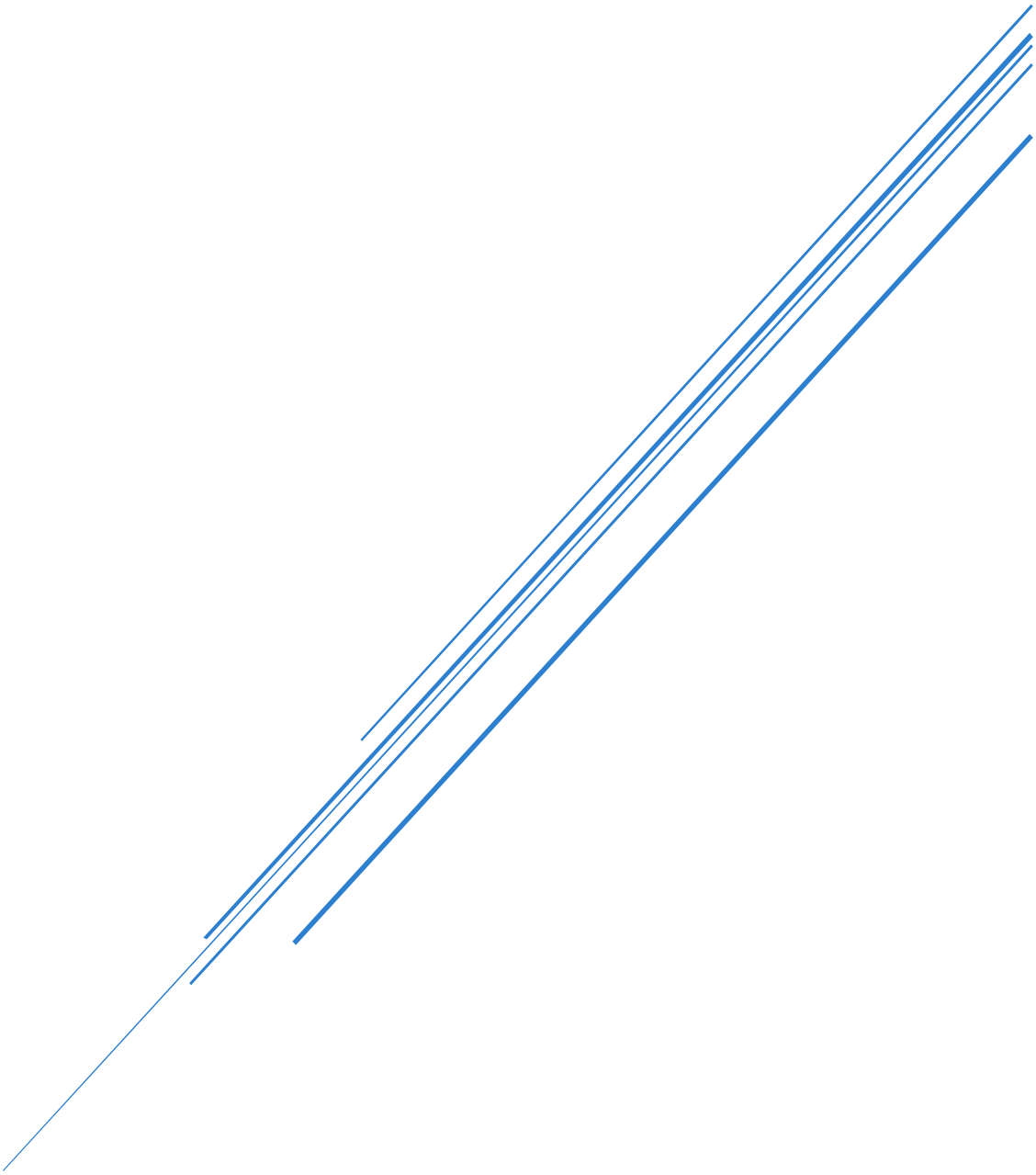
ITP4507 ASSIGNMENT

Report



Fung Wai Lim (230070312)

HONG KONG INSTITUTE OF VOCATIONAL EDUCATION

DEPARTMENT OF INFORMATION TECHNOLOGY

HIGHER DIPLOMA IN SOFTWARE ENGINEERING (IT114105)

ITP4507 Assignment

Module Name: Contemporary Topics in Software Engineering

Module Code: ITP4507

Student Name: Fung Wai Lim

Student ID: 230070312

Date: 15 Nov 2024

I declare that no part of this submission has been copied from any other student's work or from any other source except where due acknowledgement is made explicitly in the text, nor has any part been written for us by another person.

Table of Content

Introduction

Snow Storm Company is developing an RPG game titled "Fantastic World (FW)" for PC, where players engage with various characters known as HEROs, each possessing unique traits. Currently, players can choose between two hero types: Warriors, who excel in defense, and Warlocks, who specialize in magic damage. As the game evolves, additional hero types like healers will be introduced, necessitating a flexible design that adheres to the Open Closed Principle.

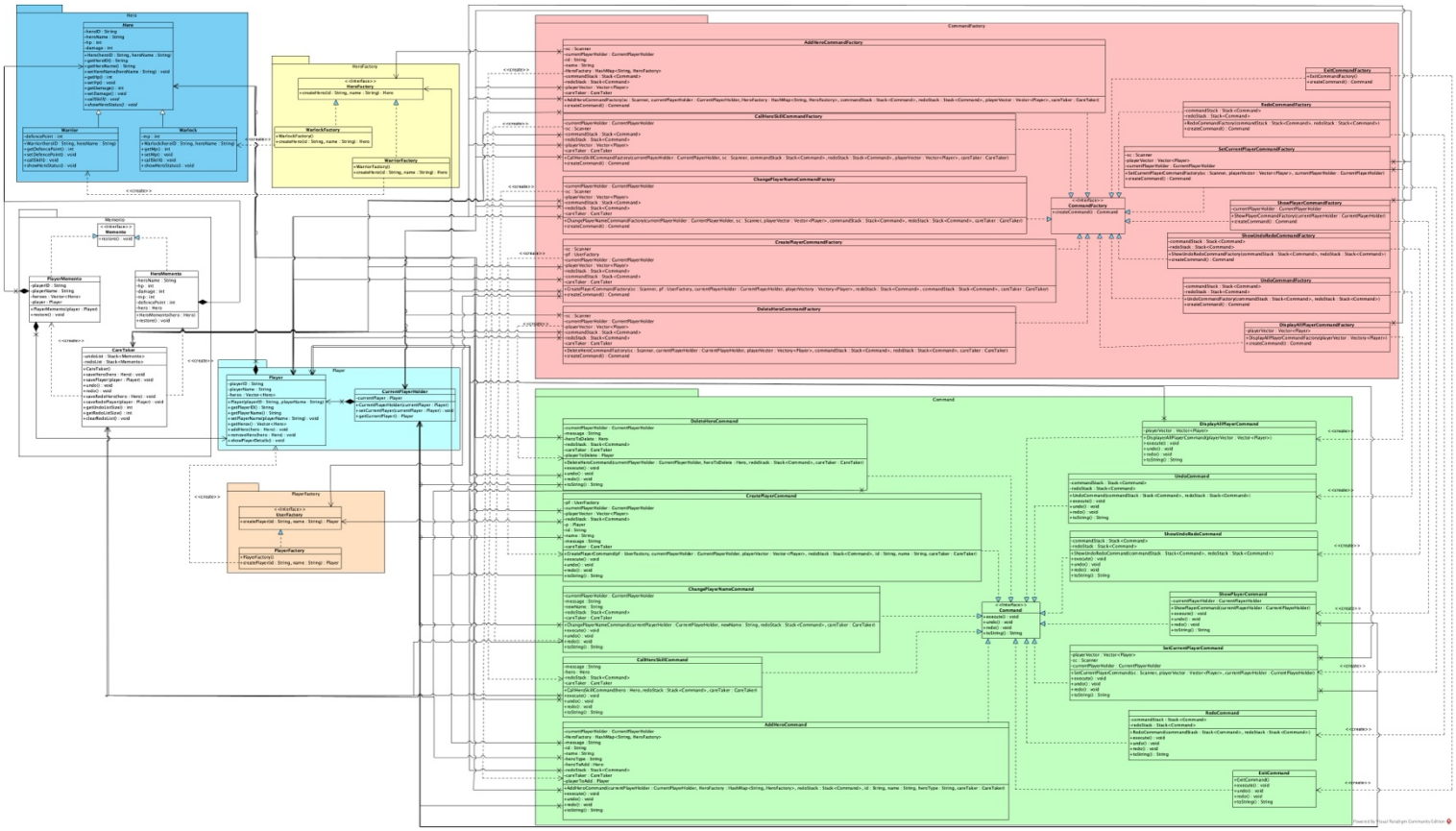
This report will outline key assumptions made during the system's design process and explore how specific design patterns—such as Command, Factory, and Memento—are applied to ensure that the system remains extensible for future hero additions.

Assumptions regarding the problem context

1. The user cannot create new player using existing player ID
2. Each player can have more than one heroes
3. The user cannot create new hero using existing hero ID
4. The user cannot call a hero that the mp or defence point is 0
5. The user cannot perform undo action if undo list is empty
6. The user cannot perform redo action if redo list is empty
7. The user cannot input command input which does not exist
8. The user cannot input empty value for any input
9. The user cannot delete hero that does not belong to current player

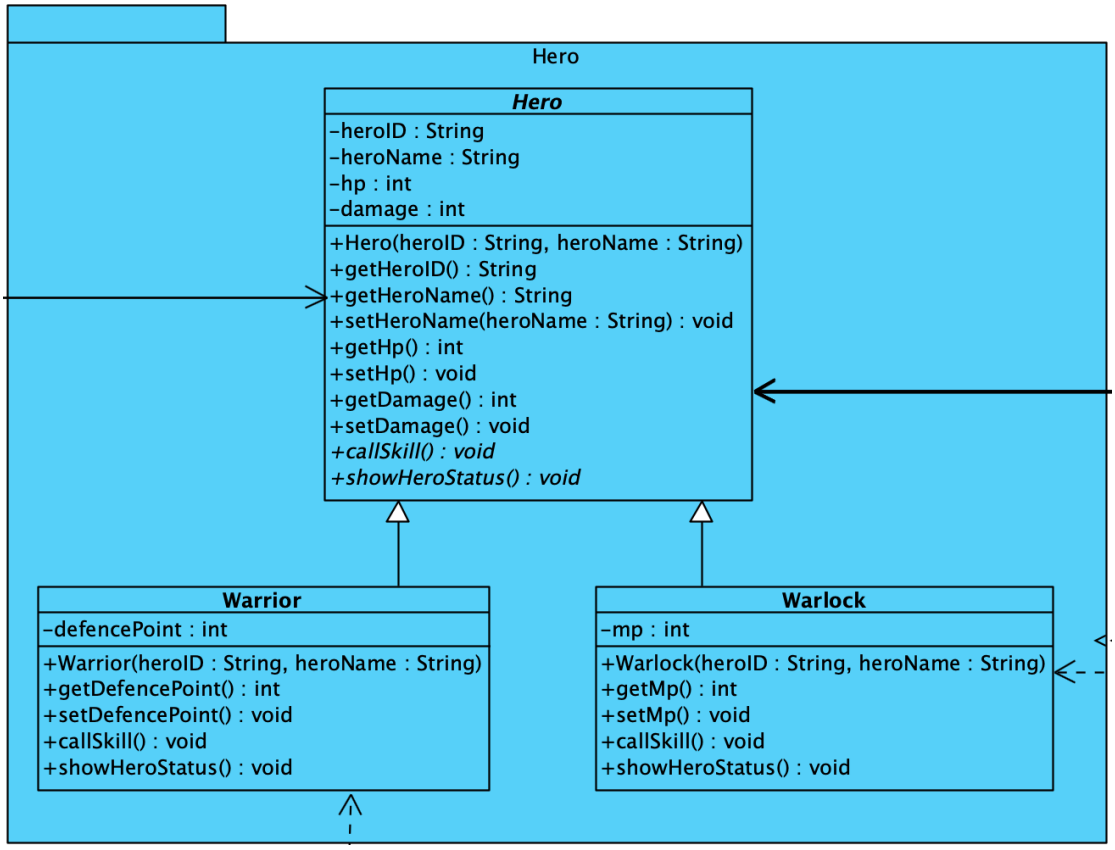
Application design with class diagram

Full Diagram



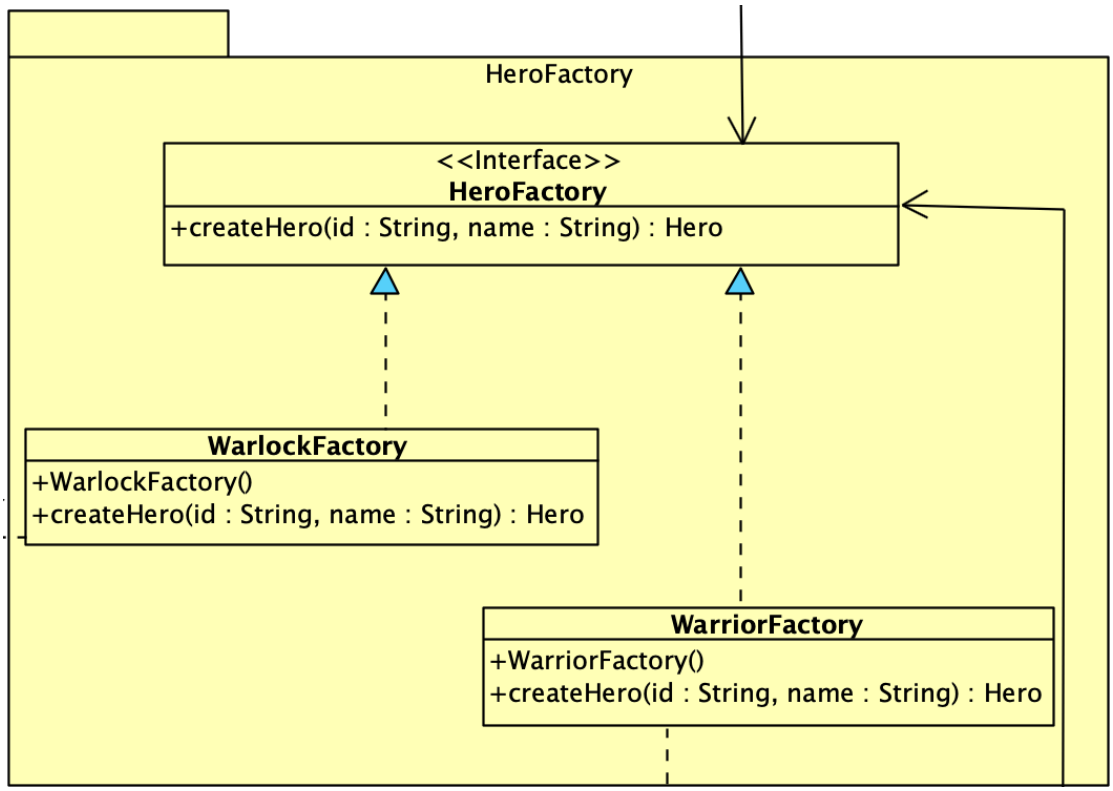
Hero Package

This is the hero package which consist of 3 classes: Hero, Warrior and Warlock. The Hero abstract class is the super class of Warrior and Warlock, which define some common attribute and method that all hero should have. The Warrior and Warlock class inherit the Hero class and add their own attribute and method such as defencePoint and getDefencePoint().



