**Final Report for Course Object Oriented Programming**



**Lecturer:**

**Jude Joseph Lamug Martinez, MCS**

**Made By:**

**Michael Bengawan 2602184290**

**Binus School of Computer Science Undergraduate Program**

**Universitas Bina Nusantara**

**Jakarta**

**2023**

**Table of Content**

Cover1

Table of Contents2

Project Specifications3

Idea3

Creation3

Finalization4

Solution Design6

Class Diagram6

Implementation7

Import7

Class7

Methods8

Evidence13

Useful Resources19

**Project Specification**

1. **Idea**

The idea of the project is to create a Pokémon game with students from Binus International to be the Pokémon and the teachers as the Pokémon trainers. The idea came about when me and Brandon were brain storming ideas in class and ended up giving me idea to make a Pokémon game. After that the idea evolved further after me and Kennan started to discuss our projects and inevitably, I decided to take on his idea of having the hp of the player be emotional damage. This aspect led to the creation of the commend system to encourage their students in a positive way this aspect replaced the abilities of the students. Most of the basic Pokémon game features stay, in my original plan. Example would include the turn system, having multiple students, challenging opponents, gaining experience, having and open world like the FX campus and having a final opponent that being the esteemed Head of Program similar of the elite four. Sprites I wanted to make myself and with the help of a few friends. And the code I would do myself without outside help. And I finalize the name of the game being Binusmon the combination of Binus and Pokémon.

1. **Creation**

Early in the development of binusmon I made a spreadsheet containing all the characters, students, move, and the type chart that I wanted to use in the game.

A screenshot of a computer

Description automatically generated with medium confidence

1.2.1 Original Type Chart

A picture containing text, screenshot, number, parallel

Description automatically generatedA picture containing text, screenshot, number, parallel

Description automatically generated

1.2.2 Student Chart 1.2.3 Move Chart

These were the original type, students, and move that were to be implemented in the game in the final game most of these did not make it as some where to hard to implement or time constraint. At the moment the opponent and player characters were not in the mix as of, yet the only idea was for the Head of Program to be the first and final opponent the player would be facing the idea of all the teacher assistants being the player character were later done. At first, I made a text version of the game as easier way to start of the development using command line to input and output data. This was done without GUI to make it easier for me to understand the methods that have to implemented into the game. This version started to be created on early December when holidays were starting.

A screenshot of a computer

Description automatically generated

* + 1. The Command Line Version

A screen shot of a computer program

Description automatically generated with medium confidence

* + 1. All the Methods in the earlier version

The end of the command line version ended like this with all the methods in image 2.5. The earlier version did not have any type of GUI or End or Lose screen it only had the battle mechanics and all the class for the game. Which included student, moves, player, and opponent. These were all later recycled into the finished game. This version was also the testing grounds for the GUI which at first, I was using java swing a java library used to make GUI’s. But after tinkering around with it for a few days. I realised that using javafx would be more suitable to what I was going for with binusmon. When I was creating the finished game, I was too focused on creating the animation for the game, but I notice that time was already running very short, so I instead focus a lot more into creating the finished game for the last week of the project. Most of the game just using event handlers and using the same concept from my python project to make the open world.

1. **Finalization**

After finishing all the necessary features that are needed to make the game certain flaws are there with a lot of primary features left out of the finished project including,

