



BINUS UNIVERSITY
BINUS INTERNATIONAL

Assignment Cover Letter
(Individual Work)

Student Information:

Surname: Wardana **Given Name:** Sandrian **Student ID Number:** 2502016411

Course Code : COMP6699001 **Course Name :** Object Oriented Programming

Class : L2CC **Lecturer :** Jude Joseph Lamug Martinez

Type of Assignments: Final Project

Submission Pattern

Due Date : 20 June 2022 **Submission Date :** 21 June 2022

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer's instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

Plagiarism/Cheating

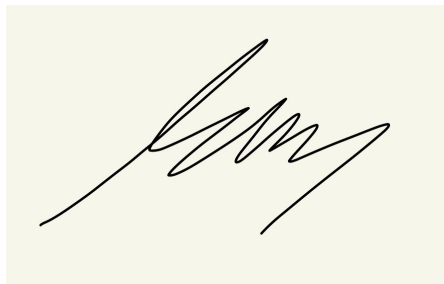
BiNus International seriously regards all forms of plagiarism, cheating, and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity, and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

Declaration of Originality

By signing this assignment, I understand, accept, and consent to BiNus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student:

Sandrian Wardana

A handwritten signature in black ink on a light yellow background. The signature is stylized, starting with a long horizontal stroke followed by several loops and a final upward stroke.

Dank Souls: Text Based RPG

Sandrian Wardana - 2502016411

CONTENTS:

Program Description	5
Class Diagram	6
Program Flow	7
Start Screen:	7
Asks for Name:	7
Show Story Part (Outros):	7
Choose Trait:	8
Show Story Part (Intros):	8
Main Menu:	9
Choosing Resume Journey:	9
Battle:	9
Asked To Take A Rest:	12
Choosing View Character Info:	12
The Final Battle (Act 4):	13
Ending/Win Screen:	14
Death Screen:	14
Code Explanation	15
Program Structure:	15
Main Class:	15
GameLogic class:	16
abstract Character class:	18
Player class:	18
Enemy Class:	19
Story Class	20
Lessons Learned	21
Project Github Link	21
References	21

