Lab 3

	Student Name	Student CSUSM ID	Contribution percentage
1	EJ Lilagan	lilag002	*33.333*
2	Dalynna Nguyen	nguye1051	*33.333*
3	Neo Argatides	argat001	*33.333*

100

Grading Rubrics (for instructor only):

Criteria	1. Beginning	2. Developing	3. Proficient	4. Exemplary
Modeling	0-14	15-19	20-24	25-30
Program: functionality ***correctness***	0-9	10-14	15-19	20
Program: functionality ***Behavior Testing***	0-9	10-14	15-19	20
Program: quality -> ***Readability***	0-2	3-5	6-9	10
Program: quality -> ***Modularity***	0-2	3-5	6-9	10
Program: quality -> ***Simplicity***	0-2	3-5	6-9	10
Total Grade (100)				

Problems:

A video game has three modes: beginner, intermediate and advanced. For each mode chosen by a player, the game GUI shows two control objects: a character selection panel and a weapon selection panel. Note that (a) under different modes the system displays different character selection panels and weapon selection panels, and (b) it is possible that new modes and/or new control objects may be added in the future.

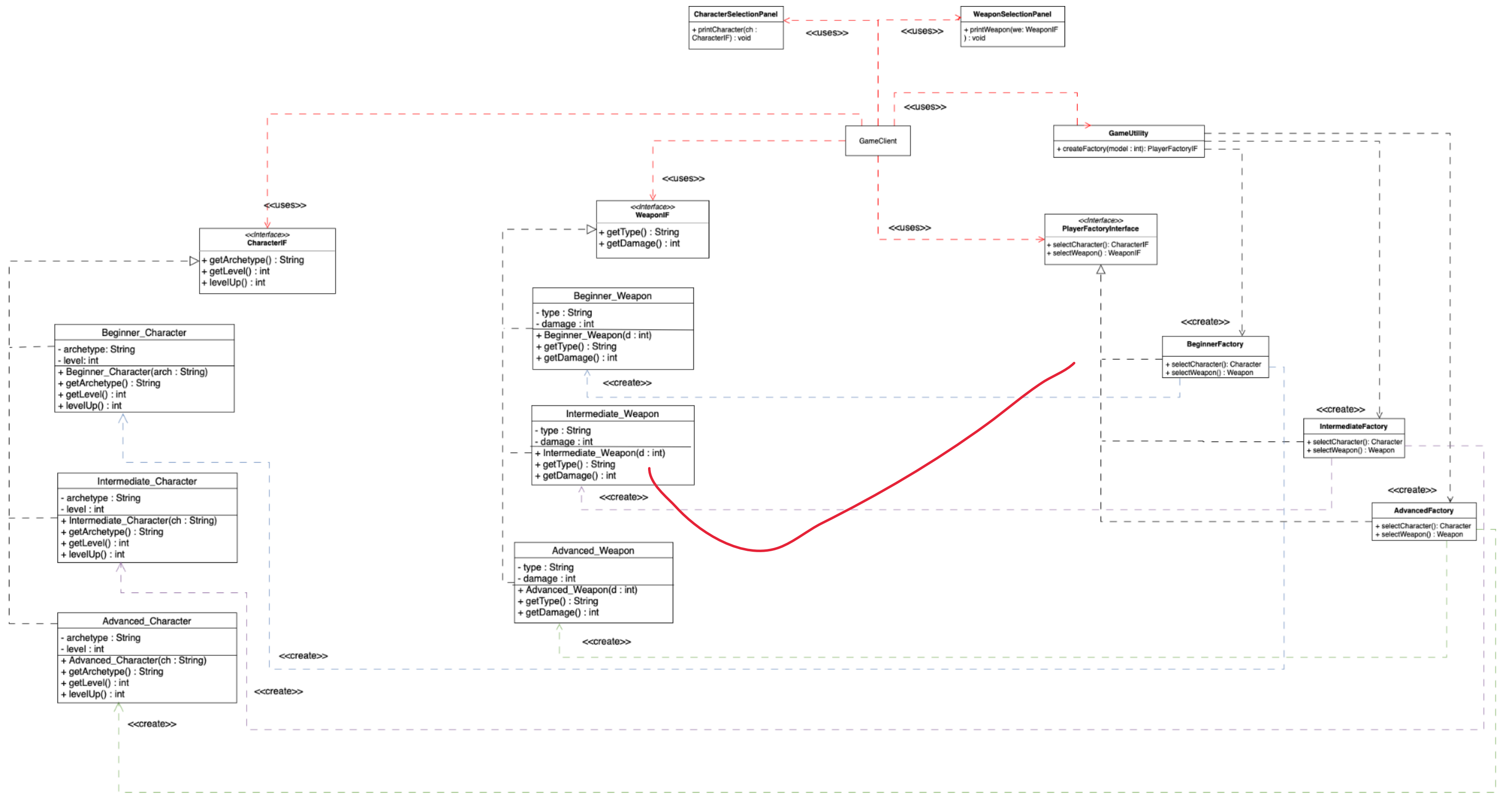
beginner mode
intermediate mode
advanced mode