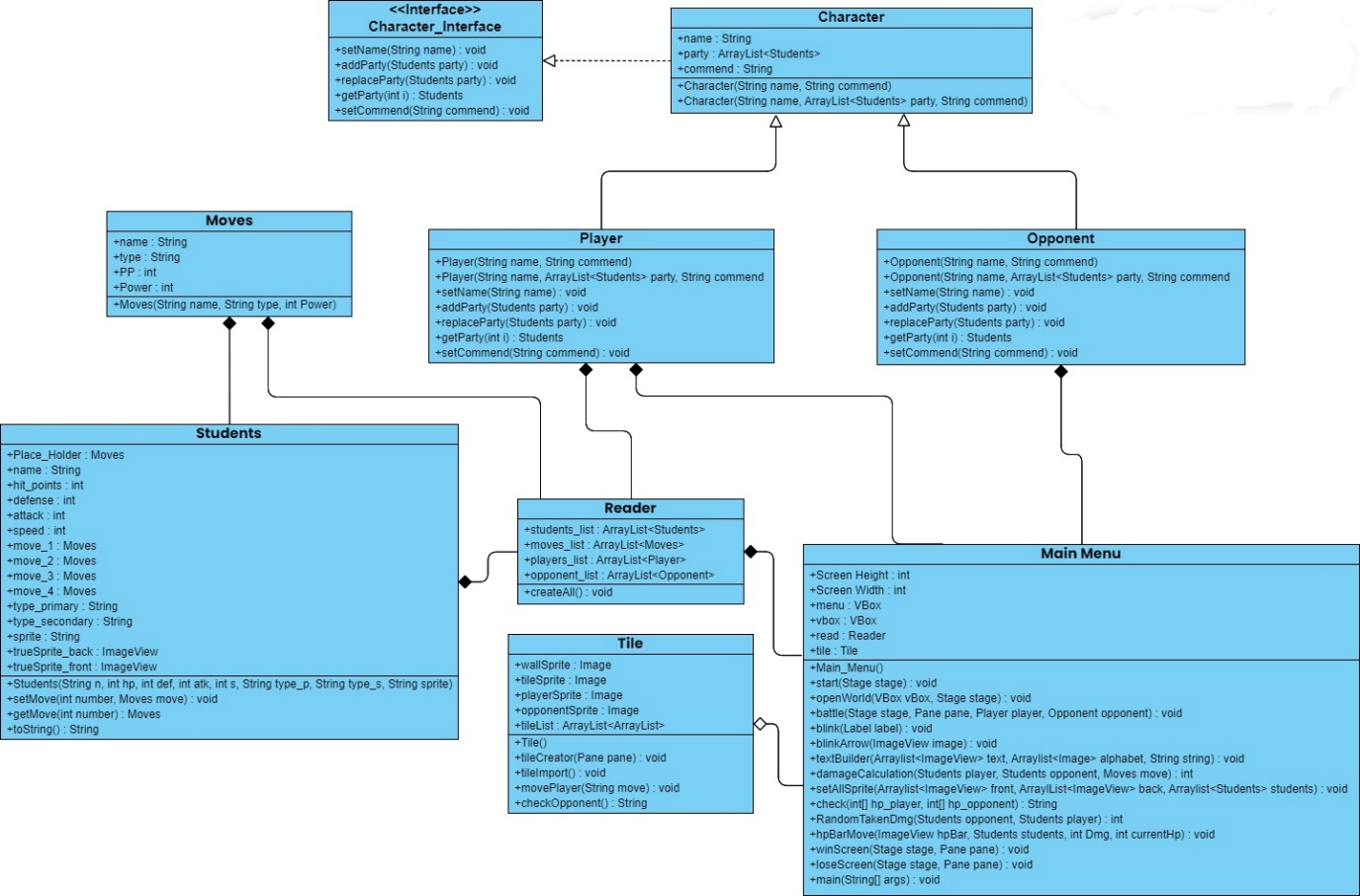
1. Opponent Commend
2. Save Feature
3. New Game Feature
4. Sounds and Music
5. Speed Mechanic
6. Student Dex

These were the multitude of things that I could add in time for the finished project due to time constraints while some were only extra features like music or new game. Others were primary mechanics that I had to cut from the game due to limited time. But the game did have,

1. Working Menu
2. Overworld
3. Collision Detection
4. Working GUI
5. Text Reader
6. Damage Calculation
7. Damage Taken Calculation
8. Working Battle System
9. Working Hp Bar
10. Working Lose and End Screen
11. Working Move System
12. Working Buffs
13. Working Player Commend
14. Two Working Opponents to Battle

