# Chapter I - Introduction

For the last project of Object-Oriented Programming class, we have produced a project which is using basic and kind of advantaged of OOP knowledge and design pattern like Singleton, Publish-Subscribe.

Java is an only programming language we have used in this project, a representative language in OOP. Besides that, we have used Swing, Socket for our display and networking.

In this report, we will give a detail description on our project. The structure of our report is outlined as follows:

Chapter I: Introduction.

Chapter II: Rules and gameplay.

Chapter III: The details of Game technique.

Chapter IV: UML class diagram.

Chapter V: Evaluation.

# Chapter II - Rules and gameplay

* 1. Gameplay

This is a type-based 1v1 street fighting game in which players will have to type a combination of keys in order to use skills. Each skill has its own graphics, element type, and a set of combination of keys.



**

**

*Figure 1 - Several scenes of game*

On each 5ms, the game frame will be reset and received, rendered the key which is pressed by player or taking next position of the existing skills.

****The game will end when one of the players being taken down (out of blood).

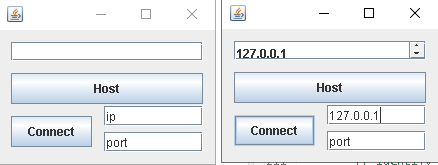
*Figure 2. Ending of the game*

* 1. Networking

Our game just only taken one game per time. If you want play again, just reopen it and play it again.

Beside the main game, we product the networking system that connect two devices in one Lan network, one is host, and one is client.

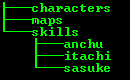
So that, two players can play through the Internet.

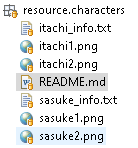
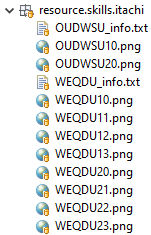


*Figure 3. Game connecting*

# Chapter III – The detail of Game technique

* 1. Main game tree
     1. Introduction about function of used class:
* Controllers:
* class Main():
  + Implement ActionListener class:
  + By implementing ActionListener, we can use many offered functions like:
    - actionPerformed() (doing when the timer call the class per 5ms).
    - KeyListener() (get the keyboard action in order to take the combination skill).
  + There are two variables which Main() received from ServerUI in order to determine the information of player that is: id (id of player) and character (character chosen by player)
* class RenderManager():
  + Render background, player and existing skill. Then, adding to jframe in order to display it.
* class PlayerManager():
  + Saving the information of two player, HP, mana, shield, and status of player.
* class SkillManager():
  + Saving the existing skill of both player.
  + Detecting when skill hit the opponent player or hit skill of the opponent player.
* class NetworkManager():
  + aaa
* Player:
* Player():
  + Implement Serializable(): serialize the data, in order to transport the data.
  + Received name of character and its id.
  + Get information of that character.
  + Call SkillRender() in order to generate own skills.
* PlayerRender():
  + Extends JPanel(): render the image, hp, mana, shield.
  + Calling only by RenderManager() each frame in order to update display information.
* Skill:
* SkillRender():
  + Check the legalization skill.
  + Save information of each unit of skill.
  + Update the next position of each skill.
    1. Flow of main game
* First, there is a singleton class Main() in which cover all of the window, frame, action… of the game.
* Second, the Main() call init() in order to:
  + Define the jframe.
  + Generate background.
  + Generate player.
* Next
  + Each frame will listening for the KeyListener() in order to get key from keyboard, saving it.
  + If ‘Space’ is pressed, Player will check the number of combination keys. Then, call the SkillRender() class to checking and generating that skill.
  + If the skill is generated, the skill will be append into SkillManager(). Therefore, we can re-calling and re-painting each frame.
  1. Main server tree
  2. Resources
* The resource of game will be held in ./src/resource folder.



* Include:
* Characters:
  + <character\_info.txt>: save information of character, structure in README.md file.
  + Two images of character, one for the left side, and one for the right side.
* Maps:
  + Image of background.
  + Image of ending scene.
* Skills
  + The folder ‘anchu’ contain the mini skill.
  + Putting in the folder named like the character own that skill.
  + The name of the skill file is the corrected combination of that skill.
  + <key\_info.txt> contain the information about the skill. Structure of info file is in ./src/resource/skills/README.md
  + After the combination is two digits. First digit is the side of the owner. Second digit is the order of animation.
* Once you want to expand or improve resource of this game, just following the structure and add whatever skill you want.

# Chapter IV - UML Class Diagram

Diagram

Description automatically generated

Diagram

Description automatically generated

*Figure 4. Diagram of the daRIEUnaruto*

* There are two part corresponding with our tasks for this project.
* First for game play, the other for game server.

# Chapter V - Evaluation

* This project is designed and built for the simplest and flexible way that can expand and freely.
* Follow the structure of player, you can add any character.
* Like player, with the combination of 12 keys, you can create and improve a lot of skills for just one character.
* Because the server and core game play is designed and built independently, you can replace with advantage server, or the stronger core game play.