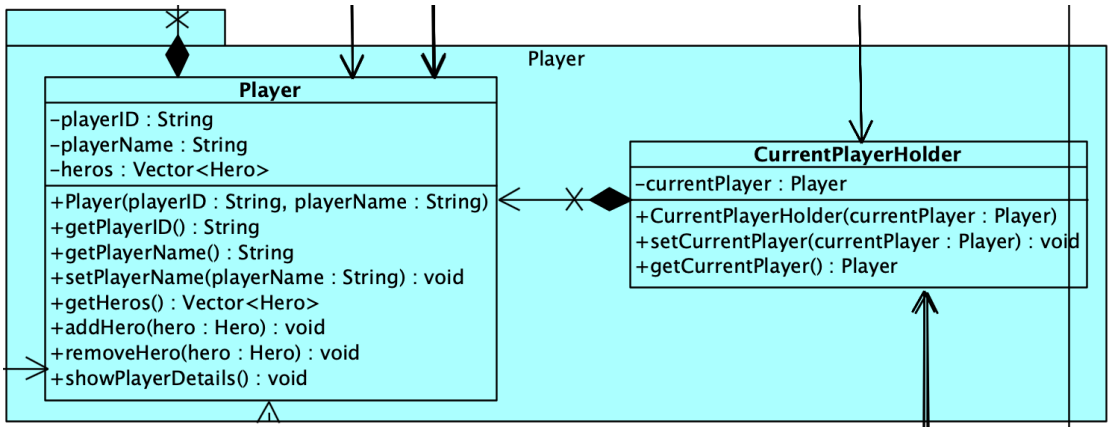
Hero Factory Package

This is the hero factory package which consist of 3 classes: HeroFactory, WarlockFactory and WarriorFactory. The HeroFactory is an interface which define the method it subclass should have, the createHero() is the only method for each hero factory to have. The WarlockFactory and WarriorFactory inherit the HeroFactory interface and implement the createHero() method to create heroes.



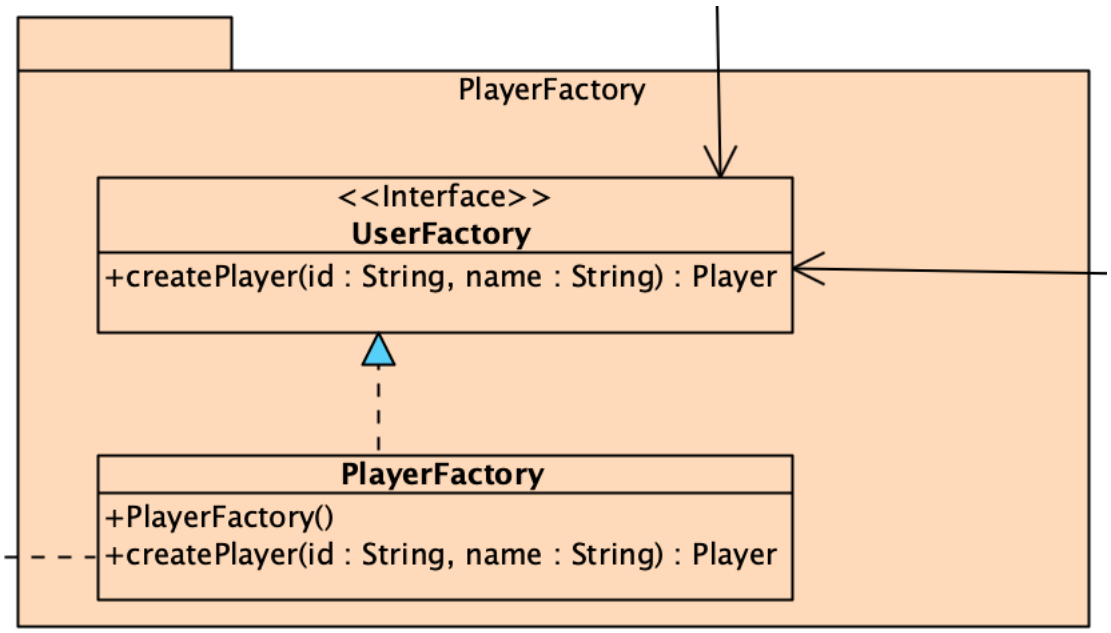
Player Package

This is the player package which consist of 2 classes: Player and CurrentPlayerHolder. The Player Class is the player to play the game, it will be created by PlayerFactory in the Player Factory Package. The CurrentPlayerHolder is to hold the current player of the game, it has a getter and setter to get the current player and set the current player.



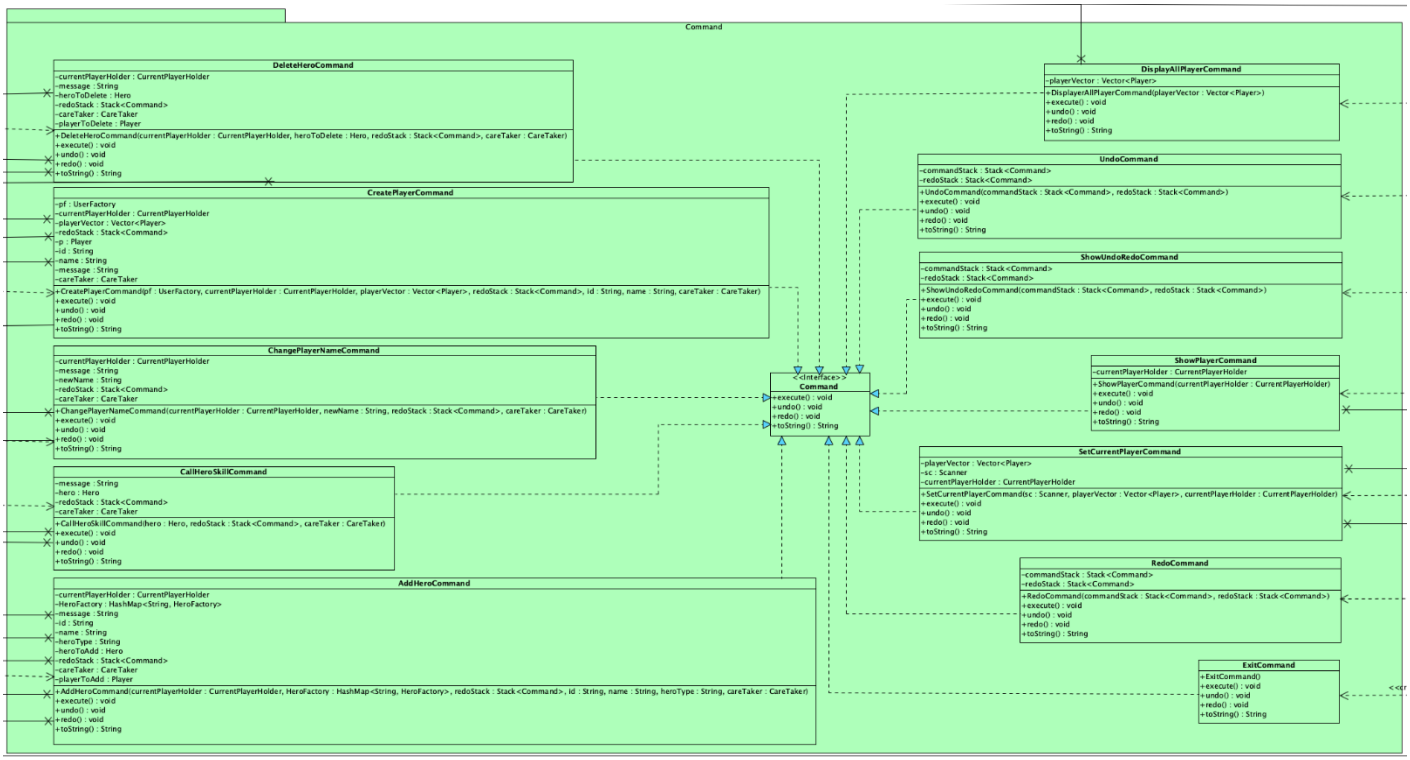
Player Factory Package

This is the player factory package which consist of 2 class: UserFactory and PlayerFactory. The UserFactory is an interface which define the method it subclass should have, the createPlayer() is the only method for player factory to have. The PlayerFactory inherit the UserFactory interface and implement the createPlayer() method. It returns a new player when user want to create new player.



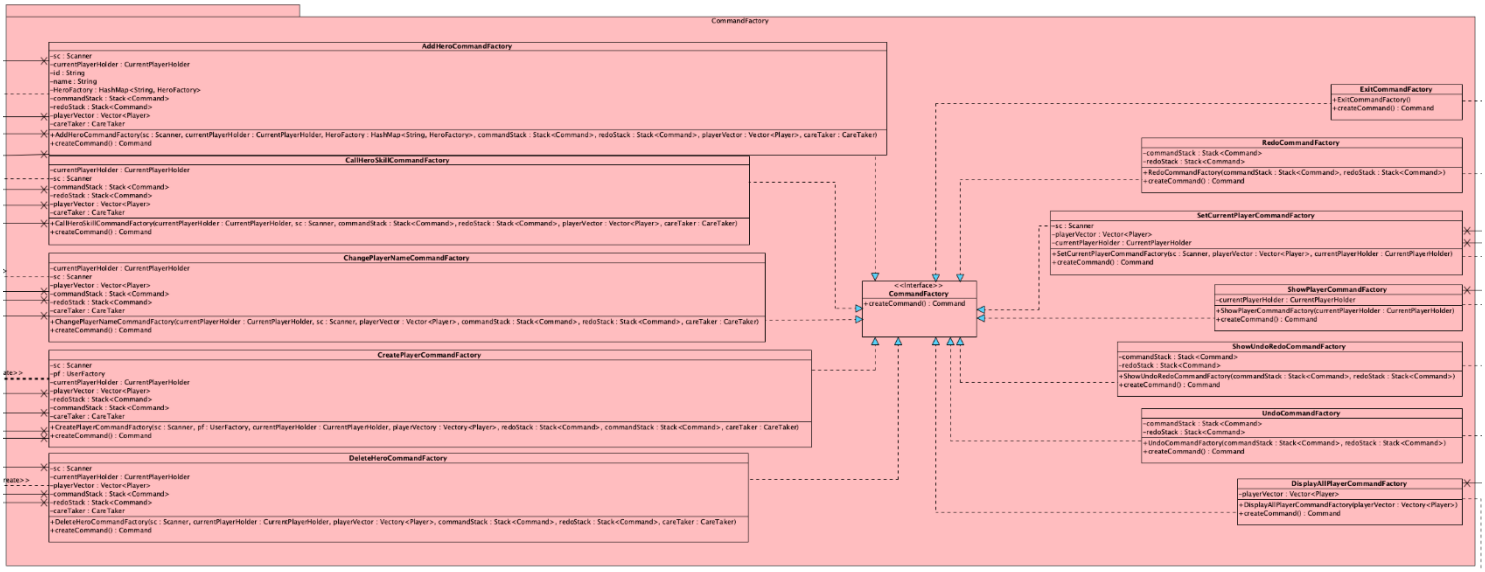
Command Package

This is the command package which consist of 13 classes, which include an interface call Command. The Command interface define what method each command should have included execute(), undo(), redo() and toString(). Each command class inherit the Command interface and implement the above four method. Once the user input a new command, the corresponding command class will be created by command factory and call the execute() method to perform action



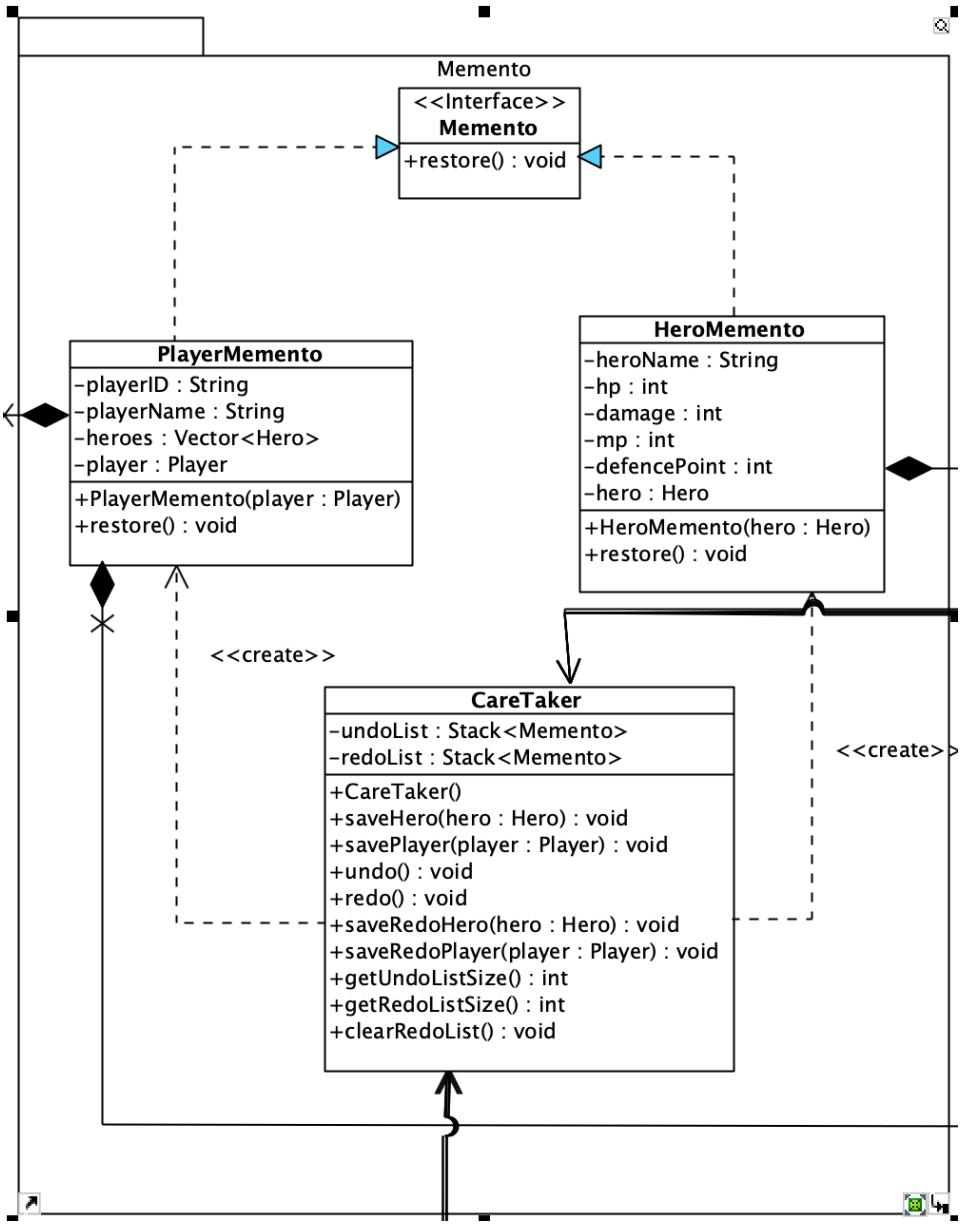
Command Factory Package

This is the command factory package which consist of 13 class, which include a CommandFactory interface. The CommandFactory is an interface which define the method it subclass should have, the createCommand() is the only method for each command factory to have. Each command factory class inherit the CommandFactory interface and implement the createCommand() method. Once user input for a command, the corresponding command factory class will create and return a command, the command created will be executed to perform user's action



Memento Package

This is the memento package which consist of 4 class: Memento, PlayerMemento, HeroMemento and CareTaker. The Memento is an interface which define each memento's method, restore() is the only method defined by Memento interface. The PlayerMemento and HeroMemento inherit the Memento interface to save player when the player's name is changed and save hero when the hero's skill is called respectively. The CareTaker class is to save the memento when the command executed, it also provides undo and redo function to restore the PlayerMemento and HeroMemento to the previous state.