1. Apply a design pattern to design the system such that the model can be easily extended to cover future changes without affecting the code on the client side. You should use a UML class diagram to document your design.
2. Write Java code to implement your design. You should have a simple test class to show how it works.

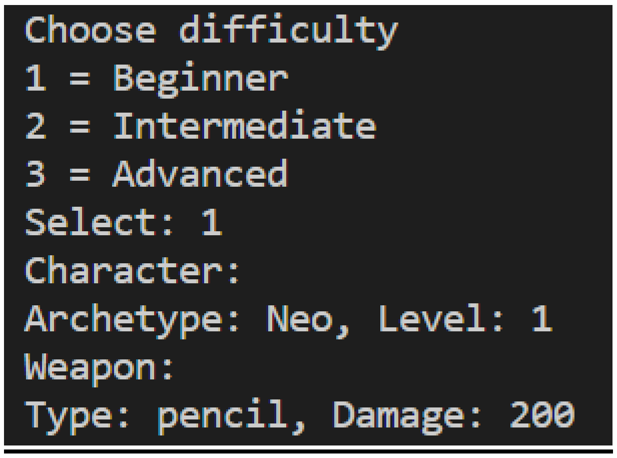
Solution:

- First, remember to zip the src folder of your project and submit the zip file to the ungraded assignment named "Lab3CodeSubmission". One submission from each team.
- Paste a screenshot of a run of your program here.
- Also paste all you source code here.
- Save this report in PDF, then **each student** needs to submit the pdf report to the graded assignment named "Lab3ReportSubmission".

