Program report(world of zuul text-based game)

The purpose of the game:

This gaming program I did is a simple text-based game that player could have an experience on entering different caves and fighting with monsters. Player will enjoy on killing monster, earning gold and updating attack or defence rate. At the beginning, as a player, player will start player gaming progress from outside the cave entrance without fighting with monster. After that section, player have several options that player can choose which cave player prefer to get in. There will be a random weight monster in every caves include the outside when player return from any other caves. Different weight monster will deal different damage to player based on their weight. Player also have several tool options during the fight. The tool deal more damage on monster will also have larger damage to player as well. Meanwhile, player could also gain gold after killing the monster. Gold could be used in shop(input "shop"), attack(deal more damage to monster), defence(monster deal less damage to player) and blood(add playerr current blood) tool are available for player.

Classes review:

There are 6 classes have been added to the work.

Blood: Return the player's blood.

Gold: Return gold that player has.

itemsRate: Initialize and build setter and getter of attack rate and defence rate(could be bought in shop).

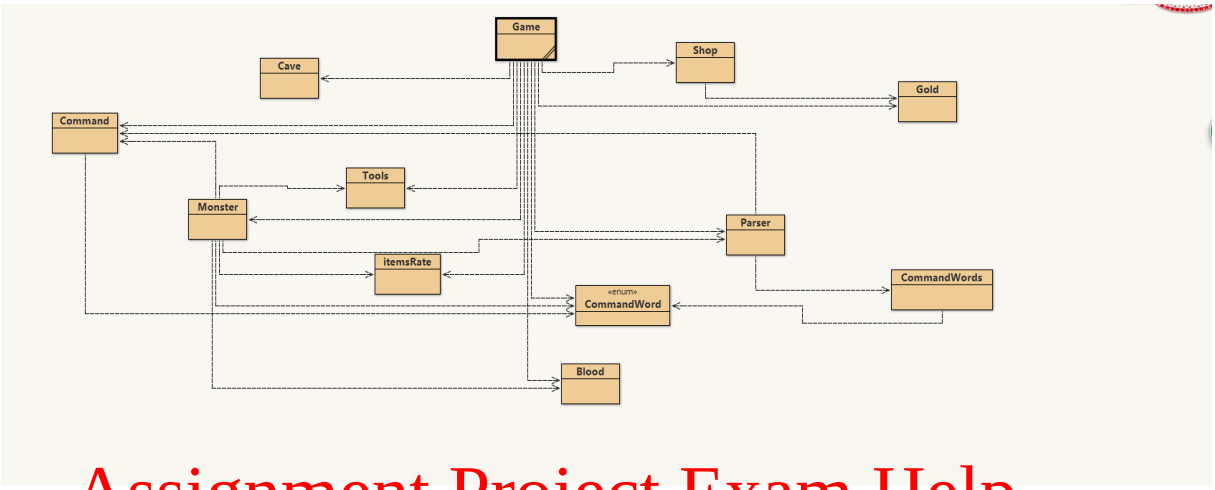
Tools: Initialize and generate setter and getter of three different tools' rate and description. ("start", "intermediate", "upper") .

Shop: Return text after buying attack tool or defence tool in shop. Print text after buying blood tool.

Monster: Build an object which is "monster", with random weight between 0-20(include 0). Three tools have been added to this class. Calculation and fighting rules have been added as well in this class. This class also has receiving function of second command words which is the keyword of executing order

Other classes: Other classes from start code have not been modified too much. However some functions in original code have not been used in the game, for example the description of room. Hopefully, these functions will be used in next version of the game.

Classes diagram(screenshot):



Running progress of gaming program

Print welcome notification:


```
Welcome to Monster game!
There are three caves infront you, in the cave there is a monster, you need use tools to kill it
Please choose a tool to kill monster (start, intermediate, upper)
Type 'help' if you need help.

You are outside the entrance of caves.
Exits: east south west
Now, you have 50 blood
>
```

Input go east, meet with first monster. Print reminder of choosing tool to attack monster. (Choose + tool name)

```
You are outside the entrance of caves.
Exits: east south west
Now, you have 50 blood
> go east
You are in cave1.
Exits: west
Now, the tools in yours hand is start tool , and rate 1
This monster has  10 blood
please choose a tool to attack
> >
```

Print reward player get from killing the monster.

```
I don't know what you mean...
please choose a tool to attack
> >  5 blood
You have 42 blood
please choose a tool to attack
> > I don't know what you mean...
please choose a tool to attack
> > choose upper tool
Now, the tools in yours hand is upper , and rate 5
The monster has 0 blood
You have 37 blood
Congratulation, you kill a monster, you win 10 gold! You totally have 10 gold
>
```


Input "go west" return to outside, monster will automatically generated.

```
Congratulation, you kill a monster, you win 10 gold! You totally have 10 gold
> go west
You are outside the entrance of caves.
Exits: east south west
Now, the tools in yours hand is start tool , and rate 1
This monster has  6 blood
please choose a tool to attack
> >
```