**Solution Diagram**

1. **Class Diagram**



2.1.1 Full Class Diagram

The class diagram consist of the Character which is at the top of the diagram as the Player and Opponent class inherent from the Character class as all the basic functions of the class are within the Character class. The Character Interface is implemented to the Character to simplify all the methods in the Character class. The Moves class is required to run the Students class as the students depend on the Moves class the Reader class depends on all the Moves, Students, Player, and Opponent class as the Reader read the txt data to be able to put them in the correct Array List. After that the tile class can run on its own, but the Main Menu class requires all the class to run.

**Implementation**

1. **Import**
2. JavaFX

JavaFX is a GUI builder for the java language that is built to create applications in java. JavaFX is used over java swing due to JavaFX having animation and other useful methods like CSS like stylings. But within the application itself there is no use of the FXML within the GUI. This is due to it being simpler to use it without the use of FXML or CSS. Instead, the JavaFX that build the GUI only uses inline code from java only. The use of JavaFX is the scene builder, panes and other parent nodes, timeline, and animations from pane to pane.

1. JavaUtil

JavaUtil or Java Utility is a package containing all the utility classes in Java. Out of the many packages that are available to use the only ones used in the project are, List, Array List, and Random. Array List is used the most cause all the data is mostly stored in Array List. While other methods are possible like using a normal array the flexibility an Array List adds is very much useful in this situation. List are used only for creating sub-List within the Array List. And the Random utility is only used to randomize the value within damage calculation.

1. JavaIO

JavaIO or Java I/O is a java utility package that is used for I (input) and O (output). JavaIO used in the project is BufferedReader, File, FileReader, IOException, FileInputStream, and FileNotFoundException. File, FileReader, and BufferedReader are all used to read txt files to be able to import character into the readable forms of text. This is done to create the world map and generate all the characters and moves. FileInputStream is used to generate a stream of bytes that can be intrepid as a image for all the sprites and background images. Both the IOException and FileNotFoundException is to handle if any input or output are not able to be read and if the file is unable to be located it would handle the error message.

1. **Class**
2. Character Implementation

This class is only an interface for the character class. It has all the methods the character class needs. With the methods being the setters and getters.

1. Character

This class contains all the necessary functions the player and opponent needs being the attribute name, array list of party, and the commend. This also includes the two constructor one with name and commend the other having name, party, and commend.

1. Moves

This class has four attributes which are the name, the type, the pp count, and the power. This class only has a constructor which takes the name, type, and power. PP is currently not implemented so it remains as zero.

1. Opponent

This class inherence the character class and implements the character interface. Which means it has all the attributes and methods from the character class and has created all the methods from the implementation.

1. Player

The player class is the exact same as the opponent class with virtually no difference. The only reason their apart from each other is to make differentiating a lot easier in the other methods.

1. Reader

The reader class has four attributes which are four array list of all the Student, Moves, Player, and Opponent classes. This is done to easily categorize them for later use. This class contains the createAll method which is used to create all the characters and move in from a txt file format. So, the primary use of this class is to organize and create all the characters for use in other classes.

1. Students

The students class is a central class as it is required to run most other classes as the game revolves around them, like how the Pokémon games revolve around the Pokémon. The student class has 14 attributes those being name, hit points, defence, attack, speed, move 1, move 2, move 3, move 4, type primary, type secondary, sprite, true sprite back, and true sprite front. The methods in this class are the constructor, and a setter and getter for moves. Other then that there is a to String method for debugging. This class is primary used to just store all the data for the students as most of these are for combat purposes and only the sprites are for the GUI.

1. Tile

The tile class has five attributes being wall Sprite, tile Sprite, player Sprite, opponent Sprite, and the tile list. The methods are the constructor, tile creator, tile import, move player, and check opponent. The tile class itself is focused on the open world with the constructor already importing all the tile list using the tile import method. And it always being called in the open world method of the project. The tile class is pretty much just a tile manager for the game.