Discussion and explanation on each of the design patterns applied to the application

Command Pattern

The command pattern is used in command package. The Command interface define four abstract method that each command class should implement including execute(), undo(), redo() and toString().

Role	Class
Invoker	Main
Command	Command
Concrete Command	AddHeroCommand CallHeroSkillCommand ChangePlayerNameCommand CreatePlayerCommand DeleteHeroCommand DisplayAllPlayerCommand ExitCommand RedoCommand SetCurrentPlayerCommand ShowPlayerCommand ShowUndoRedoCommand UndoCommand
Invoker	WarriorFactory WarlockFactory Warlock Warrior Player PlayerFactory CurrentPlayerHolder

Abstract Factory Pattern

The abstract factory pattern is used in CommandFactory package. The CommandFactory interface define an abstract method call createCommand() that each command factory class should implement. Each command factory implements the createCommand() method to create a new command object per execution.

Role	Class
Abstract Factory	CommandFactory
Concrete Factory	AddHeroCommandFactory CallHeroSkillCommandFactory ChangePlayerNameCommandFactory CreatePlayerCommandFactory DeleteHeroCommandFactory DisplayAllPlayerCommandFactory ExitCommandFactory RedoCommandFactory SetCurrentPlayerCommandFactory ShowPlayerCommandFactory ShowUndoRedoCommandFactory UndoCommandFactory
Concrete Product	AddHeroCommand CallHeroSkillCommand ChangePlayerNameCommand CreatePlayerCommand DeleteHeroCommand DisplayAllPlayerCommand ExitCommand RedoCommand SetCurrentPlayerCommand ShowPlayerCommand ShowUndoRedoCommand UndoCommand
Client	Main

The abstract factory pattern is used in HeroFactory package. The HeroFactory interface define an abstract method call createHero() that each command factory class should implement. Each hero factory implements the createHero() method to create a new hero object per execution.

Role	Class
Abstract Factory	HeroFactory
Concrete Factory	WarlockFactory WarriorFactory
Concrete Product	Warlock Warrior
Client	Main

The abstract factory pattern is used in Player package. The UserFactory interface define an abstract method call createPlayer() that the PlayerFactory class should implement. PlayerFactory implement the createPlayer() method to create a new player object per execution.

Role	Class
Abstract Factory	UserFactory
Concrete Factory	PlayerFactory
Concrete Product	Player
Client	Main

Memento Pattern

The memento pattern is use in the Memento package. The Memento interface define a restore() method that it's subclass should implement. The PlayerMemento and HeroMemento inherit Memento interface and implement the restore() method. When a new memento is created, it will be saved as the current state of Player or Hero right before the changes is made. If user perform undo or redo action, it will be restored so that the Player or Hero change back to the saved state. The CareTaker is used to create a new memento and save it. It also responsible to perform undo and redo action.

Role	Class
Caretaker	UserFactory
Originator	Player Warlock Warrior
Memento	PlayerMemento HeroMemento