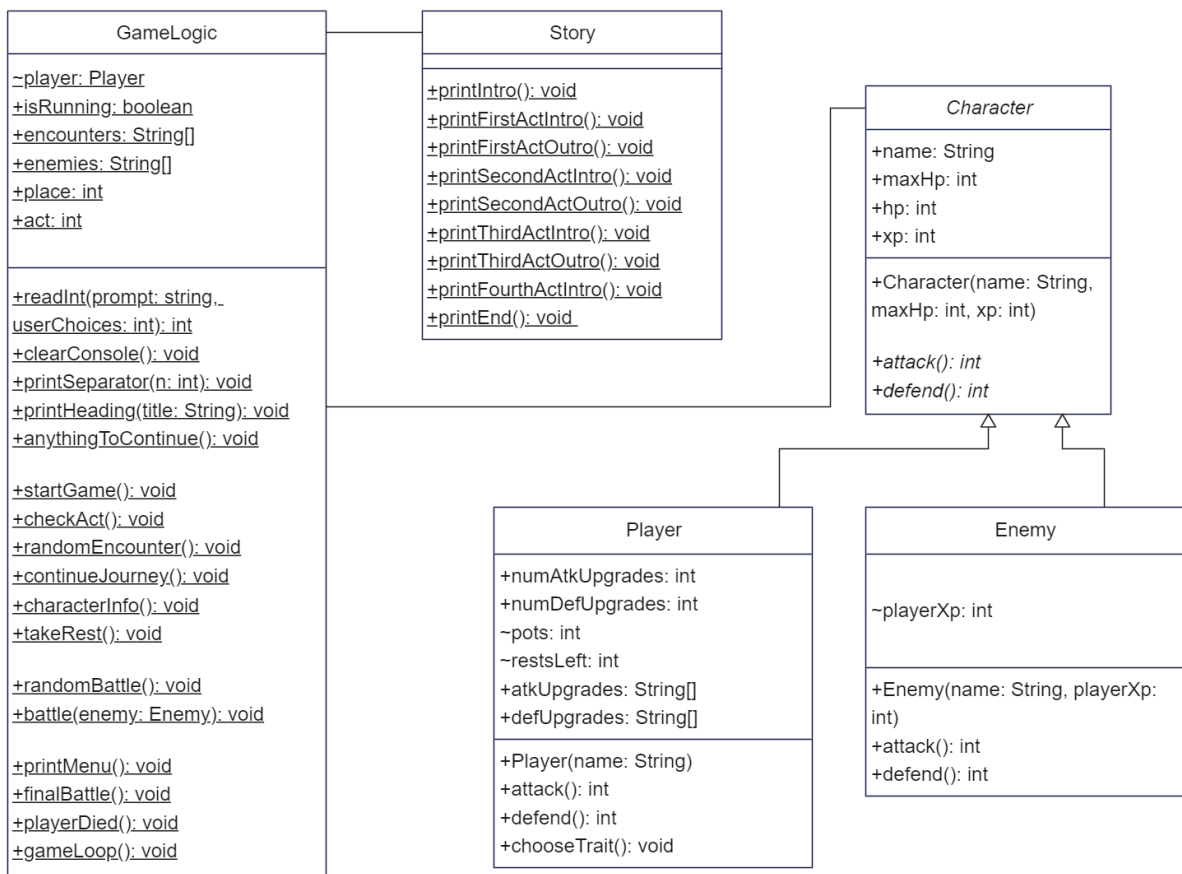
1. Program Description

This is a text-based Role-Playing Game. You play in the console and choose different options of action for you to progress the story of this game. In order to progress the game you must gain XP points throughout your journey by defeating enemies. There are 4 different acts of the story in which they require a certain amount of XP for you to have to pass each one of them. In the end, you'll be encountering the final boss that you need to defeat in order to win the game.

For this project, I thought to make something that would be fun in the end so I picked to create a game. A text-based RPG was something that came to my mind when thinking of a game to implement.

2. Class Diagram

- The GameLogic class: This class controls the game's settings and how the game runs until the end of the game.
- The Story class: This small class only handles the methods that display the story as the player progresses through the game. It uses methods created from GameLogic.
- The **abstract** Character class: The parent class that outlines the attributes and methods of the children class of Player and Enemy.
- The Player and Enemy class: The children class that extends the Character parent class.



3. Program Flow

A. Start Screen:

The start of the game will look like this:

```
-----  
-----  
Dank Souls: Text-Based Java RPG  
by Sandrian (credit to CodeStudent for inspiration)  
-----  
-----  
  
Enter anything to continue.
```

B. Asks for Name:

You will have the option to make sure if you chose your name correctly or not after making a name. If unsure, it will ask you again.

```
-----  
-----  
I  
-----  
State your name, Tarnished.  
-----
```

GAME LOOP STARTS FROM HERE:

C. Show Story Part (Outros):

Depending on where you are in the game, or in other words, how much XP you have, the story text will change. In part C, at the start of the game, it shows you the intro to the whole story. As you progress, it will change to show you the outro story of each act.

```

-----
The Story
-----
You awaken in a cave. Besides you is a bonfire with a sword stuck into it. You remember that you need to embark on a journey to defeat the Elden Beast. The strongest Beast
You do not know what you are. All you know is that you've been assigned to this journey. For whatever reason, you cannot remember anything else about yourself, but you do
As you walk towards the light leading outside of the tunnel, a hooded figure stands at the edge of the cave entrance. You approach them. Wind gushes upon and the hooded fi

Good morning, I am The Giver. I've come here to bestow a trait upon you. You will choose those that will help you in your struggles. Unfortunately, these traits do not las
After each Act, you will be granted the will to choose another Trait. Croaks the hooded figure.
You walk slowly towards them and give your hand.

Enter anything to continue.
|

```

D. Choose Trait:

The player has the ability to choose a trait. You'll be given the option to choose between 2 different traits, one is mainly for attack upgrades and two is mainly for defense upgrades. The game calculates the amount of traits you have to the attack damage and defense damage when in battle. When progressing to different acts of the story, you'll be asked to choose 2 different traits again. The traits counter will stack and increase your damage or defense in your progressing battles.

```

-----
Choose a trait:
-----
(1) Increased Strength
(2) Increased Defence
->
|