Input "shop" to enter shop function. Then input "shop" +"(the function player want to buy)" to buy any tools.

```
> shop
shop what?? (attack, defence,blood)
Now, the tools in yours hand is start tool , and rate 1
> shop attack
You buy a attack tool cost 10 gold, you have 10
Now, the tools in yours hand is start tool , and rate 1
> shop blood
You have successfully added 100 blood
You buy a blood tool cost 10 gold, you have 0
Now, the tools in yours hand is start tool , and rate 1
```

While killed 4 monsters, player win the game, system print victory notification and quit program.

```
Congratulation, you kill a monster, you win 10 gold! You totally have 20 gold
Congratulation, you win!!!!
```

Input "help" to print keywords.

```
> help
help  shop  go  quit  choose
Now, the tools in yours hand is start tool , and rate 1
```

Input "quit" quit game. Print quitting text.

```
> quit
Now, the tools in yours hand is start tool , and rate 1
Thank you for playing.  Good bye.
```

Current status:

Program now could be run smoothly. The logic of gaming progress is clear and fundamental factors of game have been built up successfully. Enjoyability, factors and complexity of this game is acceptable as a small starter text-based game. However, victory condition is too easy for player to achieve. According to limiting knowledge and understanding of object oriented programming, I was stuck in adding a special cave and a final boss to the game. In the next version, hopefully I will find a new way to add these factors and change winning condition to kill the final boss. In the other hand, the principles and ideology have not fully implemented in this course work due to the limitation of the knowledge and practice experience on java programming. The concept of "object oriented" has not been applied in the program perfectly at this version. I may have a misunderstanding between "object oriented" and "Procedure Oriented". I will try to solve it out before next version. According to textbook and online resources, the idea is "interface" may be able to be used in next version.

Next:

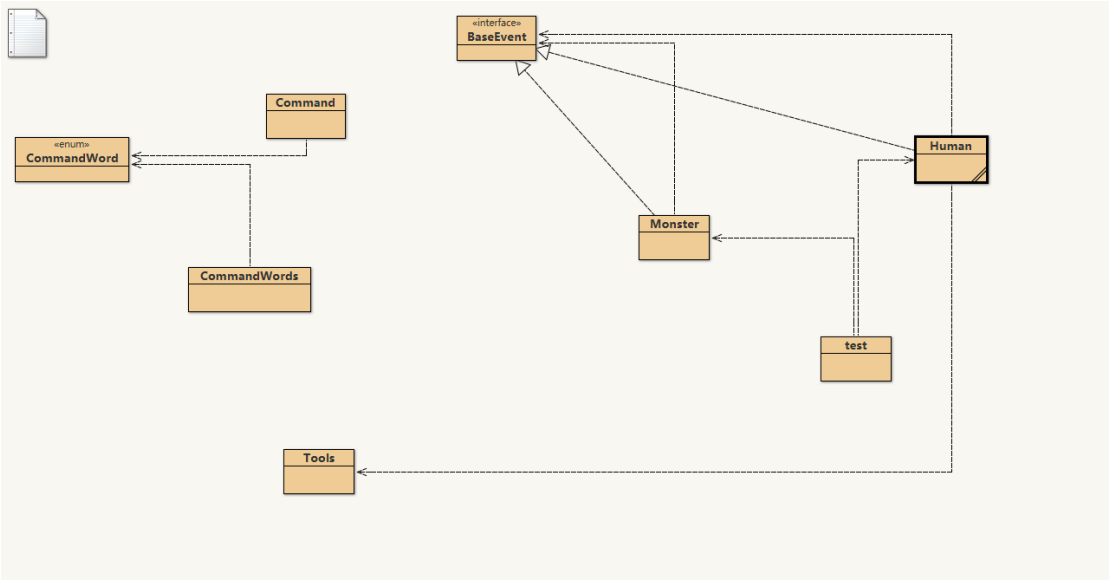
I have already started to create a new version of writing the code. "Object oriented" principle and interface should be used in new version.

The screenshot in appendix is several classes what I have done so far.