Test Plan and Test Cases

General Test

c

P001

Thomas Yiu

a

H001, peter pang

1

a

H002, john wick

2

s

c

P002

Stan Lee

p

a

H003, scarlet witch

2

a

H004, tony stark

1

s

g

P001

m

H001

s

d

H002

t

Russo Brothers

s

l

u

u

u

s

u

u

g

P002

s

u

g

P002

l

r

r

r

s

r

g

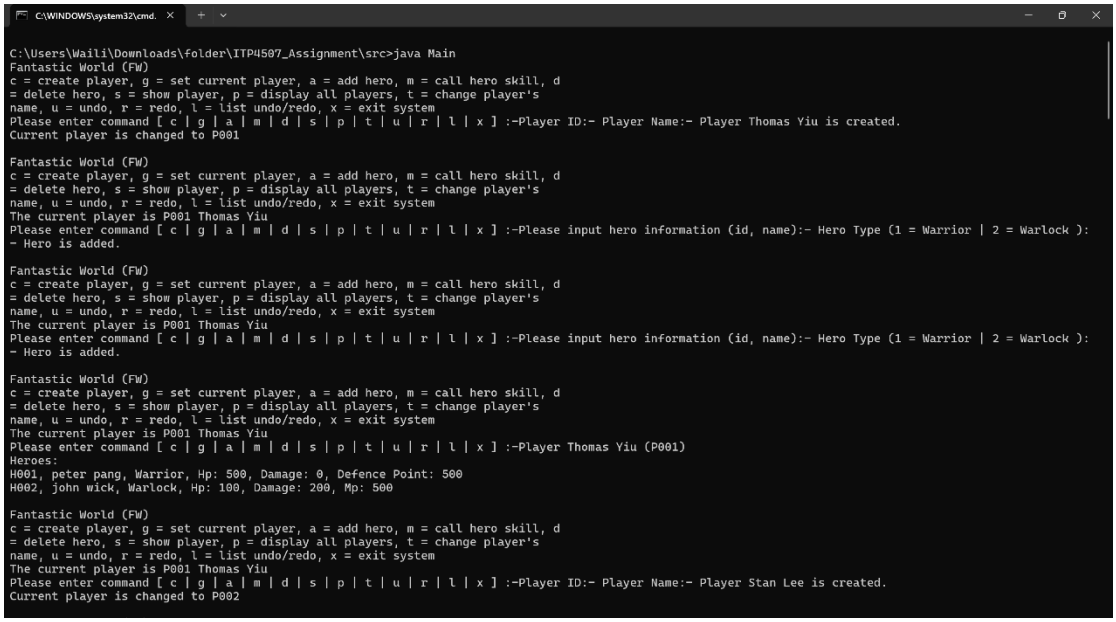
P001

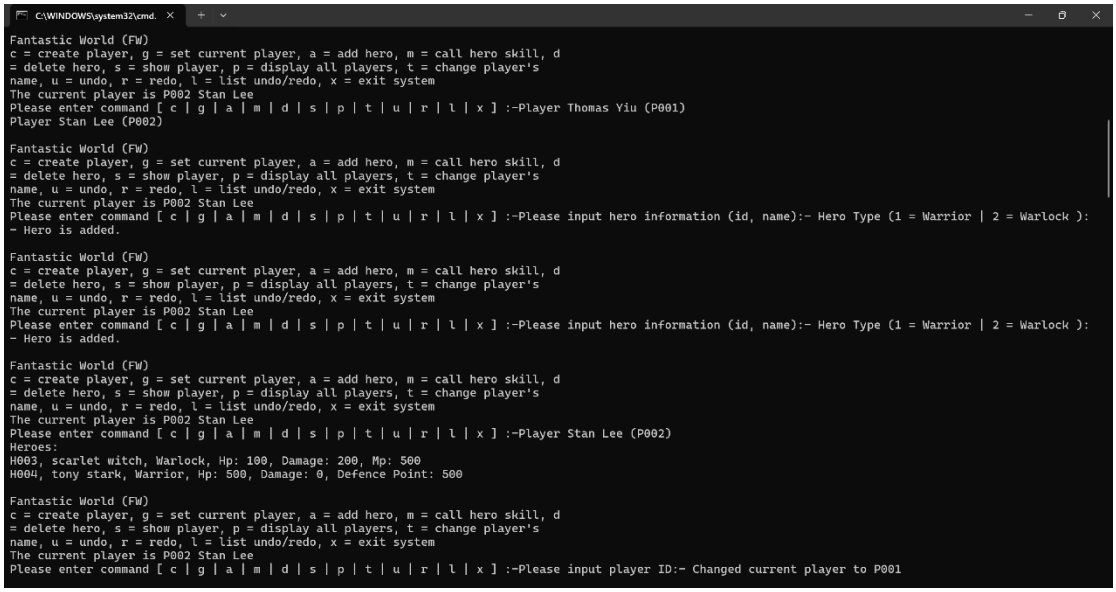
s

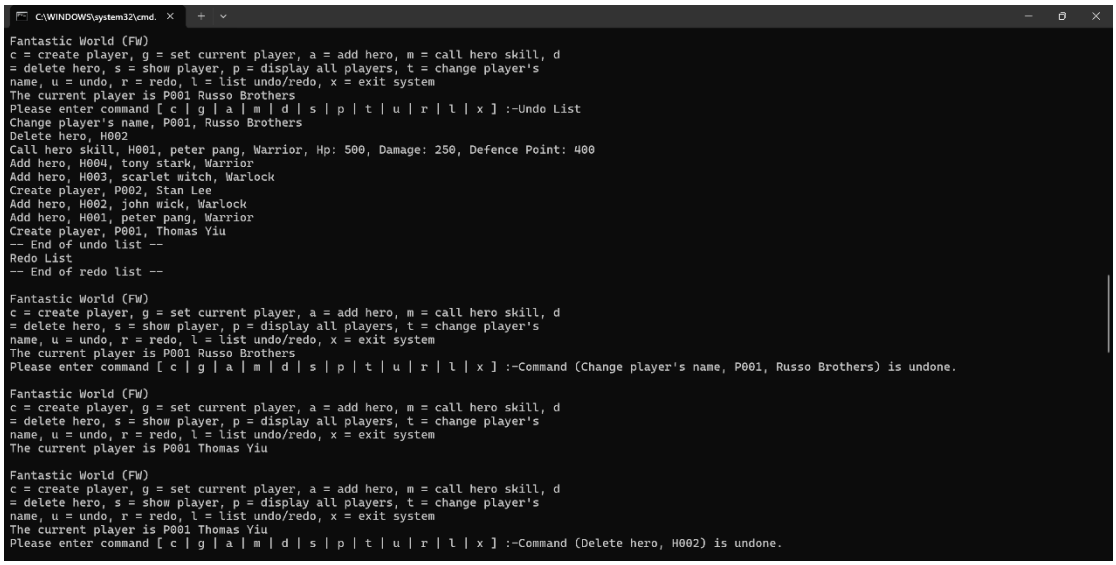
l

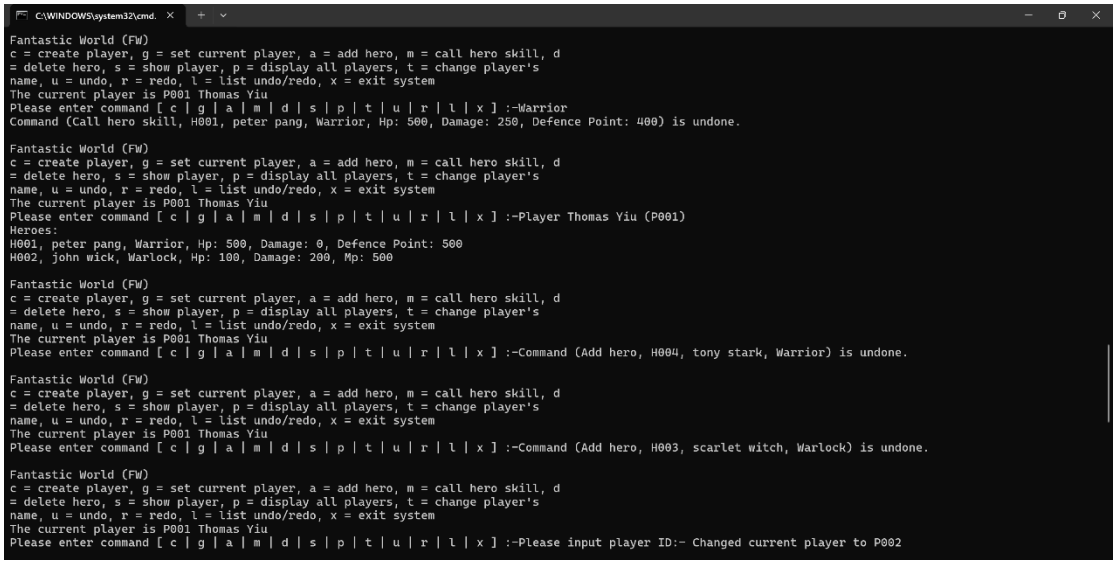
x

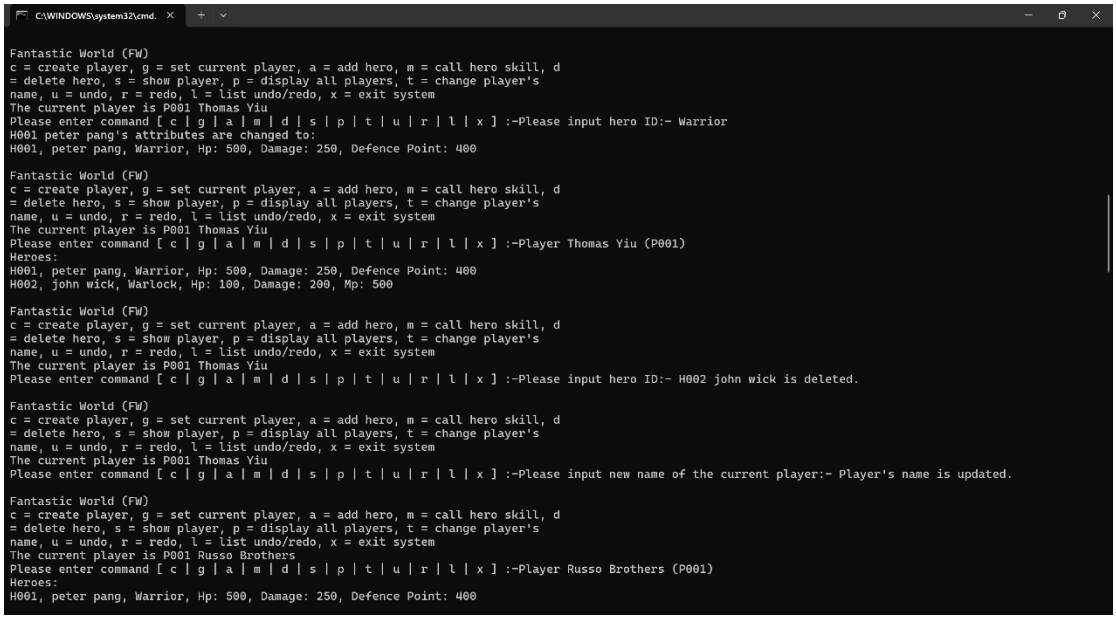
Expected Output of General Test

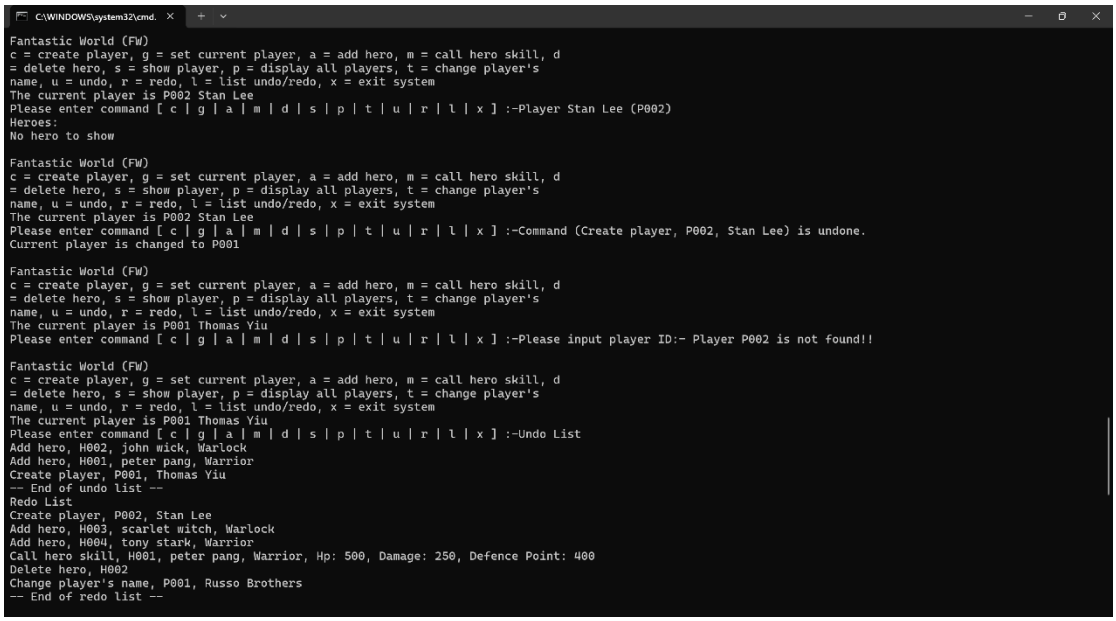


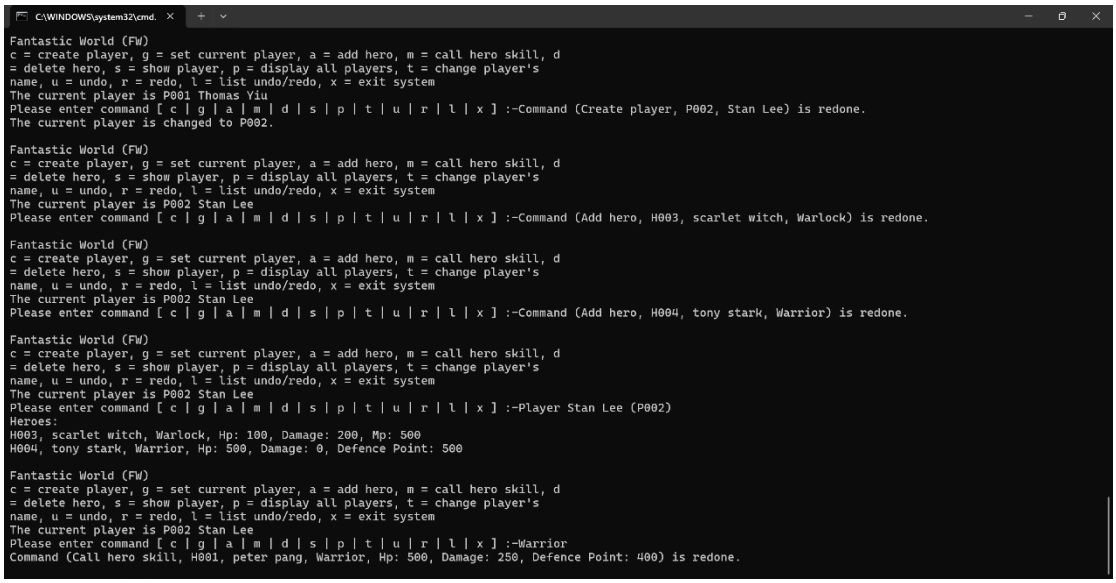


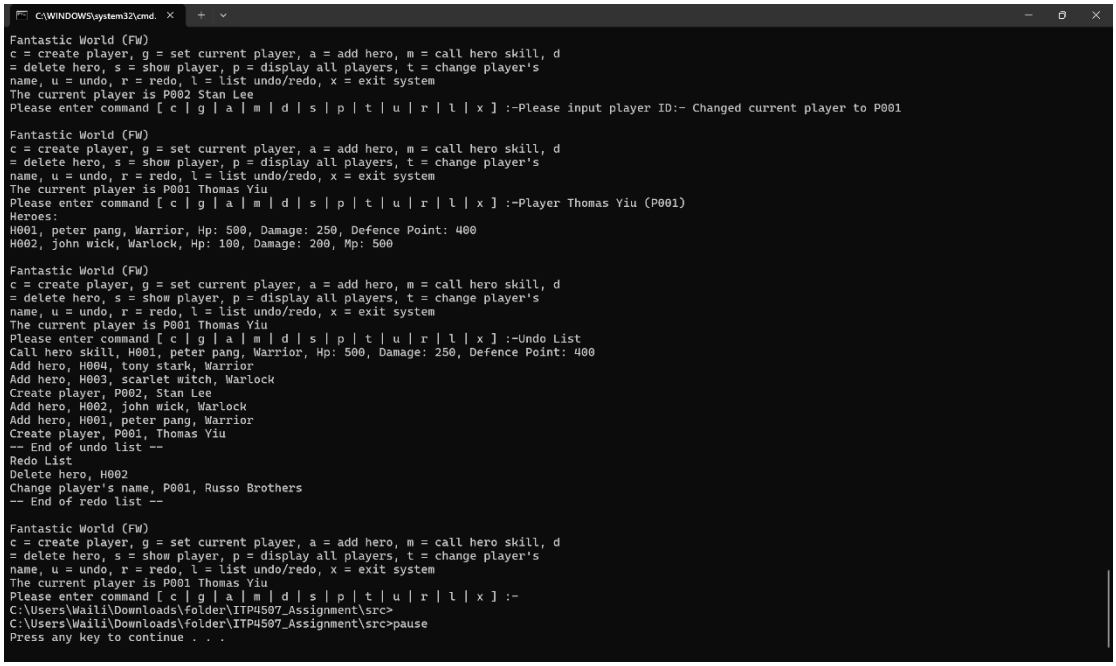

C:\WINDOWS\system32\cmd.
Fantastic World (FW)
c = create player, g = set current player, a = add hero, m = call hero skill, d
= delete hero, s = show player, p = display all players, t = change player's
name, u = undo, r = redo, l = list undo/redo, x = exit system
The current player is P001 Russo Brothers
Please enter command [c | g | a | m | d | s | p | t | u | r | l | x] :-Undo List
Change player's name, P001, Russo Brothers
Delete hero, H002
Call hero skill, H001, peter pang, Warrior, Hp: 500, Damage: 250, Defence Point: 400
Add hero, H004, tony stark, Warrior
Add hero, H003, scarlet witch, Warlock
Create player, P002, Stan Lee
Add hero, H002, john wick, Warlock
Add hero, H001, peter pang, Warrior
Create player, P001, Thomas Yiu
-- End of undo list --
Redo List
-- End of redo list --
Fantastic World (FW)
c = create player, g = set current player, a = add hero, m = call hero skill, d
= delete hero, s = show player, p = display all players, t = change player's
name, u = undo, r = redo, l = list undo/redo, x = exit system
The current player is P001 Russo Brothers
Please enter command [c | g | a | m | d | s | p | t | u | r | l | x] :-Command (Change player's name, P001, Russo Brothers) is undone.
Fantastic World (FW)
c = create player, g = set current player, a = add hero, m = call hero skill, d
= delete hero, s = show player, p = display all players, t = change player's
name, u = undo, r = redo, l = list undo/redo, x = exit system
The current player is P001 Thomas Yiu
Fantastic World (FW)
c = create player, g = set current player, a = add hero, m = call hero skill, d
= delete hero, s = show player, p = display all players, t = change player's
name, u = undo, r = redo, l = list undo/redo, x = exit system
The current player is P001 Thomas Yiu
Please enter command [c | g | a | m | d | s | p | t | u | r | l | x] :-Command (Delete hero, H002) is undone.


C:\WINDOWS\system32\cmd.
Fantastic World (FW)
c = create player, g = set current player, a = add hero, m = call hero skill, d
= delete hero, s = show player, p = display all players, t = change player's
name, u = undo, r = redo, l = list undo/redo, x = exit system
The current player is P001 Thomas Yiu
Please enter command [c | g | a | m | d | s | p | t | u | r | l | x] :-Warrior
Command (Call hero skill, H001, peter pang, Warrior, Hp: 500, Damage: 250, Defence Point: 400) is undone.
Fantastic World (FW)
c = create player, g = set current player, a = add hero, m = call hero skill, d
= delete hero, s = show player, p = display all players, t = change player's
name, u = undo, r = redo, l = list undo/redo, x = exit system
The current player is P001 Thomas Yiu
Please enter command [c | g | a | m | d | s | p | t | u | r | l | x] :-Player Thomas Yiu (P001)
Heroes:
H001, peter pang, Warrior, Hp: 500, Damage: 0, Defence Point: 500
H002, john wick, Warlock, Hp: 100, Damage: 200, Mp: 500
Fantastic World (FW)
c = create player, g = set current player, a = add hero, m = call hero skill, d
= delete hero, s = show player, p = display all players, t = change player's
name, u = undo, r = redo, l = list undo/redo, x = exit system
The current player is P001 Thomas Yiu
Please enter command [c | g | a | m | d | s | p | t | u | r | l | x] :-Command (Add hero, H004, tony stark, Warrior) is undone.
Fantastic World (FW)
c = create player, g = set current player, a = add hero, m = call hero skill, d
= delete hero, s = show player, p = display all players, t = change player's
name, u = undo, r = redo, l = list undo/redo, x = exit system
The current player is P001 Thomas Yiu
Please enter command [c | g | a | m | d | s | p | t | u | r | l | x] :-Command (Add hero, H003, scarlet witch, Warlock) is undone.
Fantastic World (FW)
c = create player, g = set current player, a = add hero, m = call hero skill, d
= delete hero, s = show player, p = display all players, t = change player's
name, u = undo, r = redo, l = list undo/redo, x = exit system
The current player is P001 Thomas Yiu
Please enter command [c | g | a | m | d | s | p | t | u | r | l | x] :-Please input player ID:- Changed current player to P002


C:\WINDOWS\system32\cmd.
Fantastic World (FW)
c = create player, g = set current player, a = add hero, m = call hero skill, d
= delete hero, s = show player, p = display all players, t = change player's
name, u = undo, r = redo, l = list undo/redo, x = exit system
The current player is P001 Thomas Yiu
Please enter command [c | g | a | m | d | s | p | t | u | r | l | x] :-Please input hero ID:- Warrior
H001 peter pang's attributes are changed to:
H001, peter pang, Warrior, Hp: 500, Damage: 250, Defence Point: 400
Fantastic World (FW)
c = create player, g = set current player, a = add hero, m = call hero skill, d
= delete hero, s = show player, p = display all players, t = change player's
name, u = undo, r = redo, l = list undo/redo, x = exit system
The current player is P001 Thomas Yiu
Please enter command [c | g | a | m | d | s | p | t | u | r | l | x] :-Player Thomas Yiu (P001)
Heroes:
H001, peter pang, Warrior, Hp: 500, Damage: 250, Defence Point: 400
H002, john wick, Warlock, Hp: 100, Damage: 200, Mp: 500
Fantastic World (FW)
c = create player, g = set current player, a = add hero, m = call hero skill, d
= delete hero, s = show player, p = display all players, t = change player's
name, u = undo, r = redo, l = list undo/redo, x = exit system
The current player is P001 Thomas Yiu
Please enter command [c | g | a | m | d | s | p | t | u | r | l | x] :-Please input hero ID:- H002 john wick is deleted.
Fantastic World (FW)
c = create player, g = set current player, a = add hero, m = call hero skill, d
= delete hero, s = show player, p = display all players, t = change player's
name, u = undo, r = redo, l = list undo/redo, x = exit system
The current player is P001 Thomas Yiu
Please enter command [c | g | a | m | d | s | p | t | u | r | l | x] :-Please input new name of the current player:- Player's name is updated.
Fantastic World (FW)
c = create player, g = set current player, a = add hero, m = call hero skill, d
= delete hero, s = show player, p = display all players, t = change player's
name, u = undo, r = redo, l = list undo/redo, x = exit system
The current player is P001 Russo Brothers
Please enter command [c | g | a | m | d | s | p | t | u | r | l | x] :-Player Russo Brothers (P001)
Heroes:
H001, peter pang, Warrior, Hp: 500, Damage: 250, Defence Point: 400


C:\WINDOWS\system32\cmd.
Fantastic World (FW)
c = create player, g = set current player, a = add hero, m = call hero skill, d
= delete hero, s = show player, p = display all players, t = change player's
name, u = undo, r = redo, l = list undo/redo, x = exit system
The current player is P002 Stan Lee
Please enter command [c | g | a | m | d | s | p | t | u | r | l | x] :-Player Stan Lee (P002)
Heroes:
No hero to show
Fantastic World (FW)
c = create player, g = set current player, a = add hero, m = call hero skill, d
= delete hero, s = show player, p = display all players, t = change player's
name, u = undo, r = redo, l = list undo/redo, x = exit system
The current player is P002 Stan Lee
Please enter command [c | g | a | m | d | s | p | t | u | r | l | x] :-Command (Create player, P002, Stan Lee) is undone.
Current player is changed to P001
Fantastic World (FW)
c = create player, g = set current player, a = add hero, m = call hero skill, d
= delete hero, s = show player, p = display all players, t = change player's
name, u = undo, r = redo, l = list undo/redo, x = exit system
The current player is P001 Thomas Yiu
Please enter command [c | g | a | m | d | s | p | t | u | r | l | x] :-Please input player ID:- Player P002 is not found!!
Fantastic World (FW)
c = create player, g = set current player, a = add hero, m = call hero skill, d
= delete hero, s = show player, p = display all players, t = change player's
name, u = undo, r = redo, l = list undo/redo, x = exit system
The current player is P001 Thomas Yiu
Please enter command [c | g | a | m | d | s | p | t | u | r | l | x] :-Undo List
Add hero, H002, john wick, Warlock
Add hero, H001, peter pang, Warrior
Create player, P001, Thomas Yiu
-- End of undo list --
Redo List
Create player, P002, Stan Lee
Add hero, H003, scarlet witch, Warlock
Add hero, H004, tony stark, Warrior
Call hero skill, H001, peter pang, Warrior, Hp: 500, Damage: 250, Defence Point: 400
Delete hero, H002
Change player's name, P001, Russo Brothers
-- End of redo list --