```

E. Show Story Part (Intros):

After choosing your trait, you'll be greeted with the intro story of each act.

```

-----
Act 1
-----
After inquiring your new trait, you start to move forward.
You enter the land known as The Valley of the Castle Gate. Past here is The Castle, which is the path you'll be taking to the Elden Beast.
However, There bares a barrier in between those two places and it can only be broken once you reach 5 XP. Defeat enemies and gain XP. That is the only way.
Getting ready to fight, you check your inventory to confirm the 3 potions of healing you've stored. Once done, you begin your journey.

Enter anything to continue.
|

```


F. Main Menu:

The gameplay starts here. you will be given the option to choose whether you want to continue your journey (check part G), show your character information (check part H), or exit the game. After each encounter (battle or resting), it will go back to this main menu screen.

This only changes when the amount of XP you gained from battles have reached the amount needed in order to progress the story; when that happens the program loops back to [part C] but the act will increase (act: 1 changes to act: 2) which progresses the game.

When you reach act 4, the program will show the story texts but will skip the main menu and go straight to the final battle (shown in part I).

```
-----  
Valley of the Castle Gate  
-----  
Choose an action:  
-----  
(1) Resume Journey  
(2) View Character Info  
(3) Exit Game  
->                                     I  
|
```

G. Choosing **Resume Journey**:

When resuming journey, you will mainly be *encountering a battle* or in rare occasions, *be asked to rest* to regain your hp. These encounters happen at random, but it's highly likely you'll be encountering battles more often than be asked to rest.

a. **Battle:**

When you start a battle, it will show the enemies' and your hp. You'll be given the option to fight, use your potion [of healing], or run away. If you die in battle, you will be shown the death screen (in part K).

- Screen that shows the **main battle screen**:

```
-----
Hellhound
HP: 5/5
-----
-----
Sans
HP: 20/20
-----
Choose your action.
-----
(1) Fight
(2) Use Potion
(3) Run Away
-> |
```

- After choosing to **(1)Fight**, it will display the results:

```
-----
Battle:
-----
You dealt 4 damage to the Hellhound.
-----
The Hellhound dealt 0 damage to you.

Enter anything to continue.
|
```

- After choosing to **(2)Use Potion**, it will ask you if you are sure that you want to take a potion. By default, you have 3 potions:

```
-----  
Do you want to drink a potion? (2 left.)  
-----  
(1) Yes  
(2) No, maybe later.  
->  
1|
```

- After choosing to **(3)Run Away**, you have the chance to either run away successfully which means you've escaped unharmed, or run away unsuccessfully (as shown below) which punishes you by taking hp:

```
-----  
You failed to escape successfully.  
-----  
In your attempt to flee, you took 3 damage.  
  
Enter anything to continue.  
|
```

- After **defeating the enemy**, it will show a result screen as shown below:

```
-----  
You defeated the Hellhound.  
-----  
You earned 2 XP!  
  
Enter anything to continue.  
|
```

b. Asked To Take A Rest:

This should not happen a lot as I've made it a rare occurrence but once it happens, the program will ask you if you want to rest or not. As shown, there is a rest counter, by default, you only have 1. If you have used a rest opportunity and you encounter a resting opportunity, it will simply just tell you that you don't have any more rests, therefore you cannot rest and be asked to continue. Continuing will lead you back to part F/the main menu.

```
-----
You found the rare opportunity to take a rest. Do you want to do so? (1 rest left).
-----
(1) Yes
(2) No, not right now.
-> I
|
```

H. Choosing View Character Info:

After choosing (2) in the main menu, it will show you information regarding your player such as you hp, xp, number of potions, and your traits. Continuing after this will go back to the main menu.

```
-----
Character Info
-----
Sans    HP: 20/20
-----
XP: 0
-----
Number of Potions: 3
-----
Attack Trait: Increased Strength
-----
Enter anything to continue. I
|
```

I. The Final Battle (Act 4):

As stated in part F, when you reach the final act (act 4), you will skip the Main Menu and go straight to battling the boss; the reason why is because you're in the final stage not encountering any random enemies or events so you won't need to choose to resume your journey.