1. Main Menu

The main menu class is the primary class in the project, it having the main method and having the most methods in the project. The class has 6 attributes but most of them are for initializing objects to be use later in other methods. These include 2 static integers being screen height and width, two Vboxs, and the reader and tile class. Methods in this class include the start, open world, battle, blink, blink arrow, text builder, damage calculation, set all sprite, check, random taken damage, hp bar move, win and lose screen, and the main method. This class is used for the GUI and running the game in general, the main menu is the glue that sticks all the classes together and runs them in union.

1. **Methods**
2. blink

A very basic method that takes in a label and then puts the label under 2 transitions that are sequential to achieve blinking text. This later then loops indefinitely until the screen ends.

1. blinkArrow

Like blink this puts an image instead of a label the image then is put under the same 2 transition effects that are sequential to achieve blinking. Again, this is put in an indefinite loop.

1. textBuilder

This method takes two Array List of image view and image and a string. Using a for loop for as long as the string’s length to put all the characters in the current array list of image views. How this works is the array list having all the image views of a specific text in the GUI this then gets replace by the image array list based on the current string index in the for loop. To generate an array list of image view that is based on the string that was given. After this it then checks if the string inputted was “Non” and if it is it would use the 27 character in the alphabet array list to make it a blank text view.

1. damageCalculation

To calculate damage this method, take two Students the attacker and the defender, the move being used, and the buffs the two students are experiencing at the time. This either method takes two parts the first being the actual damage integer value first. Which is based on the student attacking attack stat and all the buffs that are gained to become attack, and then defence is calculated by student being attack’s defence plus any buffs that there is divided by two. This is done to ensure defence does not become to overpower. After that it gets the move’s power which if zero will check its move name to then assign a random damage value to certain moves being used. And then the damage formula goes as,

This then gets the move damage which is used to reduce the opponent’s hit points. After this the second part of the method starts as it then takes the damage and puts it through a switch case to compare the move type with the opponent’s primary type and secondary type to get the damage multipliers for each type refer to image 1.2.1 to get the full type chart. After this it then returns the resulting damage.

1. setAllSprite

This method takes the array list of front and back sprites for the students. And it takes all the students sprite value and assigns a front and back sprite for all the students. The two sprites are for the player view and the opponent view.

1. check

This method takes the hp player list and the hp opponent list and checks if any of them are all zero or lower of value. If the opponent is all zero it returns the string “Win” which results in the player winning. If the player is all zero, then it returns the string “Lose”. Which result in the player losing. If neither are true, then the string “Continue” will be returned.

1. RandomTakenDmg

This method takes all the same values as the damage calculation method the only difference is that this is used for the opponent so there is a random change between 1 till 4 so it can choose to use any one of those 4 moves. It then uses damage calculation to return the damage taken integer value.

1. hpBarMove

This method takes the image view of the hp bar, the student that hp is being moved, the damage taken, and the current hp of the student. It then calculates the full hp of the student based on the 113-pixel size of the image view of the hp bar to then calculate how much to move the hp bar by how much. This is done by using the translate transition available in JavaFX.

1. winScreen / loseScreen

Both methods achieve the same thing being they both display an image and take you back into the overworld to rechallenge or move to fight a different opponent in the overworld. This method takes the current stage and the current pane, the current pane then is faded out with the transition effect and then it creates a new Vbox to be fade in with the background image of the win or lose screen. It then sets a key event if the space bar is pressed it will then take it back into the overworld. Doing another fade transition back into the open world.