C:\WINDOWS\system32\cmd.
Fantastic World (FW)
c = create player, g = set current player, a = add hero, m = call hero skill, d
= delete hero, s = show player, p = display all players, t = change player's
name, u = undo, r = redo, l = list undo/redo, x = exit system
The current player is P001 Thomas Yiu
Please enter command [c | g | a | m | d | s | p | t | u | r | l | x] :-Command (Create player, P002, Stan Lee) is redone.
The current player is changed to P002.
Fantastic World (FW)
c = create player, g = set current player, a = add hero, m = call hero skill, d
= delete hero, s = show player, p = display all players, t = change player's
name, u = undo, r = redo, l = list undo/redo, x = exit system
The current player is P002 Stan Lee
Please enter command [c | g | a | m | d | s | p | t | u | r | l | x] :-Command (Add hero, H003, scarlet witch, Warlock) is redone.
Fantastic World (FW)
c = create player, g = set current player, a = add hero, m = call hero skill, d
= delete hero, s = show player, p = display all players, t = change player's
name, u = undo, r = redo, l = list undo/redo, x = exit system
The current player is P002 Stan Lee
Please enter command [c | g | a | m | d | s | p | t | u | r | l | x] :-Command (Add hero, H004, tony stark, Warrior) is redone.
Fantastic World (FW)
c = create player, g = set current player, a = add hero, m = call hero skill, d
= delete hero, s = show player, p = display all players, t = change player's
name, u = undo, r = redo, l = list undo/redo, x = exit system
The current player is P002 Stan Lee
Please enter command [c | g | a | m | d | s | p | t | u | r | l | x] :-Player Stan Lee (P002)
Heroes:
H003, scarlet witch, Warlock, Hp: 100, Damage: 200, Mp: 500
H004, tony stark, Warrior, Hp: 500, Damage: 0, Defence Point: 500
Fantastic World (FW)
c = create player, g = set current player, a = add hero, m = call hero skill, d
= delete hero, s = show player, p = display all players, t = change player's
name, u = undo, r = redo, l = list undo/redo, x = exit system
The current player is P002 Stan Lee
Please enter command [c | g | a | m | d | s | p | t | u | r | l | x] :-Warrior
Command (Call hero skill, H001, peter pang, Warrior, Hp: 500, Damage: 250, Defence Point: 400) is redone.


C:\WINDOWS\system32\cmd.
Fantastic World (FW)
c = create player, g = set current player, a = add hero, m = call hero skill, d
= delete hero, s = show player, p = display all players, t = change player's
name, u = undo, r = redo, l = list undo/redo, x = exit system
The current player is P002 Stan Lee
Please enter command [c | g | a | m | d | s | p | t | u | r | l | x] :-Please input player ID:- Changed current player to P001
Fantastic World (FW)
c = create player, g = set current player, a = add hero, m = call hero skill, d
= delete hero, s = show player, p = display all players, t = change player's
name, u = undo, r = redo, l = list undo/redo, x = exit system
The current player is P001 Thomas Yiu
Please enter command [c | g | a | m | d | s | p | t | u | r | l | x] :-Player Thomas Yiu (P001)
Heroes:
H001, peter pang, Warrior, Hp: 500, Damage: 250, Defence Point: 400
H002, john wick, Warlock, Hp: 100, Damage: 200, Mp: 500
Fantastic World (FW)
c = create player, g = set current player, a = add hero, m = call hero skill, d
= delete hero, s = show player, p = display all players, t = change player's
name, u = undo, r = redo, l = list undo/redo, x = exit system
The current player is P001 Thomas Yiu
Please enter command [c | g | a | m | d | s | p | t | u | r | l | x] :-Undo List
Call hero skill, H001, peter pang, Warrior, Hp: 500, Damage: 250, Defence Point: 400
Add hero, H004, tony stark, Warrior
Add hero, H003, scarlet witch, Warlock
Create player, P002, Stan Lee
Add hero, H002, john wick, Warlock
Add hero, H001, peter pang, Warrior
Create player, P001, Thomas Yiu
-- End of undo list --
Redo List
Delete hero, H002
Change player's name, P001, Russo Brothers
-- End of redo list --
Fantastic World (FW)
c = create player, g = set current player, a = add hero, m = call hero skill, d
= delete hero, s = show player, p = display all players, t = change player's
name, u = undo, r = redo, l = list undo/redo, x = exit system
The current player is P001 Thomas Yiu
Please enter command [c | g | a | m | d | s | p | t | u | r | l | x] :-
C:\Users\Waili\Downloads\folder\ITP4507_Assignment\src>
C:\Users\Waili\Downloads\folder\ITP4507_Assignment\src>pause
Press any key to continue . . .

Invalid Input Test

k ← invalid command

c

P01

Player 1

a

H01, Hero1

3 ← invalid hero type

4 ← invalid hero type

1

c

P02

Player 2

g

P09 ← invalid playerID

g

P01

m

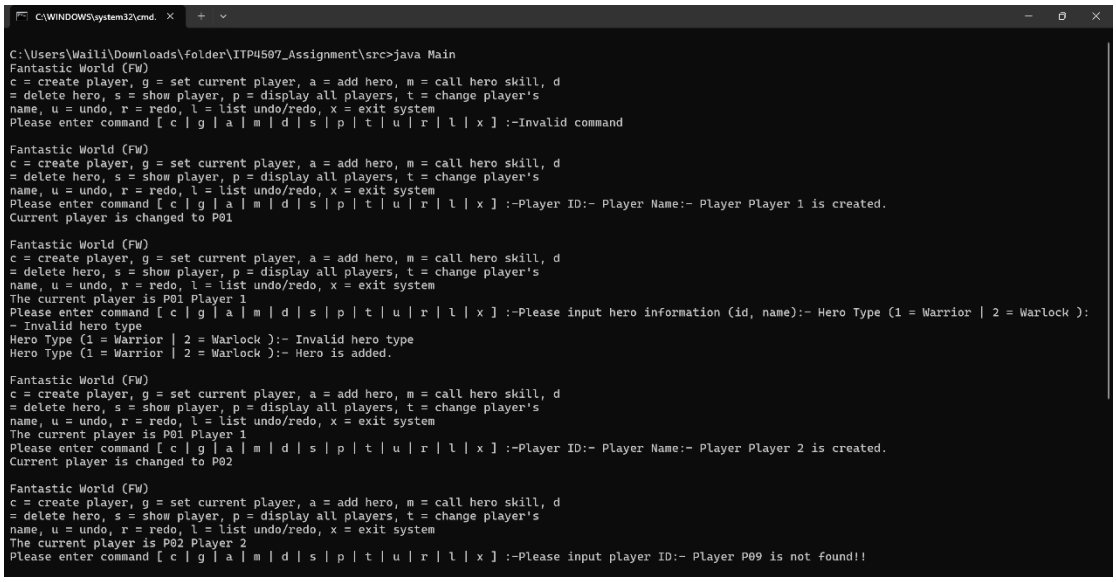
H09 ← invalid heroID

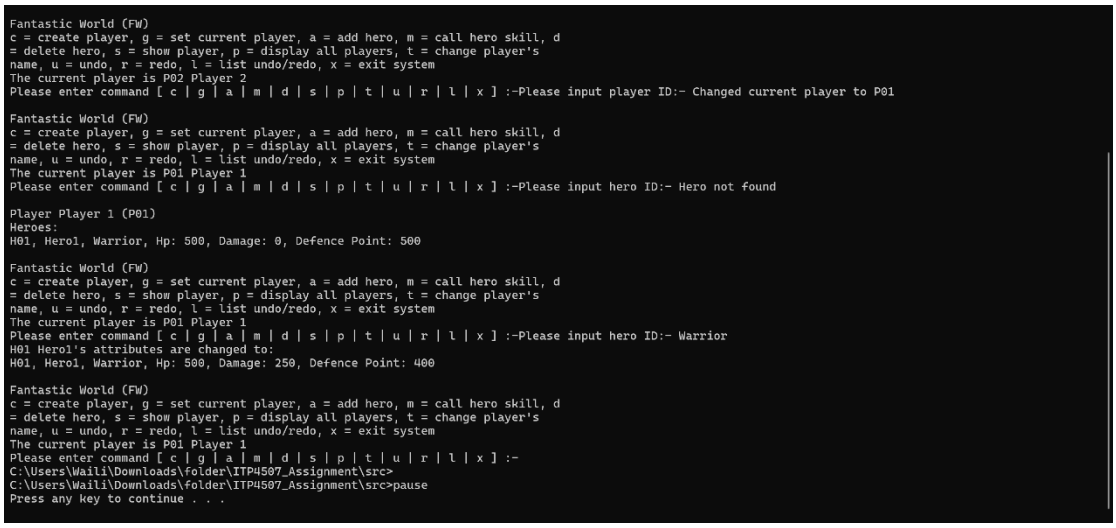
m

H01

x

Expect Output of Invalid Input Test





Duplicate Creation Test

c

P01

Player 1

c

P01 ←Duplicate Player ID

c

P02

Player 2

c

P02 ←Duplicate Player ID

c

P03

Player 3

a

H01, Hero1

1

a

H01, Hero2 ←Duplicate Hero ID

H02, Hero1 ←Test duplicate hero name but different name, should success

1

g

P01

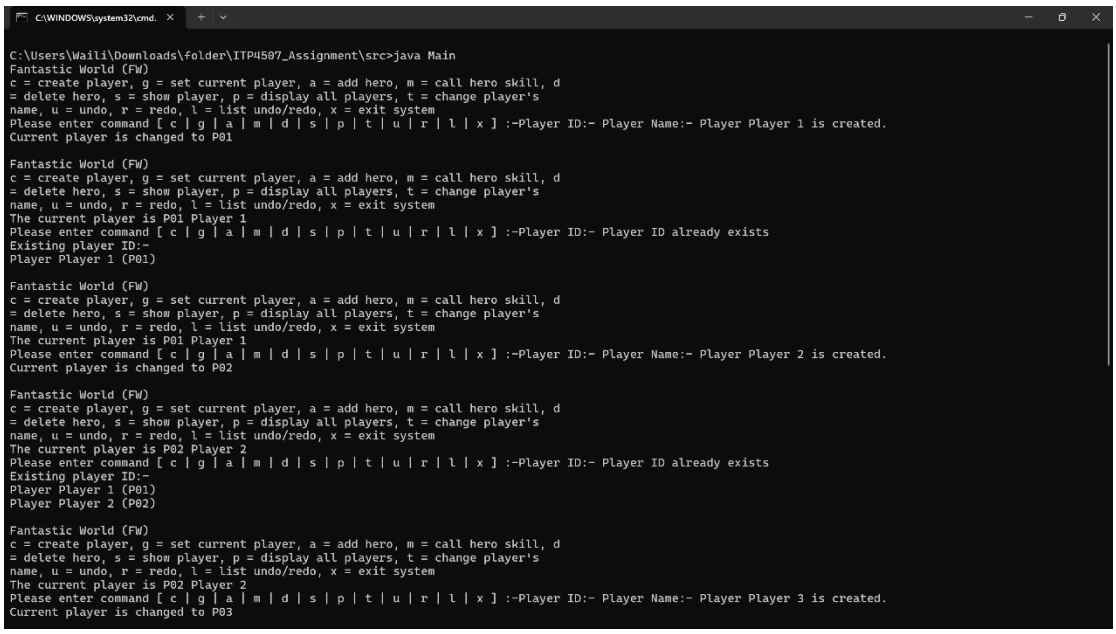
a

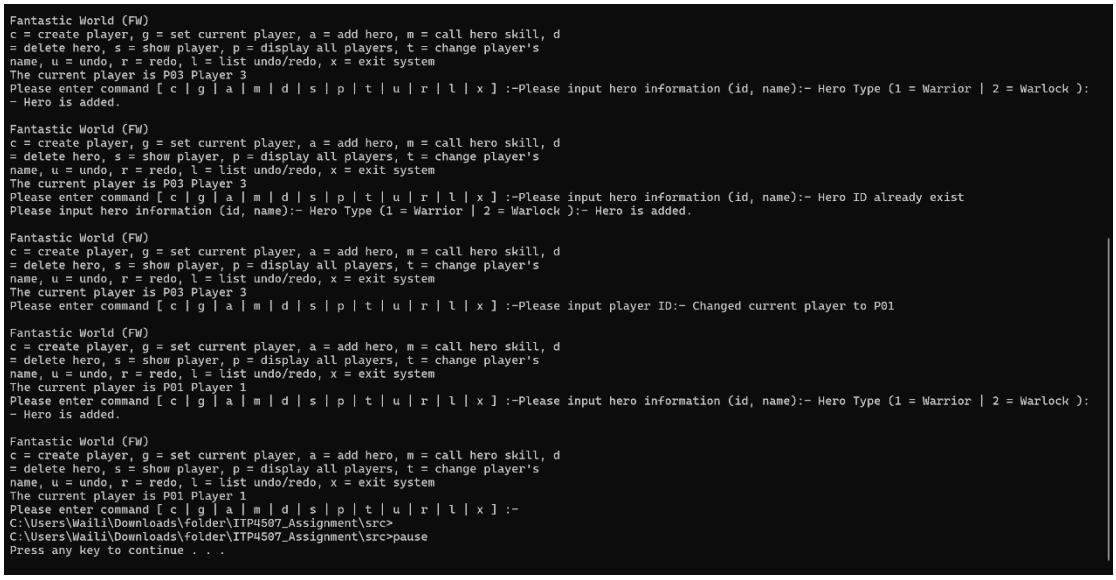
H01, Hero99 ← different player add heroID exist in other player, should success