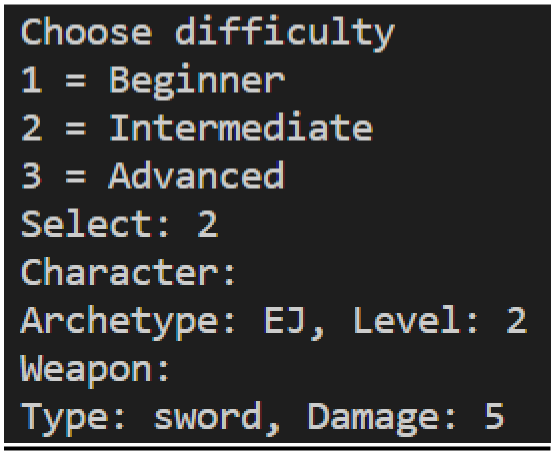
UML Class Diagram



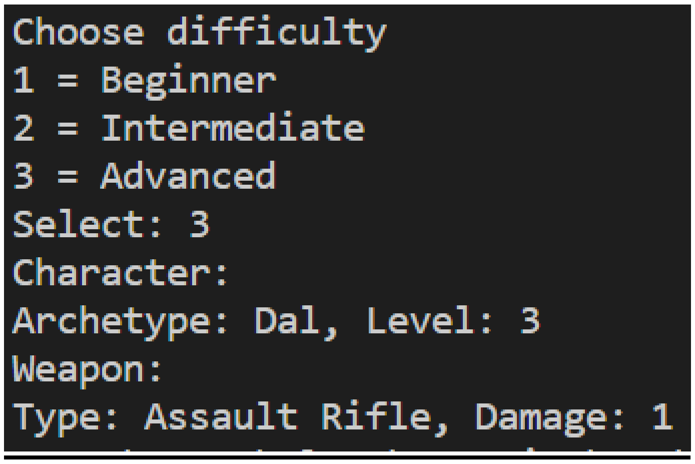
Screenshots



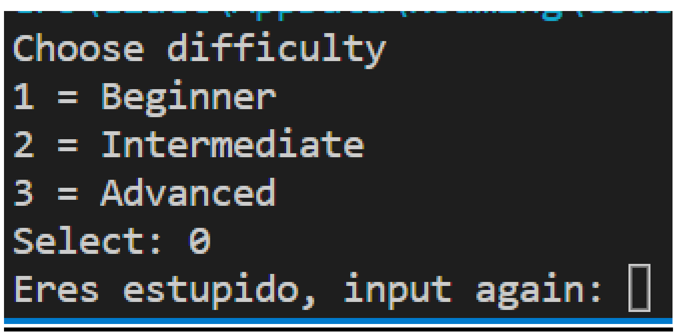
```
Choose difficulty
1 = Beginner
2 = Intermediate
3 = Advanced
Select: 1
Character:
Archetype: Neo, Level: 1
Weapon:
Type: pencil, Damage: 200
```



```
Choose difficulty
1 = Beginner
2 = Intermediate
3 = Advanced
Select: 2
Character:
Archetype: EJ, Level: 2
Weapon:
Type: sword, Damage: 5
```



```
Choose difficulty
1 = Beginner
2 = Intermediate
3 = Advanced
Select: 3
Character:
Archetype: Dal, Level: 3
Weapon:
Type: Assault Rifle, Damage: 1
```



```
Choose difficulty
1 = Beginner
2 = Intermediate
3 = Advanced
Select: 0
Eres estupido, input again:
```

SOURCE CODE

Advanced_Character:

```
package src;

public class Advanced_Character implements CharacterIF {
   private String archetype;  //string fo different archetypes
   private int level; //string for levels
```

```
    public Advanced_Character(String ch){
        this.archetype = ch;
        this.level = 3; //1,2,3 different levels
    }

    public String getArchetype() {
        return archetype;

    }

    public int getLevel() {
        return level;
    }

    public int levelUp(){
        return level++;
    }
}
```

Advanced_Weapon:

```
package src;
public class Advanced_Weapon implements WeaponIF {
    //private weaponType weapon;

    //weapon for advanced level
    String type = "Assault Rifle";
    int damage;
    public Advanced_Weapon(int d) {
        damage = d;
    }
    public String getType() {
        return type;
    }
    public int getDamage() {
        return damage;
    }
}
```

AdvancedFactory:

```
package src;

public class AdvancedFactory implements PlayerFactoryIF {

    public CharacterIF selectCharacter() {
        return new Advanced_Character("Dal");
    }

    public WeaponIF selectWeapon() {
        return new Advanced_Weapon(1);
    }
}
```

Beginner_Character:

```
package src;

public class Beginner_Character implements CharacterIF {
   private String archetype;
   private int level;

   public Beginner_Character(String arch){
       this.archetype = arch;
       this.level = 1;
   }

   public String getArchetype() {
       return archetype;

   }

   public int getLevel() {
       return level;
   }

   public int levelUp(){
       return level++;
   }
}
```