1. createAll

This method is part of the reader class which is used to read the txt file to set all the characters and moves in the game. The create all method takes nothing and when started creates a new file that takes in the current txt file that is being used. And then puts in into a file reader to read the contents and a buffered reader to ensure that it is read one by one. This is all then put into an array list full of all the strings based on the txt file. The method then puts in value based on the position of the string “Player”, “Opponent”, “Students”, and “Moves” to get all the starting values and ending values of all the strings in the array list. This is then moved into four separate list that contain each of the four strings. This is then later all created into all their respective objects. For the player and opponent, they have the same method that is creating two strings and getting the full string from the list then making four integer value for all the points and a check value. Using all four integers the first point is to get the start of the name, second points is the end of the name, the third point is the start of the commend name and the last point is the end of the string. The check value is to check whether the first value is obtained. This is because the method finds the points with the “:” char so without the check the first and third point would be the same point with the check it would skip the first “:” char. This is then combined to get the string name and commend to create the player or opponent object which is later added into the player list and the opponent list.

The student and the move part of the method uses a different method, this method involves first having the base length of the string without the name. Since the only major different between student and move is the name which can be of any size this results in it being possible to calculate the exact position of the name with the current string length. All the other values are gaining the exact same way with checking the current name length using the string length to gain the exact position of hit point, defence, speed, attack, type primary, type secondary, and the sprite of the student. This is done the same way in the move part of the method to gain the exact position of name, type, and power which is used to create a moves object. Both are put into an array list same as the player and opponent objects that are to be used later.

1. tileImport

This method is used to import the tile list that is used to create the overworld. This is done by again using the file reader and buffered file to read the txt file line by line. This is then used to put all the values in x and y positions. This is done by check if the char at value of the string is a anything other then a “ “ (space) it then puts in into the x array list and if its finished reading that part of the txt file it puts then x array list into the y array list and then repeats this process until the entire tile list is finished.

1. tileCreator

Tile creator updates the open world map every time it is called. This method takes the current pane for it to draw on and creates a 15 by 20 grid of image views that area 50 by 50 pixels long. This is then added into the full view array list which is the entire array list of the image views. The method then finds the players position within the tile list and takes its x and y values. The method then takes the player x and y values and -7 for the x value and -9 for the y value. This is done so the player will always be drawn on the centre of the screen. It then checks the tile list for all the values and assigns the image to each image view to paint the full pane.

1. movePlayer

This method takes a string value of the position the player is supposed to move to. This method first gets the players exact position within the tile list, it then checks whether the player moves “Up”, “Down”, “Right”, or “Left” and then based on the string inputted it would change the tile list based on if the tile the player wants to move to is a “o” tile which is an empty space. If this condition is true, the player will replace the “o” tile and the old “p” tile would be replace by an “o” tile. If the tile the player wants to move too is a tile other than “o” the player will fail to move, and the tile list remains unchanged.

1. checkOpponent

The check opponent method is used to check if the player has an opponent tile above, below, or next to them. This method does so by getting the player position and then checking all the values if the value matches with any opponent tile, then it returns a string with their opponent tile list value. If non comes up, then the value stays as “Non”.

1. Start

The second longest method in the project start acts as the starting menu for the game. The press any key to continue screen is a simple screen with one label that uses the blink method to blink the text. The key event handles the transition and later it changes into the title screen which shows the title, and all the options available. Most of this is done with basic JavaFX methods the base menu itself being based on the Vbox parent node. That is used with 1 label and 4 buttons. Each button has padding to ensure that they are in the correct position. To achieve the on-hover effect it uses a mouse event when enter it reduces the padding to equal the change in font size. And when mouse exit the button, it reverts the changes creating a seamless transition effect. Pressing the buttons requires another mouse event this time press and it open the open world. The other buttons was not implemented and only exit works.

1. openWorld

This method first take the Vbox from the start method and uses it to create a transition effect to fade out the menu then creates another transition effect with the timeline to time it correctly. After doing so it then uses the key events to set up the world itself. First, it uses the tileCreator to generate the open world first based on the imported txt. After this it then uses the key events to check if any keys are pressed at the current screen. If a key is pressed it uses the movePlayer method to move the player based on the key pressed and then generates the new update world using the tileCreator method again. If the player presses the space button it then runs the checkOpponent method and then checks if it comes up with any opponent tile it would innate the battle method to start a battle with the opponent in question.