The difference between the final boss battle and the previous battles is that the Elden Beast will have a random hp that's much higher than any other enemies, and in addition, you cannot try to run away, if you do try to, it will show you a screen telling you that you can't run away.

```
-----  
Elden Beast  
HP: 43/43  
-----  
-----  
sans  
HP: 15/20  
-----  
Choose your action.  
-----  
(1) Fight  
(2) Use Potion  
(3) Run Away  
->
```

- If you try to **Run Away in act 4**, you will be told that you cannot run as shown below:

```
-----  
The path behind is blocked, there is no escape from the Elden Beast.  
-----  
  
Enter anything to continue.
```

J. Ending/Win Screen:

After defeating the final boss, you'll be greeted with the ending story text and the program will end.

```
-----  
The End  
-----  
You've defeated the Elden Beast. Suddenly, the surrounding starts to flash and a blinding light appears. You find yourself in a grassy field under bright blue skies. It's p  
The hooded figure appears again.  
  
You did it. You've defeated him. Now the curse upon the land has been lifted. No more will we be suffering in that retched place. Thank you, Sans.  
  
You feel relieved. You now understand why you've been set upon your journey. You take a sit down on the grassy field and start to admire the beautiful sky. You feel tired a  
  
Thank you for playing this game. Credit to CodeStudent.  
  
Process finished with exit code 0
```

K. Death Screen:

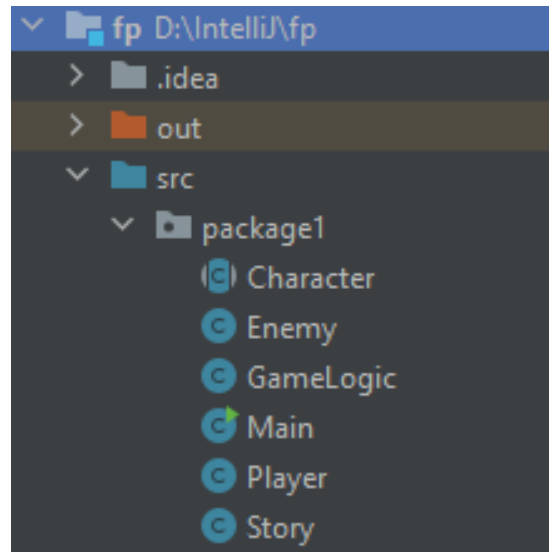
If at any point in the game, while battling random enemies or battling the final boss, and you die, you will be subjected to the death screen which tells you that you're dead and the program ends.

```
-----  
YOU ARE DEAD  
-----  
-----  
You earned 0 XP on your journey.  
-----  
Thank you for playing this game. Credit to CodeStudent.  
  
Process finished with exit code 0
```

4. Code Explanation

A. Program Structure:

The program structure is relatively simple here. There's only one package called "package1" and they contain the **Main** class, **GameLogic** class, abstract **Character** class, **Player** class, **Enemy** class, and **Story** class.



Some of the code screenshots aren't full code showcases, they're too long to include here, so please check the full code in my Github repository.

B. Main Class:

This is just the simple main class. It calls the `GameLogic.startGame()` method from `GameLogic.java` to start the game.

```
1 package package1;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         //Starting the game
7         GameLogic.startGame();
8     }
9 }
10
11
```

C. GameLogic class:

The GameLogic class includes variables and methods that dictate the game progression and [literal] events. It also includes all of the important methods that are used to dictate the game functions.

An example of a basic method that helps the game function is *readInt* int method which takes a string and an int as parameters, which is used to read the inputs from users and returns an int to be used for determining the player's chosen actions. Another example is *clearConsole* void method which just simulates clearing the console. These basic methods are small but help organize the overall program's presentation.

Examples of the important methods of this class include *startGame()*, *checkAct()*, *battle()*, and *gameLoop()*. *startGame()* is a method that starts the game, it displays the things needed for the intro, it asks for the user's name to set it up, and it's the method that checks to start the other important methods. *battle()* is the method that sets the battle system; it creates the enemies, calculates the damages taken and given to the player and the enemy, and has the functions for when the player wants to fight, take a potion, or run away. It's going to take long to explain all of it, so the point is that these important methods talk to each other to run/initialize/etc and help the game function.