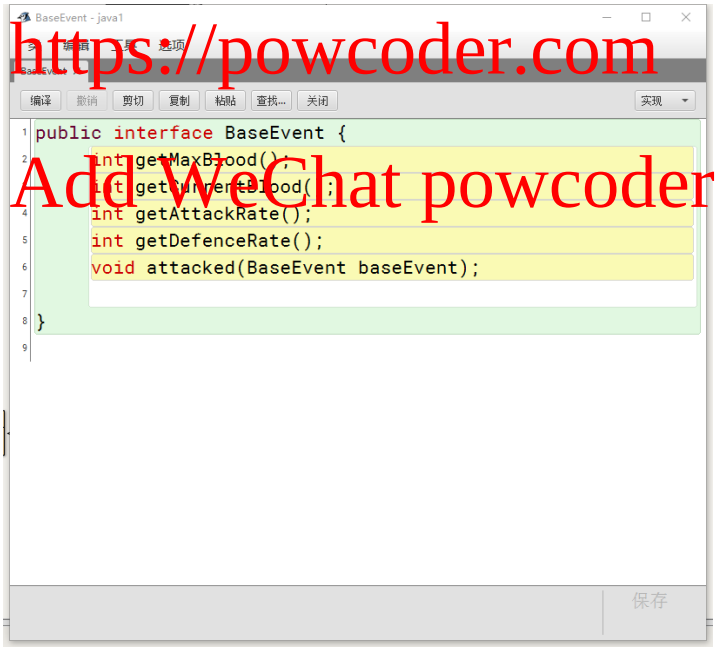
Appendix

Classes diagram:

Here is part of work screenshots of the works using interface and object oriented ideology should be fully applied（Unfinished）:



Class BaseEvent(interface):



Class Human:

```java
import java.util.ArrayList;


public class Human implements BaseEvent {
    private int maxBlood;
    private int currentBlood;
    private int attachRate;
    private int defenceRate;
    private ArrayList<Tools> toolsList;
    private Tools currentTool;

    public int getMaxBlood() {
        return maxBlood;
    }

    public int getCurrentBlood() {
        return currentBlood;
    }

    public int getAttackRate() {
        if(currentTool==null){
            return attachRate;
        }else{
            return currentTool.getRate()*(attachRate+1);
        }
    }

    public int getDefenceRate() {
        return defenceRate;
    }

    public Tools getCurrentTool() {
        return currentTool;
    }

    public void setCurrentTool(String toolStr) {
        Tools tools=null;
        for(Tools item :toolsList){
            if(item.getDescription().equalsIgnoreCase(toolStr)){
                tools=item;
            }
        }
        this.currentTool = tools;
    }

    public ArrayList<Tools> getToolsList() {
        return toolsList;
    }

    public void attacked(BaseEvent baseEvent) {
        int lossBlood=baseEvent.getAttackRate()-getDefenceRate();
        if(lossBlood>0){
            if(lossBlood>this.currentBlood){
                this.currentBlood=0;
                System.out.println("You has  "+this.currentBlood+" blood,dead");
            }else{
                this.currentBlood=this.currentBlood-lossBlood;
                System.out.println("You has  "+this.currentBlood+" blood.");
            }
        }
    }

    public Human() {
        this.maxBlood= 80;
        this.currentBlood = this.maxBlood;
        this.attachRate=10;
        this.defenceRate=10;
        this.toolsList=new ArrayList<Tools>();
        this.toolsList.add(new Tools("start", 1));
        this.toolsList.add(new Tools("intermediate", 2));
        this.toolsList.add(new Tools("upper", 5));
        System.out.println("you has  "+this.maxBlood+" blood"+toString());
    }
}
```

Class Monster:

```
public class Tools {
    private String description;
    private int rate;

    public Tools(String description, int rate) {
        this.description = description;
        this.rate = rate;
    }

    public String getDescription() {
        return description;
    }
    public void setDescription(String description) {
        this.description = description;
    }
    public int getRate() {
        return rate;
    }
    public void setRate(int rate) {
        this.rate = rate;
    }
```

Class Tools:

```
import java.util.Random;

public class Monster implements BaseEvent{
    private int maxBlood;
    private int currentBlood;
    private int attachRate;
    private int defenceRate;

    public int getMaxBlood() {
        return maxBlood;
    }

    public int getCurrentBlood() {
        return currentBlood;
    }

    public int getAttackRate() {
        return attachRate;
    }

    public int getDefenceRate() {
        return defenceRate;
    }


    public void attacked(BaseEvent baseEvent) {
        int lossBlood=baseEvent.getAttackRate()-getDefenceRate();
        if(lossBlood>0){
            if(lossBlood>this.currentBlood){
                this.currentBlood=0;
                System.out.println("This monster has  "+this.currentBlood+" blood,dead");
            }else{
                this.currentBlood=this.currentBlood-lossBlood;
                System.out.println("This monster has  "+this.currentBlood+" blood.");
            }
        }
    }


    public Monster() {
        this.maxBlood= new Random().nextInt(20);
        this.currentBlood = this.maxBlood;
        this.attachRate=2;
        this.defenceRate=2;
        System.out.println("This monster has  "+this.maxBlood+" blood"+toString());
    }

    public Monster(int Blood, int attachRate, int defenceRate) {
        this.maxBlood = Blood;
        this.currentBlood = Blood;
        this.attachRate = attachRate;
        this.defenceRate = defenceRate;
    }

    @Override
    public String toString() {
        return ", attachRate=" + attachRate +
                ", defenceRate=" + defenceRate ;
    }
}
```