1. battle

The battle methods takes the current stage, current pane, the player and the opponent. It then makes the pane and name it battle. Then it starts by first fading out the current pane and fading in the battle pane with sequential transition. The transition this time is two fades to create the screen flash effect. And then it creates another 4 blocks using JavaFX shapes. These shapes are place in negative values and higher values of the screen and then transition into the pane using the translate transition. Before doing all of this the pane has already imported all the necessary GUI components using file input stream and put them into image view and then added them into the battle pane according to the layers they need to be in. And then after it has done all of that another translate block occur this time putting them back into their original position but before that it sets all the values of the image views to true to reveal the GUI after it has finished loading all of them creating a seamless transition into the battle menu. While doing this it also sets up two other things which are the setAllSprites method and creates the alphabet list for the text image view later. And then it finally starts the battle function it first sets up the current arrow which the position the arrow is currently in, the current status buff for the player and the opponent, sets all the arrows to blink, creates the player hp, the opponent hp, the player count, the opponent count, and sets up a for loop to put in all the player hp and opponent hp. And then it creates the current student list and the status of the battle which on base it is on main. It then after 5 seconds will start the battle setting both the players and opponents current sprites to true and starts the key events. Then the player is able to move in up, down, left, right motion. To move the current arrow into 1, 2, 3, or 4. And as it does that it changes the current arrows visibility to true or false based on the current arrow. And then by pressing space one is able to choose their action this is based on the current arrow, the current status, and if the player or opponent still has hp left. After checking all those values, it then executes the following code. For the status “Main” it starts by changing the status to either “Fight” to enter the fight page, “Student” to change the student, and the “Commend” to enter the commend page to choose to power your student. All this status will change the current text. To keep the illusion that its seamless between the GUI text and the PNG text, and to not overlap between each other I use a white cover which just a PNG of a white box that covers the text to make it not overlap and then use the text builder to generate the text based on the status. To make it nicer in general to handle pressing escape will return the player back into the main status. Now, for the fight status the choice then becomes the current moves the current student has. And choosing the option by pressing space will take up a turn and damage calculation will then take into effect as well as damage taken which will then reduce the player and opponent health. After dealing both damages it will then check if the opponent has died and if so, it will replace them and if not, it will keep the opponents current student. After that it will then use the check method to check if the player or opponent has lost. If neither then the battle continues as it sets the status back to main and repeats the process. For the student status it will change the text to be all the current student names and allow the player to change students. For the commend option it shows the players commend ability and allows them to use it. Using the student or commend option will not consume a turn only choosing the battle option consumes a turn. The forfeit status current does not work. But once the check function has determined a player or opponent have lost then it starts the win screen or lose screen method.

**Evidence**

**A screenshot of a computer

Description automatically generated with medium confidence**

**A picture containing indoor, furniture, desk, chair

Description automatically generated**

**A picture containing screenshot, text, diagram, symbol

Description automatically generatedA screen shot of a computer

Description automatically generated with low confidence**

**A picture containing screenshot, symbol, diagram, graphics

Description automatically generated**

**A screenshot of a video game

Description automatically generated**

**A screenshot of a video game

Description automatically generatedA screenshot of a video game

Description automatically generated**

**A screenshot of a video game

Description automatically generatedA screenshot of a video game

Description automatically generatedA group of people in a classroom

Description automatically generated with medium confidenceA soldier in uniform standing next to a coffin

Description automatically generated with low confidence**

**Useful Resources**

Sprite sheet:

<https://www.spriters-resource.com/game_boy_advance/pokemonfireredleafgreen/>

Java Help:

<https://docs.oracle.com/en/java/javase/20/>

<https://www.javatpoint.com/>

<https://www.geeksforgeeks.org/>