1

x

Expected Output of Duplicate Creation Test





Empty Undo/ Redo Test

u ←undo list is empty now

r ←redo list is empty now

c

P01

Player 1

a

H01, Hero1

1

m

H01

u

u

u

u ←undo list is empty now

r

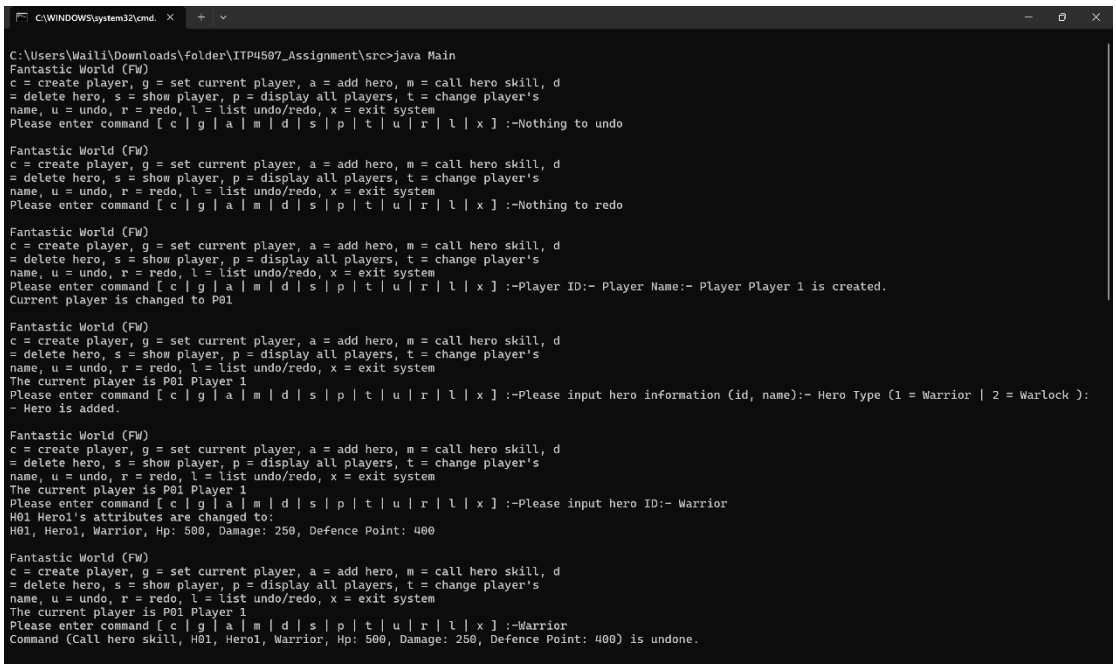
r

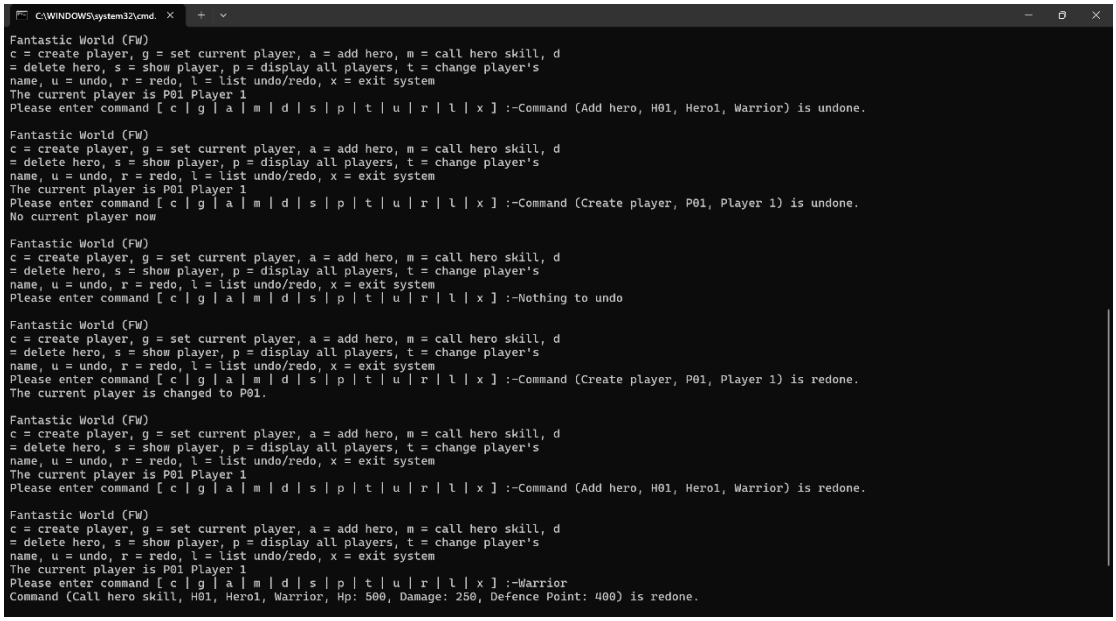
r

r ←redo list is empty now

x

Expected Output of Empty Undo/ Redo Test





Well documented Source Code

Main.java

```
import CommandFactory.*;
import HeroFactory.*;
import Memento.*;
import PlayerFactory.*;
import Command.*;
import Player.*;
import java.util.*;

public class Main {
	public static Scanner sc = new Scanner(System.in);

	public static void main(String[] args) {
		Vector<Player> playerVector = new Vector<Player>(); // Vector to store all players
		Stack<Command> commandStack = new Stack<Command>(); // Stack to store executed commands
		Stack<Command> redoStack = new Stack<Command>(); // Stack to store all commands to be redo

		CurrentPlayerHolder currentPlayerHolder = new CurrentPlayerHolder(null); // store current player

		HeroFactory warlockFactory = new WarlockFactory();
		HeroFactory warriorFactory = new WarriorFactory();
		UserFactory pf = new PlayerFactory();

		HashMap<String, HeroFactory> HeroFactory = new HashMap<>();
		HeroFactory.put("1", warriorFactory);
		HeroFactory.put("2", warlockFactory);

		CareTaker careTaker = new CareTaker();

		HashMap<String, CommandFactory> commandFactories = new HashMap<>();
		commandFactories.put("c", new CreatePlayerCommandFactory(sc, pf, currentPlayerHolder, playerVector,
redoStack, commandStack, careTaker));
		commandFactories.put("x", new ExitCommandFactory());
		commandFactories.put("u", new UndoCommandFactory(commandStack, redoStack));
		commandFactories.put("r", new RedoCommandFactory(commandStack, redoStack));
```

```
        commandFactories.put("l", new ShowUndoRedoCommandFactory(commandStack, redoStack));
        commandFactories.put("s", new ShowPlayerCommandFactory(currentPlayerHolder));
        commandFactories.put("p", new DisplayAllPlayerCommandFactory(playerVector));
        commandFactories.put("g", new SetCurrentPlayerCommandFactory(sc, playerVector, currentPlayerHolder));
        commandFactories.put("a", new AddHeroCommandFactory(sc, currentPlayerHolder, HeroFactory,
commandStack, redoStack, playerVector, careTaker));
        commandFactories.put("d", new DeleteHeroCommandFactory(sc, currentPlayerHolder, playerVector,
commandStack, redoStack, careTaker));
        commandFactories.put("m", new CallHeroSkillCommandFactory(currentPlayerHolder, sc, commandStack,
redoStack, playerVector, careTaker));
        commandFactories.put("t", new ChangePlayerNameCommandFactory(currentPlayerHolder, sc, playerVector,
commandStack, redoStack, careTaker));


        while (true) {
            System.out.println("Fantastic World (FW) \n" +
                    "c = create player, g = set current player, a = add hero, m = call hero skill, d \n" +
                    "= delete hero, s = show player, p = display all players, t = change player's \n" +
                    "name, u = undo, r = redo, l = list undo/redo, x = exit system ");
            if (currentPlayerHolder.getCurrentPlayer() != null) {
                System.out.println("The current player is " + currentPlayerHolder.getCurrentPlayer().getPlayerID() +
" " +
                        currentPlayerHolder.getCurrentPlayer().getPlayerName());
            }
            System.out.print("Please enter command [ c | g | a | m | d | s | p | t | u | r | l | x ] :-");


            try{
                String input = sc.nextLine();
                commandFactories.get(input).createCommand().execute();
            }catch (Exception e){
                System.out.println("Invalid command");
            }
            System.out.println();
        }
    }
}
```

Hero Package

Hero.java

```
package Hero;

public abstract class Hero {
    private String heroID;
    private String heroName;
    private int hp;
    private int damage;

    public Hero(String heroID, String heroName) {
        this.heroID = heroID;
        this.heroName = heroName;
        this.hp = 200;
    }

    public String getHeroID() {
        return heroID;
    }

    public String getHeroName() {
        return heroName;
    }

    public void setHeroName(String heroName) {
        this.heroName = heroName;
    }

    public int getHp() {
        return hp;
    }

    public void setHp(int hp) {
        this.hp = hp;
    }
```

```
    public int getDamage() {
        return damage;
    }

    public void setDamage(int damage) {
        this.damage = damage;
    }

    public abstract void callSkill();

    public abstract void showHeroStatus();

}
```

Warlock.java

```
package Hero;

public class Warlock extends Hero {
    private int mp;

    public Warlock(String heroID, String heroName) {
        super(heroID, heroName);
        this.setHp(100);
        this.mp = 500;
        this.setDamage(200);
    }

    public int getMp() {
        return mp;
    }

    public void setMp(int mp) {
        this.mp = mp;
    }

    @Override
    public void callSkill() {
```

```
            if (mp > 0) mp -= 100;
            else mp = 0;
            setDamage(100);
        }

        @Override
        public void showHeroStatus() {
            System.out.println(getHeroID() +
                    ", "+getHeroName() + ", Warlock, Hp: "+getHp() +
                    ", Damage: "+getDamage() + ", Mp: "+mp);
        }
}
```

Warrior.java

```
package Hero;

public class Warrior extends Hero {
    private int defencePoint;

    public Warrior(String heroID, String heroName) {
        super(heroID, heroName);
        this.defencePoint = 500;
        this.setHp(500);
        this.setDamage(0);
    }

    public int getDefencePoint() {
        return defencePoint;
    }

    public void setDefencePoint(int defencePoint) {
        this.defencePoint = defencePoint;
    }

    @Override
    public void callSkill() {
        if (defencePoint<0) defencePoint = 0;
```

```
            this.setDamage(defencePoint/2);
            defencePoint -= 100;
        }

        @Override
        public void showHeroStatus() {
            System.out.println(getHeroID() +
                    ", "+getHeroName() + ", Warrior, Hp: "+getHp() +
                    ", Damage: "+getDamage() + ", Defence Point: "+defencePoint);
        }
}
```

HeroFactory Package

HeroFactory.java

```
package HeroFactory;
import Hero.*;

public interface HeroFactory {
	Hero createHero(String id, String name);
}
```

WarlockFactory.java

```
package HeroFactory;

import Hero.*;

public class WarlockFactory implements HeroFactory {
	public Hero createHero(String id, String name) {
		return new Warlock(id, name);
	}
}
```

WarriorFactory.java

```
package HeroFactory;

import Hero.*;

public class WarriorFactory implements HeroFactory {
	public Hero createHero(String id, String name) {
		return new Warrior(id, name);
	}
}
```

Player Package

Player.java

```
package Player;
import Hero.*;
import java.util.Vector;

public class Player {
    private String playerID;
    private String playerName;
    private Vector<Hero> heroes;

    public Player(String playerID, String playerName) {
        this.playerID = playerID;
        this.playerName = playerName;
        heroes = new Vector<>();
    }

    public String getPlayerID() {
        return playerID;
    }

    public String getPlayerName() {
        return playerName;
    }

    public void setPlayerName(String playerName) {
        this.playerName = playerName;
    }

    public Vector<Hero> getHeroes() {
        return heroes;
    }

    public void addHero(Hero hero) {
        heroes.add(hero);
    }
```

```
    public void removeHero(Hero hero) {
        heroes.remove(hero);
    }

    public void showPlayerDetails() {
        System.out.println("Player " + getPlayerName() + " (" + getPlayerID() + ")");
        System.out.println("Heroes: ");
        Vector<Hero> playerHeroVector = getHeroes();
        if (!heroes.isEmpty()) {
            for (int i = 0; i < heroes.size(); i++) {
                playerHeroVector.get(i).showHeroStatus();
            }
        } else {
            System.out.println("No hero to show");
        }
    }
}
```

CurrentPlayerHolder.java

```
package Player;
public class CurrentPlayerHolder {
    private Player currentPlayer;
    public CurrentPlayerHolder(Player currentPlayer) {
        this.currentPlayer = currentPlayer;
    }

    public void setCurrentPlayer(Player currentPlayer) {
        this.currentPlayer = currentPlayer;
    }

    public Player getCurrentPlayer() {
        return currentPlayer;
    }
}
```

PlayerFactory Package

UserFactory.java

```
package PlayerFactory;


import Player.*;


public interface UserFactory {
	public Player createPlayer(String id, String name);
}
```

PlayerFactory.java

```
package PlayerFactory;


import Player.*;


public class PlayerFactory implements UserFactory {
	public Player createPlayer(String id, String name){
		return new Player(id,name);
	}
}
```

Command Package

Command.java

```
package Command;


public interface Command {
	public void execute();
	public void undo();
	public void redo();
	public String toString();
}
```

AddHeroCommand.java

```
package Command;


import Hero.*;
import Memento.*;
import Player.*;
import java.util.*;
import HeroFactory.*;


public class AddHeroCommand implements Command {
	private CurrentPlayerHolder currentPlayerHolder;
	private HashMap<String, HeroFactory> HeroFactory;
	private String message;
	private String id;
	private String name;
	private String heroType;
	private Hero heroToAdd;
	private Stack<Command> redoStack;
	private CareTaker careTaker;
	private Player playerToAdd; //store the player that the hero is added to, for undo and redo


	public AddHeroCommand(CurrentPlayerHolder currentPlayerHolder, HashMap<String, HeroFactory> HeroFactory
```

```
                                ,Stack<Command> redoStack, String id, String name,String heroType, CareTaker
careTaker) {
        this.currentPlayerHolder = currentPlayerHolder;
        this.HeroFactory = HeroFactory;
        this.redoStack = redoStack;
        this.id = id;
        this.name = name;
        this.heroType = heroType;
        this.careTaker = careTaker;
    }

    public void execute() {
        heroToAdd = HeroFactory.get(heroType).createHero(id, name);
        currentPlayerHolder.getCurrentPlayer().addHero(heroToAdd);
        playerToAdd = currentPlayerHolder.getCurrentPlayer();
        System.out.println("Hero is added.");
        message = "Add hero, " + heroToAdd.getHeroID() + ", " + heroToAdd.getHeroName() + ", " +
heroToAdd.getClass().getSimpleName();
        redoStack.clear();
        careTaker.clearRedoList();
    }
    public void undo() {
        playerToAdd.removeHero(heroToAdd);
        System.out.println("Command (" + message + ") is undone.");

    }
    public void redo() {
        playerToAdd.addHero(heroToAdd);
        System.out.println("Command (" + message + ") is redone.");
    }

    public String toString(){
        return message;
    }
}
```