So overall, the GameLogic class is the class that literally makes the game work. All of the methods talk to each other to create the game and set the bounds and rules for the game. I've added extensive commenting in the code itself so that anyone could try to understand it better.


```

1 package package1;
2 import java.util.Scanner;
3
4 public class GameLogic {
5     static Scanner scanner = new Scanner(System.in);
6
7     static Player player;
8
9     public static boolean isRunning;
10
11     //random encounters
12     public static String[] encounters = {"Battle", "Battle", "Battle", "Battle", "Rest"};
13
14     //enemy names
15     public static String[] enemies = {"Hellhound", "Hellhound", "Skeleton", "Skeleton", "You (Mimic)"};
16
17     //Story elements
18     public static int place = 0, act = 1;
19     public static String[] places = {"Valley of the Castle Gate", "The Castle", "The Dungeon", "The Dungeon - Throne Room"};
20
21
22
23
24
25
26
27 // the method to get user input from console
28 public static int readInt(String prompt, int userChoices) {
29     int input;
30
31     do{
32         System.out.println(prompt);
33         try{
34             input = Integer.parseInt(scanner.next());
35         } catch (Exception e){
36             input = -1;
37             System.out.println("Please enter an integer!");
38         }
39     } while (input < 1 || input > userChoices);
40

```

```

41     return input;
42 }
43
44 //method to simulate clearing out the console
45 public static void clearConsole(){
46     for(int i = 0; i < 100; i++)
47         System.out.println();
48 }
49
50 //method to print a separator with length n
51 public static void printSeparator (int n){
52     for(int i = 0; i < n; i++)
53         System.out.print("-");
54     System.out.println();
55 }
56
57 //method to print a heading
58 public static void printHeading(String title){
59     printSeparator( 30);
60     System.out.println(title);
61     printSeparator( 30);
62 }
63
64 //method to stop the game until user enters anything
65 public static void anythingToContinue(){
66     System.out.println("\nEnter anything to continue.");
67     scanner.next();
68 }
69
70
71
72
73
74
75 //method to start the game
76 public static void startGame(){
77     boolean nameSet = false;
78     String name;
79

```

D. abstract Character class:

This abstract class is the parent class for the Player and Enemy class. This class initializes the variables used for Player and Enemy; also has the abstract methods attack() and defend().

```
1 package package1;
2
3 public abstract class Character {
4
5     // The attributes of a character
6     public String name;
7     public int maxHp, hp, xp;
8
9     //constructor for character
10    public Character(String name, int maxHp, int xp){
11        this.name = name;
12        this.maxHp = maxHp;
13        this.xp = xp;
14        this.hp = maxHp;
15    }
16
17    //main methods/functions of the character
18    public abstract int attack();
19    public abstract int defend();
20
21 }
22
```

E. Player class:

This is one of the children classes of the Character parent class. Other than the variables and methods from the parent class, this Player class initializes the numAtkUpgrades and numDefUpgrades variable which stores how many attack and defense traits the player has.

In addition, it creates the string array of atkUpgrades and defUpgrades to store the names of the available upgrades in the game.

This class also includes the method chooseTrait() which asks the user to choose the traits they want based on the string arrays stated before, and then stores the one you choose to the variables that store attacks and defenses. It also uses the GameLogic methods to structure and organize the display of text.

```

1 package package1;
2
3 public class Player extends Character{
4
5     //ints to store number of upgrades/skills in each path
6     public int numAtkUpgrades, numDefUpgrades;
7
8     //additional player stats
9     int pots ,restsLeft;
10
11     //array that stores skill names
12     public String[] atkUpgrades = {"Increased Strength", "Increased Dexterity", "Might", "Godlike Strength"};
13     public String[] defUpgrades = {"Increased Defence", "Attack Resistance", "Holy Knight Armor", "A God's Blessing"};
14
15     //player specific constructor
16     public Player(String name) {
17         //calling constructor of superclass
18         super(name, maxHp: 20, xp: 0);
19         //Setting # of upgades to 0
20         this.numAtkUpgrades = 0;
21         this.numDefUpgrades = 0;
22         //set the additional stats
23         this.restsLeft = 1;
24         this.pots = 3;
25         //let the player choose trait when creating character
26         chooseTrait();
27     }
28
29     //override the superclass methods for specific methods in player.java
30     @Override
31     public int attack() {
32         return (int) (Math.random() * (xp / 4 + numAtkUpgrades * 3 + 3) + xp / 10 + numAtkUpgrades * 2 + numDefUpgrades + 1);
33     }
34
35     @Override
36     public int defend() {
37         return (int) (Math.random() * (xp / 2 + numDefUpgrades * 3 + 3) + xp / 10 + numDefUpgrades * 2 + numAtkUpgrades + 1);
38     }
39 }

