 && selectModifications != 6){

            cout << “Incorrect option, please type in 1 to save
and exit, type in 2 to treat the Pokémon, type in 3 to treat the
Pokémon, type in 4 to play with the Pokémon, type in 5 to rest
the Pokémon, or type in 6 to gym the Pokémon << endl;

            cin >> selectModifications;
}



if(selectModifications == 1){
      classObj2.saveTheData();
      cout << “Successfully saved and exited game << endl;
      break;
}

else if(selectModifications == 2){
      classObj2.feedTheAnimal();
      classObj2.gameRangeLimitation();
      classObj2.nextHour();
}

else if(selectModifications == 3){
      classObj2.treatTheAnimal();
      classObj2.gameRangeLimitation();
      classObj2.nextHour();
}

else if(selectModifications == 4){
      classObj2.playWithTheAnimal();
      classObj2.gameRangeLimitation();
      classObj2.nextHour();
}

else if(selectModifications == 5){
      classObj2.restTheAnimal();
      classObj2.gameRangeLimitation();
      classObj2.nextHour();
}

else if(selectModifications == 6){
      classObj2.gymTheAnimal();
```

```
                classObj2.gameRangeLimitation();
                classObj2.nextHour();
          }



     if(selectPokemon == 3){
          Ice classObj3;
          //Make an if statement for loading a saved file.


          if(selectOption == 1){
                classObj3.loadTheData();
          }

          else{
                cout << “Please enter what you would like to name this
Electric Pokemon” << endl;

                cin >> name;

                Ice classObj3(name);
}


          while(theTruth){
                cout << “Select options 1-6 to modify the Pokémon following
ways” << endl;

                cout << Type in 1 to save and exit from the game << endl;


                cout << Type in 2 to feed the Pokémon << endl;

                cout << Type in 3 to treat the Pokémon << endl;

                cout << Type in 4 to play with the Pokémon << endl;

                cout << Type in 5 to rest the Pokémon << endl;

                cout << Type in 6 to gym the Pokémon << endl;


                cin >> selectModifications;


                while(selectModifications != 1 && selectModifications !=2
          && selectModifications !=3 && selectModifications != 4 &&
          selectModifications != 5 && selectModifications != 6){
```

```
                cout << “Incorrect option, please type in 1 to save
            and exit, type in 2 to treat the Pokémon, type in 3 to treat the
            Pokémon, type in 4 to play with the Pokémon, type in 5 to rest
            the Pokémon, or type in 6 to gym the Pokémon << endl;

                cin >> selectModifications;
            }



            if(selectModifications == 1){
                  classObj3.saveTheData();
                  cout << “Successfully saved and exited game << endl;
                  break;
            }

            else if(selectModifications == 2){
                  classObj3.feedTheAnimal();
                  classObj3.gameRangeLimitation();
                  classObj3.nextHour();
            }

            else if(selectModifications == 3){
                  classObj3.treatTheAnimal();
                  classObj3.gameRangeLimitation();
                  classObj3.nextHour();
            }

            else if(selectModifications == 4){
                  classObj3.playWithTheAnimal();
                  classObj3.gameRangeLimitation();
                  classObj3.nextHour();
            }

            else if(selectModifications == 5){
                  classObj3.restTheAnimal();
                  classObj3.gameRangeLimitation();
                  classObj3.nextHour();
            }

            else if(selectModifications == 6){
                  classObj3.gymTheAnimal();
                  classObj3.gameRangeLimitation();
                  classObj3.nextHour();
            }
return 0;

}
```