CallHeroSkillCommand.java

```
package Command;


import Hero.*;
import Memento.*;
import Player.*;
import java.util.*;


public class CallHeroSkillCommand implements Command {
	private String message;
	private Hero hero;
	private Stack<Command> redoStack;
	private CareTaker careTaker;


	public CallHeroSkillCommand(Hero hero, Stack<Command> redoStack, CareTaker careTaker){
		this.hero = hero;
		this.redoStack = redoStack;
		this.careTaker = careTaker;
	}


	public void execute(){
		careTaker.saveHero(hero);
		hero.callSkill();
		System.out.println(hero.getHeroID() + " " + hero.getHeroName() + "'s attributes are changed to:");
		hero.showHeroStatus();
		redoStack.clear();
		careTaker.clearRedoList();


		if(hero instanceof Warlock){
			message = "Call hero skill, " + hero.getHeroID() + ", " + hero.getHeroName() + ", Warlock, Hp:
"+hero.getHp() + ", " +
				"Damage: "+hero.getDamage() + ", Mp: "+((Warlock) hero).getMp();
		}else if(hero instanceof Warrior){
			message = "Call hero skill, " + hero.getHeroID() + ", " + hero.getHeroName() + ", Warrior, Hp:
"+hero.getHp() + ", " +
				"Damage: "+hero.getDamage() + ", Defence Point: "+((Warrior) hero).getDefencePoint();
		}
```

```
    }
    public void undo(){
        careTaker.saveRedoHero(hero);
        careTaker.undo();
        System.out.println("Command (" + message + ") is undone.");
    }
    public void redo(){
        careTaker.saveHero(hero);
        careTaker.redo();
        System.out.println("Command (" + message + ") is redone.");
    }

    public String toString(){
        return message;
    }
}
```

ChangePlayerNameCommand.java

```
package Command;
import Memento.CareTaker;
import Player.*;
import java.util.*;

public class ChangePlayerNameCommand implements Command {
    private CurrentPlayerHolder currentPlayerHolder;
    private String message;
    private String newName;
    private Stack<Command> redoStack;
    private CareTaker careTaker;

    public ChangePlayerNameCommand(CurrentPlayerHolder currentPlayerHolder, String newName,
Stack<Command> redoStack, CareTaker careTaker){
        this.currentPlayerHolder = currentPlayerHolder;
        this.newName = newName;
        this.redoStack = redoStack;
        this.careTaker = careTaker;
    }
```

```
    public void execute(){
        careTaker.savePlayer(currentPlayerHolder.getCurrentPlayer());
        currentPlayerHolder.getCurrentPlayer().setPlayerName(newName);
        redoStack.clear();
        careTaker.clearRedoList();
        message = "Change player's name, " + currentPlayerHolder.getCurrentPlayer().getPlayerID() + ", " +
newName;
    }
    public void undo(){
        careTaker.saveRedoPlayer(currentPlayerHolder.getCurrentPlayer());
        careTaker.undo();
        System.out.println("Command (" + message + ") is undone.");
        System.out.println("\nFantastic World (FW) \n" +
                "c = create player, g = set current player, a = add hero, m = call hero skill, d \n" +
                "= delete hero, s = show player, p = display all players, t = change player's \n" +
                "name, u = undo, r = redo, l = list undo/redo, x = exit system ");
        if (currentPlayerHolder.getCurrentPlayer() != null) {
            System.out.println("The current player is " + currentPlayerHolder.getCurrentPlayer().getPlayerID() + " " +
                    currentPlayerHolder.getCurrentPlayer().getPlayerName());
        }
    }
    public void redo(){
        careTaker.savePlayer(currentPlayerHolder.getCurrentPlayer());
        careTaker.redo();
        System.out.println("Command (" + message + ") is redone.");
    }

    public String toString(){
        return message;
    }
}
```

CreatePlayerCommand.java

```
package Command;

import Memento.*;
import Player.*;
```

```
import PlayerFactory.*;

import java.util.*;

public class CreatePlayerCommand implements Command {
    private UserFactory pf;
    private CurrentPlayerHolder currentPlayerHolder;
    private Vector<Player> playerVector;
    private Stack<Command> redoStack; // Stack to store all commands to be redo
    private Player p;
    private String id;
    private String name;
    private String message;
    private CareTaker careTaker;

    public CreatePlayerCommand(UserFactory pf, CurrentPlayerHolder currentPlayerHolder,
                               Vector<Player> playerVector, Stack<Command> redoStack,
                               String id, String name, CareTaker careTaker) {
        this.pf = pf;
        this.currentPlayerHolder = currentPlayerHolder;
        this.playerVector = playerVector;
        this.redoStack = redoStack;
        this.id = id;
        this.name = name;
        this.careTaker = careTaker;
    }

    public void execute() {
        p = pf.createPlayer(id,name);
        playerVector.add(this.p);
        System.out.println("Player " + p.getPlayerName() + " is created.");
        currentPlayerHolder.setCurrentPlayer(p);
        System.out.println("Current player is changed to " + p.getPlayerID());
        message = "Create player, " + p.getPlayerID() + ", " + p.getPlayerName();
        redoStack.clear();
        careTaker.clearRedoList();
    }

    public void undo() {
```

```
            playerVector.remove(this.p);
            System.out.println("Command (" + message + ") is undone.");
            if (!playerVector.isEmpty()) {
                currentPlayerHolder.setCurrentPlayer(playerVector.get(0));
                System.out.println("Current player is changed to " +
currentPlayerHolder.getCurrentPlayer().getPlayerID());
            } else {
                currentPlayerHolder.setCurrentPlayer(null);
                System.out.println("No current player now");
            }
        }

        public void redo() {
            playerVector.add(this.p);
            currentPlayerHolder.setCurrentPlayer(this.p);
            System.out.println("Command (" + message + ") is redone.");
            System.out.println("The current player is changed to " + currentPlayerHolder.getCurrentPlayer().getPlayerID()
+ ".");
        }

        public String toString(){
            return message;
        }
}
```

DeleteHeroCommand.java

```
package Command;

import Hero.*;
import Memento.*;
import Player.*;
import java.util.*;

public class DeleteHeroCommand implements Command {
        private CurrentPlayerHolder currentPlayerHolder;
        private String message;
        private Hero heroToDelete;
```

```
    private Stack<Command> redoStack;
    private CareTaker careTaker;
    private Player playerToDelete;

    public DeleteHeroCommand(CurrentPlayerHolder currentPlayerHolder, Hero heroToDelete, Stack<Command>
redoStack,
                                        CareTaker careTaker){
        this.currentPlayerHolder = currentPlayerHolder;
        this.heroToDelete = heroToDelete;
        this.redoStack = redoStack;
        this.careTaker = careTaker;

    }

    public void execute(){
        System.out.println(heroToDelete.getHeroID() + " " + heroToDelete.getHeroName() + " is deleted.");
        currentPlayerHolder.getCurrentPlayer().removeHero(heroToDelete);
        message = "Delete hero, " + heroToDelete.getHeroID();
        playerToDelete = currentPlayerHolder.getCurrentPlayer();
        redoStack.clear();
        careTaker.clearRedoList();
    }

    public void undo(){
        playerToDelete.addHero(heroToDelete);
        System.out.println("Command (" + message + ") is undone.");
    }
    public void redo(){
        playerToDelete.removeHero(heroToDelete);
        System.out.println("Command (" + message + ") is redone.");
    }
    public String toString(){
        return message;
    }
}
```

DisplayAllPlayerCommand.java

```
package Command;
```

```
import Player.*;
import java.util.*;

public class DisplayAllPlayerCommand implements Command {
	private Vector<Player> playerVector;
	public DisplayAllPlayerCommand(Vector<Player> playerVector) {
		this.playerVector = playerVector;
	}
	public void execute(){
		if (playerVector.size() > 0) {
			for (int i = 0; i < playerVector.size(); i++) {
				System.out.println("Player " + playerVector.get(i).getPlayerName() + " (" +
playerVector.get(i).getPlayerID() + ")");
			}
		} else {
			System.out.println("No player to show");
		}
	}
	public void undo(){
		//no need implementation
	}
	public void redo(){
		//no need implementation
	}

	public String toString(){
		return "";
	}
}
```

ExitCommand.java

```
package Command;

public class ExitCommand implements Command {
	public void execute() {
		System.exit(0);
	}
```

```
    public void undo(){
        //no need implementation
    }
    public void redo(){
        //no need implementation
    }

    public String toString(){
        return "";
    }
}
```

RedoCommand.java

```
package Command;
import java.util.*;

public class RedoCommand implements Command {
    private Stack<Command> commandStack;
    private Stack<Command> redoStack;



    public RedoCommand(Stack<Command> commandStack, Stack<Command> redoStack) {
        this.commandStack = commandStack;
        this.redoStack = redoStack;

    }

    public void execute() {
        if(!redoStack.isEmpty()) {
            Command command = redoStack.pop();
            command.redo();
            commandStack.push(command);

        }else{
            System.out.println("Nothing to redo");
```

```
        }
    }
    public void undo() {
        //no need implementation
    }
    public void redo() {
        //no need implementation
    }
    public String toString(){
        return "";
    }
}
```

SetCurrentPlayerCommand.java

```
package Command;

import Player.*;

import java.util.*;

public class SetCurrentPlayerCommand implements Command{
    private Vector<Player> playerVector;
    private Scanner sc;
    private CurrentPlayerHolder currentPlayerHolder;

    public SetCurrentPlayerCommand(Scanner sc, Vector<Player> playerVector, CurrentPlayerHolder
currentPlayerHolder) {
        this.playerVector = playerVector;
        this.sc = sc;
        this.currentPlayerHolder = currentPlayerHolder;
    }

    public void execute(){
        if (playerVector.size() > 0) {
            System.out.print("Please input player ID:- ");
            String id = sc.nextLine();
            for (int i = 0; i < playerVector.size(); i++) {
```

```
                    if (playerVector.get(i).getPlayerID().equals(id)) {
                        currentPlayerHolder.setCurrentPlayer(playerVector.get(i));
                        System.out.println("Changed current player to " +
currentPlayerHolder.getCurrentPlayer().getPlayerID());
                        break;
                    }

                    if (i == playerVector.size() - 1) {
                        System.out.println("Player " + id + " is not found!!");
                    }
                }
            } else {
                System.out.println("No player available");
            }
        }

        public void undo(){
            //no need implementation
        }

        public void redo(){
            //no need implementation
        }
        public String toString(){
            return "";
        }
}
```

ShowPlayerCommand.java

```
package Command;

import Player.*;

public class ShowPlayerCommand implements Command {
        private CurrentPlayerHolder currentPlayerHolder;

        public ShowPlayerCommand(CurrentPlayerHolder currentPlayerHolder) {
```

```
        this.currentPlayerHolder = currentPlayerHolder;
    }

    public void execute() {
        if (currentPlayerHolder.getCurrentPlayer() != null) {
            currentPlayerHolder.getCurrentPlayer().showPlayerDetails();
        } else {
            System.out.println("No player to show");
        }
    }

    public void undo() {
        //no need implementation
    }

    public void redo() {
        //no need implementation
    }

    public String toString(){
        return "";
    }
}
```

ShowUndoRedoCommand.java

```
package Command;

import java.util.*;

public class ShowUndoRedoCommand implements Command {
    private Stack<Command> commandStack;
    private Stack<Command> redoStack;

    public ShowUndoRedoCommand(Stack<Command> commandStack, Stack<Command> redoStack) {
        this.commandStack = commandStack;
        this.redoStack = redoStack;
```

```
    }

    public void execute() {
        System.out.println("Undo List");
        for (int i = commandStack.size() - 1; i >= 0; i--) {
            System.out.println(commandStack.get(i));
        }
        System.out.println("-- End of undo list --");
        System.out.println("Redo List");
        for (int i = redoStack.size() - 1; i >= 0; i--) {
            System.out.println(redoStack.get(i));
        }
        System.out.println("-- End of redo list --");
    }
    public void undo() {
        //no need implementation
    }
    public void redo() {
        //no need implementation
    }
    public String toString(){
        return "";
    }
}
```

UndoCommand.java

```
package Command;
import java.util.*;

public class UndoCommand implements Command {
    private Stack<Command> commandStack;
    private Stack<Command> redoStack;



    public UndoCommand(Stack<Command> commandStack, Stack<Command> redoStack) {
        this.commandStack = commandStack;
        this.redoStack = redoStack;
```

```
    }

    public void execute() {
        if(!commandStack.isEmpty()){
            Command command = commandStack.pop();
            command.undo();
            redoStack.push(command);
        }else{
            System.out.println("Nothing to undo");
        }
    }
    public void undo() {
        //no need implementation
    }
    public void redo() {
        //no need implementation
    }

    public String toString(){
        return "";
    }
}
```

UndoCommandFactory Package

CommandFactory.java

```
package CommandFactory;
import Command.*;

public interface CommandFactory {
	public Command createCommand();
}
```

AddHeroCommandFactory.java

```
package CommandFactory;

import Command.*;
import Memento.*;
import Player.*;
import HeroFactory.*;
import Hero.*;

import java.util.*;

public class AddHeroCommandFactory implements CommandFactory {
	private Scanner sc;
	private CurrentPlayerHolder currentPlayerHolder;
	private String id;
	private String name;
	private HashMap<String, HeroFactory> HeroFactory;
	private Stack<Command> commandStack;
	private Stack<Command> redoStack;
	private Vector<Player> playerVector;
	private CareTaker careTaker;

	public AddHeroCommandFactory(Scanner sc, CurrentPlayerHolder currentPlayerHolder, HashMap<String,
HeroFactory> HeroFactory,
                                     Stack<Command> commandStack, Stack<Command> redoStack,
```

```
                                        Vector<Player> playerVector, CareTaker careTaker) {
        this.sc = sc;
        this.currentPlayerHolder = currentPlayerHolder;
        this.HeroFactory = HeroFactory;
        this.commandStack = commandStack;
        this.redoStack = redoStack;
        this.playerVector = playerVector;
        this.careTaker = careTaker;
    }

    public Command createCommand() {
        if (currentPlayerHolder.getCurrentPlayer() != null) {
            //check hero id and name input below, if invalid, ask again until correct, avoid creating command with
wrong input
            while (true) {
                try {
                    System.out.print("Please input hero information (id, name):- ");
                    String idName = sc.nextLine();
                    String[] split = idName.split(", ");
                    id = split[0];
                    name = split[1];
                    //check existing hero id for current player
                    boolean sameHeroID = false;
                    for (Hero h : currentPlayerHolder.getCurrentPlayer().getHeroes()) {
                        if (h.getHeroID().equals(id)) {
                            System.out.println("Hero ID already exist");
                            sameHeroID = true;
                            break;
                        }
                    }
                    if (sameHeroID){
                        continue;
                    }
                    break;
                } catch (Exception e) {
                    System.out.println("Invalid input");
                }
            }
```

```
                //check hero type input below, if invalid, ask again until correct, avoid creating command with wrong
input
                while (true) {
                    System.out.print("Hero Type (1 = Warrior | 2 = Warlock ):- ");
                    String heroType = sc.nextLine();
                    if (HeroFactory.get(heroType) != null) {
                        Command c = new AddHeroCommand(currentPlayerHolder, HeroFactory, redoStack, id, name,
heroType,careTaker);
                        commandStack.push(c);
                        return c;
                    } else {
                        System.out.println("Invalid hero type");
                    }
                }
            } else {
                System.out.println("No player to add hero");
                //return a command with no undo/redo to avoid error
                return new DisplayAllPlayerCommand(playerVector);
            }
        }
}
```