```

F. Enemy Class:

This is the other children class of the Character parent class. It's overall very similar to the Player class, except that it does not have any other methods besides the *attack()* and *defend()* method. There's one new variable called *playerXp* which is an int variable that holds the count for the player's xp. The enemy uses the player xp to calculate the amount of attacks and defenses an enemy has; this acts as a gameplay balancer.

```

1 package package1;
2
3 public class Enemy extends Character {
4
5     //variable to store the player's current xp
6     int playerXp;
7
8     //enemy specific constructor
9     public Enemy(String name, int playerXp) {
10         super(name, (int) (Math.random() * playerXp + playerXp / 3 + 5), (int) (Math.random() * (playerXp / 4 + 2) + 1));
11         //assign variable
12         this.playerXp = playerXp;
13     }
14
15     //override the superclass methods for specific methods in enemy.java
16     @Override
17     public int attack() {
18         return (int) (Math.random() * (playerXp / 4 + 1) + xp / 4 + 3);
19     }
20
21     @Override
22     public int defend() {
23         return (int) (Math.random() * (playerXp / 4 + 1) + xp / 4 + 3);
24     }
25 }
26

```

G. Story Class

This class' only purpose is to store all of the methods to print out the story text. It uses some basic GameLogic functions [explained in previous parts] to help with the presentation of the story text printing.

```

1 package package1;
2
3 //class that stores the methods of displaying the parts of the story
4 public class Story {
5
6     public static void printIntro(){
7         GameLogic.clearConsole();
8         GameLogic.printSeparator(m: 30);
9         System.out.println("The Story");
10        GameLogic.printSeparator(m: 30);
11        System.out.println("You awaken in a cave. Besides you is a bonfire with a sword stuck into it. You remember that you
12        System.out.println("You do not know what you are. All you know is that you've been assigned to this journey. For what
13        System.out.println("As you walk towards the light leading outside of the tunnel, a hooded figure stands at the edge
14        System.out.println("");
15        System.out.println("Good morning, I am The Giver. I've come here to bestow a trait upon you. You will choose those
16        System.out.println("After each Act, you will be granted the will to choose another Trait. Croaks the hooded figure.
17        System.out.println("You walk slowly towards them and give your hand.");
18        GameLogic.anythingToContinue();
19    }
20
21
22    public static void printFirstActIntro(){
23        GameLogic.clearConsole();
24        GameLogic.printSeparator(m: 30);
25        System.out.println("Act 1");
26        GameLogic.printSeparator(m: 30);
27        System.out.println("After inquiring your new trait, you start to move forward.");
28        System.out.println("You enter the land known as The Valley of the Castle Gate. Past here is The Castle, which is the
29        System.out.println("However, There bares a barrier in between those two places and it can only be broken once you r
30        System.out.println("Getting ready to fight, you check your inventory to confirm the 3 potions of healing you've sto
31        GameLogic.anythingToContinue();
32    }
33 }
34

```

5. Lessons Learned

There were tons of things that I've learned from this project. I am not a good programmer so having done this project and learning from various different sources made me understand more about the concepts of Object Oriented Programming and how to program in Java in general. One thing that I did not understand about OOP is how each class can talk to each other to achieve a goal that the user wants, how they can connect, etc.. Although I still need a lot of work, this helped me understand the things I am confused about more than nothing.

6. Project Github Link

https://github.com/Sans-drian/Sandrian_OOP_FinalProject_L2CC.git

7. References

- CodeStudent on Youtube
- <https://stackoverflow.com/>
- <https://www.javatpoint.com/>
- <https://www.geeksforgeeks.org/>
- <https://www.programiz.com/>
- <https://www.w3schools.com/>