CallHeroSkillCommandFactory.java

```
package CommandFactory;
import Command.*;
import Hero.*;
import Player.*;
import Memento.*;
import java.util.*;

public class CallHeroSkillCommandFactory implements CommandFactory {
    private CurrentPlayerHolder currentPlayerHolder;
    private Scanner sc;
    private Stack<Command> commandStack;
    private Stack<Command> redoStack;
    private Vector<Player> playerVector;
    private CareTaker careTaker;
```

```
public CallHeroSkillCommandFactory(CurrentPlayerHolder currentPlayerHolder, Scanner sc,
                                   Stack<Command> commandStack, Stack<Command> redoStack,
                                   Vector<Player> playerVector, CareTaker careTaker) {
    this.currentPlayerHolder = currentPlayerHolder;
    this.sc = sc;
    this.commandStack = commandStack;
    this.redoStack = redoStack;
    this.playerVector = playerVector;
    this.careTaker = careTaker;
}

public Command createCommand() {
    if (currentPlayerHolder.getCurrentPlayer() != null) {
        if (currentPlayerHolder.getCurrentPlayer().getHeroes().size() > 0) {
            System.out.print("Please input hero ID:- ");
            String heroID = sc.nextLine();
            for (int i = 0; i < currentPlayerHolder.getCurrentPlayer().getHeroes().size(); i++) {
                Hero hero = currentPlayerHolder.getCurrentPlayer().getHeroes().get(i);
                if (hero.getHeroID().equals(heroID)) {
                    Command c;
                    if(hero instanceof Warrior){
                        if (((Warrior) hero).getDefencePoint() < 0) {
                            System.out.println("Defence point is less than 0, cannot call skill");
                            return new ShowPlayerCommand(currentPlayerHolder);
                        }else{
                            c =    new CallHeroSkillCommand(hero,redoStack,careTaker);
                            commandStack.push(c);
                            return c;
                        }
                    } else if (hero instanceof Warlock){
                        if (((Warlock) hero).getMp() <= 0) {
                            System.out.println("Mp is less than or equal to 0, cannot call skill");
                            return new ShowPlayerCommand(currentPlayerHolder);
                        }else{
                            c =    new CallHeroSkillCommand(hero,redoStack,careTaker);
                            commandStack.push(c);
                            return c;
                        }
```

```
                        }
                    }
                }
                //if hero not found, the following three code run, telling user what hero they have
                System.out.println("Hero not found");
                System.out.println();
                return new ShowPlayerCommand(currentPlayerHolder);

            }else{
                System.out.println("No hero available");
                //because no hero for current player, show the user they have no hero
                System.out.println();
                return new ShowPlayerCommand(currentPlayerHolder);
            }
        }else{
            System.out.println("No player to call hero skills");
            //return a command with no undo/redo to avoid error
            return new DisplayAllPlayerCommand(playerVector);
        }
    }
}
```

ChangePlayerNameCommandFactory.java

```
package CommandFactory;
import Memento.*;
import Player.*;
import Command.*;
import java.util.*;
public class ChangePlayerNameCommandFactory implements CommandFactory {
    private CurrentPlayerHolder currentPlayerHolder;
    private Scanner sc;
    private Vector<Player> playerVector;
    private Stack<Command> commandStack;
    private Stack<Command> redoStack;
    private CareTaker careTaker;
```

```
    public ChangePlayerNameCommandFactory(CurrentPlayerHolder currentPlayerHolder, Scanner sc, Vector<Player>
playerVector,
                                          Stack<Command> commandStack, Stack<Command> redoStack,
CareTaker careTaker) {
        this.currentPlayerHolder = currentPlayerHolder;
        this.sc = sc;
        this.playerVector = playerVector;
        this.commandStack = commandStack;
        this.redoStack = redoStack;
        this.careTaker = careTaker;
    }

    public Command createCommand() {
        if (currentPlayerHolder.getCurrentPlayer() != null) {
            System.out.print("Please input new name of the current player:- ");
            String newName = sc.nextLine();
            System.out.println("Player's name is updated.");
            Command c = new ChangePlayerNameCommand(currentPlayerHolder, newName, redoStack, careTaker);
            commandStack.push(c);
            return c;
        }else{
            System.out.println("No player to change name");
            //return a command with no undo/redo to avoid error
            return new DisplayAllPlayerCommand(playerVector);
        }
    }
}
```

CreatePlayerCommandFactory.java

```
package CommandFactory;
import Command.*;
import Memento.*;
import Player.*;
import Player.Player;
import PlayerFactory.*;

import java.util.Scanner;
```

```
import java.util.Stack;
import java.util.Vector;

public class CreatePlayerCommandFactory implements CommandFactory {

    private Scanner sc;
    private UserFactory pf;
    private CurrentPlayerHolder currentPlayerHolder;
    private Vector<Player> playerVector;
    private Stack<Command> redoStack;// Stack to store all commands to be redo
    private Stack<Command> commandStack;    // Stack to store executed commands
    private CareTaker careTaker;

    public CreatePlayerCommandFactory(Scanner sc, UserFactory pf, CurrentPlayerHolder currentPlayerHolder,
                                      Vector<Player> playerVector, Stack<Command> redoStack,
Stack<Command> commandStack,
                                      CareTaker careTaker) {
        this.sc = sc;
        this.pf = pf;
        this.currentPlayerHolder = currentPlayerHolder;
        this.playerVector = playerVector;
        this.redoStack = redoStack;
        this.commandStack = commandStack;
        this.careTaker = careTaker;
    }

    public Command createCommand() {
        String id;
        try {
            while (true){
                System.out.print("Player ID:- ");
                id = sc.nextLine();
                if (id.equals("")) {
                    System.out.println("Player ID cannot be empty.");
                }
                else{
                    break;
                }
            }
```

```
                for (int i = 0; i < playerVector.size(); i++) {
                    if (playerVector.get(i).getPlayerID().equals(id)) {
                        throw new Exception();
                    }
                }
            } catch (Exception e) {
                System.out.println("Player ID already exists");
                System.out.println("Existing player ID:- ");
                return new DisplayAllPlayerCommand(playerVector);
            }

            String name="";
            while(true){
                System.out.print("Player Name:- ");
                name = sc.nextLine();
                if(name.equals("")) {
                    System.out.println("Player name cannot be empty.");
                    continue;
                }else{
                    break;
                }
            }

            Command c = new CreatePlayerCommand(pf, currentPlayerHolder, playerVector,
redoStack,id,name,careTaker);
            commandStack.push(c);
            return c;
        }
}
```

DeleteHeroCommandFactory.java

```
package CommandFactory;

import Command.*;
import Hero.Hero;
import Memento.CareTaker;
import Player.*;
```

```
import java.util.*;

public class DeleteHeroCommandFactory implements CommandFactory {
	private Scanner sc;
	private CurrentPlayerHolder currentPlayerHolder;
	private Vector<Player> playerVector;
	private Stack<Command> commandStack;
	private Stack<Command> redoStack;
	private CareTaker careTaker;

	public DeleteHeroCommandFactory(Scanner sc, CurrentPlayerHolder currentPlayerHolder, Vector<Player>
playerVector,
					Stack<Command> commandStack, Stack<Command> redoStack,
CareTaker careTaker) {
		this.sc = sc;
		this.currentPlayerHolder = currentPlayerHolder;
		this.playerVector = playerVector;
		this.commandStack = commandStack;
		this.redoStack = redoStack;
		this.careTaker = careTaker;
	}

	public Command createCommand() {
		if (currentPlayerHolder.getCurrentPlayer() != null) {
			if (currentPlayerHolder.getCurrentPlayer().getHeroes().size() > 0) {
				System.out.print("Please input hero ID:- ");
				String heroID = sc.nextLine();
				for (int i = 0; i < currentPlayerHolder.getCurrentPlayer().getHeroes().size(); i++) {
					Hero hero = currentPlayerHolder.getCurrentPlayer().getHeroes().get(i);
					if (hero.getHeroID().equals(heroID)) {
						Command c =    new DeleteHeroCommand(currentPlayerHolder,
hero,redoStack,careTaker);
						commandStack.push(c);
						return c;
					}
				}
				//if hero not found, the following three code run, telling user what hero they have
				System.out.println("Hero not found");
```

```
                System.out.println();
                return new ShowPlayerCommand(currentPlayerHolder);

            } else {
                System.out.println("No hero available");
                //because no hero for current player, show the user they have no hero
                System.out.println();
                return new ShowPlayerCommand(currentPlayerHolder);
            }
        } else {
            System.out.println("No current player");
            //return a command with no undo/redo to avoid error
            return new DisplayAllPlayerCommand(playerVector);
        }
    }
}
```

DisplayAllPlayerCommandFactory.java

```
package CommandFactory;
import Command.*;
import Player.*;
import java.util.*;

public class DisplayAllPlayerCommandFactory implements CommandFactory {
    private Vector<Player> playerVector;

    public DisplayAllPlayerCommandFactory(Vector<Player> playerVector) {
        this.playerVector = playerVector;
    }
    public Command createCommand() {
        return new DisplayAllPlayerCommand(playerVector);
    }
}
```

ExitCommandFactory.java

```
package CommandFactory;
import Command.*;
public class ExitCommandFactory implements CommandFactory {
	public Command createCommand() {
		return new ExitCommand();
	}
}
```

RedoCommandFactory.java

```
package CommandFactory;
import Command.*;
import java.util.*;
public class RedoCommandFactory implements CommandFactory {
	private Stack<Command> commandStack;
	private Stack<Command> redoStack;

	public RedoCommandFactory(Stack<Command> commandStack, Stack<Command> redoStack) {
		this.commandStack = commandStack;
		this.redoStack = redoStack;
	}


	public Command createCommand() {
		return new RedoCommand(commandStack, redoStack);
	}
}
```

SetCurrentPlayerCommandFactory.java

```
package CommandFactory;
import Command.*;
import Player.*;
import java.util.*;

public class SetCurrentPlayerCommandFactory implements CommandFactory {
```

```
	private Scanner sc;
	private Vector<Player> playerVector;
	private CurrentPlayerHolder currentPlayerHolder;

	public SetCurrentPlayerCommandFactory(Scanner sc, Vector<Player> playerVector, CurrentPlayerHolder
currentPlayerHolder) {
		this.sc = sc;
		this.playerVector = playerVector;
		this.currentPlayerHolder = currentPlayerHolder;
	}
	public Command createCommand() {
		return new SetCurrentPlayerCommand(sc,playerVector, currentPlayerHolder);
	}
}
```

ShowPlayerCommandFactory.java

```
package CommandFactory;
import Command.*;
import Player.*;
public class ShowPlayerCommandFactory implements CommandFactory {
	private CurrentPlayerHolder currentPlayerHolder;

	public ShowPlayerCommandFactory(CurrentPlayerHolder currentPlayerHolder) {
		this.currentPlayerHolder = currentPlayerHolder;
	}

	public Command createCommand() {
		return new ShowPlayerCommand(currentPlayerHolder);
	}
}
```

ShowUndoRedoCommandFactory.java

```
package CommandFactory;
import Command.*;
import java.util.*;
```

```
public class ShowUndoRedoCommandFactory implements CommandFactory {
	private Stack<Command> commandStack;
	private Stack<Command> redoStack;

	public ShowUndoRedoCommandFactory(Stack<Command> commandStack, Stack<Command> redoStack) {
		this.commandStack = commandStack;
		this.redoStack = redoStack;
	}
	public Command createCommand() {
		return new ShowUndoRedoCommand(commandStack, redoStack);
	}
}
```

UndoCommandFactory.java

```
package CommandFactory;
import Command.*;
import java.util.*;

public class UndoCommandFactory implements CommandFactory {
	private Stack<Command> commandStack;
	private Stack<Command> redoStack;

	public UndoCommandFactory(Stack<Command> commandStack, Stack<Command> redoStack) {
		this.commandStack = commandStack;
		this.redoStack = redoStack;
	}

	public Command createCommand() {
		return new UndoCommand(commandStack, redoStack);
	}
}
```

Memento Package

Memento.java

```
package Memento;


public interface Memento {
	public void restore();
}
```

PlayerMemento.java

```
package Memento;


import Hero.*;
import Player.*;
import java.util.*;


public class PlayerMemento implements Memento {
	private String playerID;
	private String playerName;
	private Vector<Hero> heroes;
	private Player player;

	public PlayerMemento(Player player) {
		this.player = player;
		this.playerID = player.getPlayerID();
		this.playerName = player.getPlayerName();
		this.heroes = player.getHeroes();
	}

	public void restore() {
		player.setPlayerName(playerName);
	}
}
```

HeroMemento.java

```
package Memento;
import Hero.*;

public class HeroMemento implements Memento {
	private String heroName;
	private int hp;
	private int damage;
	private int mp;
	private int defencePoint;
	private Hero hero;

	public HeroMemento(Hero hero) {
		this.hero = hero;
		this.heroName = hero.getHeroName();
		this.hp = hero.getHp();
		this.damage = hero.getDamage();
		String heroType = hero.getClass().getSimpleName();
		System.out.println(heroType);
		if (hero instanceof Warlock) {
			this.mp = ((Warlock) hero).getMp();
		}else if(hero instanceof Warrior){
			this.defencePoint = ((Warrior) hero).getDefencePoint();
		}
	}

	public void restore() {
		hero.setHeroName(heroName);
		hero.setHp(hp);
		hero.setDamage(damage);
		if (hero instanceof Warlock) {
			((Warlock) hero).setMp(mp);
		}else if(hero instanceof Warrior){
			((Warrior) hero).setDefencePoint(defencePoint);
		}
	}
}
```

CareTaker.java

```
package Memento;
import Hero.*;
import Player.Player;

import java.util.*;

public class CareTaker {
	Stack<Memento> undoList;	//stack to store memento to be undone
	Stack <Memento> redoList; //stack to store memento to be redone

	public CareTaker(){
		undoList = new Stack<Memento>();
		redoList = new Stack<Memento>();
	}

	public void saveHero(Hero hero){
		undoList.push(new HeroMemento(hero));
	}

	public void savePlayer(Player player){
		undoList.push(new PlayerMemento(player));
	}

	public void undo(){
		undoList.pop().restore();
	}

	public void redo(){
		redoList.pop().restore();
	}

	public void saveRedoHero(Hero hero){
		redoList.push(new HeroMemento(hero));
	}

	public void saveRedoPlayer(Player player){
```

```
            redoList.push(new PlayerMemento(player));
        }

        public int getUndoListSize(){
            return undoList.size();
        }

        public int getRedoListSize(){
            return redoList.size();
        }

        public void clearRedoList(){
            redoList.clear();
        }
}
```