Beginner_Weapon:

```
package src;

public class Beginner_Weapon implements WeaponIF {

    //private weaponType weapon;

    //weapon for beginner level
    String type = "pencil";
    int damage;

    public Beginner_Weapon(int d) {
        damage = d;
    }

    public String getType() {
        return type;
    }
    public int getDamage() {
        return damage;
    }

}
```

BeginnerFactory:

```
package src;

public class BeginnerFactory implements PlayerFactoryIF {


    public CharacterIF selectCharacter(){
        return new Beginner_Character("Neo");
    }
    public WeaponIF selectWeapon(){
        return new Beginner_Weapon(200);
    }
}
```

CharacterIF:

```
package src;

/*
* Character class that includes the
* height and archetype (type of character)
*/
public interface CharacterIF {
    public String getArchetype();
   public int getLevel(); //int to determine levels
   public int levelUp(); //int for level-ups

}
```

CharacterSelectionPanel:

```
package src;

public class CharacterSelectionPanel {
   public void printCharacter(CharacterIF ch) {
       System.out.println("Character:\nArchetype: " + ch.getArchetype() + ", Level: " +
ch.getLevel());
   }
}
```

GameClient:

```
package src;

import java.util.Scanner;

public class GameClient {
   static CharacterSelectionPanel charpanel = new CharacterSelectionPanel();
   static WeaponSelectionPanel weappanel = new WeaponSelectionPanel();
   static GameUtility utility = new GameUtility();
   static PlayerFactoryIF factory;

   public static void main(String[] args) {
       Scanner scanner = new Scanner(System.in);
       System.out.print("Choose  difficulty\n1  =  Beginner\n2  =  Intermediate\n3  =
Advanced\nSelect: ");
       int input = scanner.nextInt();

       //while loop for false inputs
       while(input > 3 || input < 1) {
           System.out.print("Eres estupido, input again: ");
           input = scanner.nextInt();
       }

       factory = utility.createFactory(input);
       charpanel.printCharacter(factory.selectCharacter());
       weappanel.printWeapon(factory.selectWeapon());
       scanner.close();
   }
}
```

GameUtility:

```
package src;

public class GameUtility{
   //case loop for different levels
   public PlayerFactoryIF createFactory(int model) {
       switch(model) {
           case 1:
               return new BeginnerFactory();
           case 2:
               return new IntermediateFactory();
           case 3:
               return new AdvancedFactory();
           default:
               return null;
       }
   }
}
```

Intermediate_Character:

```
package src;

public class Intermediate_Character implements CharacterIF {
   private String archetype;
   private int level;

   public Intermediate_Character(String ch){
       this.archetype = ch;
       this.level = 2;
   }

   public String getArchetype() {
       return archetype;

   }

   public int getLevel() {
       return level;
   }

   public int levelUp(){
       return level++;
   }
}
```

Intermediate_Weapon:

```
package src;

public class Intermediate_Weapon implements WeaponIF{
   //private weaponType weapon;

   String type = "sword";
   int damage;
   public Intermediate_Weapon(int d) {
       damage = d;
   }

   public String getType() {
       return type;
   }
   public int getDamage() {
       return damage;
   }
}
```

IntermediateFactory:

```
package src;

public class IntermediateFactory implements PlayerFactoryIF {

   public CharacterIF selectCharacter() {
       return new Intermediate_Character("EJ");
   }

   public WeaponIF selectWeapon() {
       return new Intermediate_Weapon(5);
   }

}
```

PlayerFactoryIF:

```
package src;

/*
* Interface for the player class
*/
public interface PlayerFactoryIF {
   public abstract CharacterIF selectCharacter();
   public abstract WeaponIF selectWeapon();
}
```

WeaponIF:

```
package src;

public interface WeaponIF {

   public String getType();

   public int getDamage();
}
```

WeaponSelectionPanel:

```java
package src;

public class WeaponSelectionPanel {
   public void printWeapon(WeaponIF we) {
       System.out.println("Weapon:\nType: " + we.getType() + ", Damage: " +
we.getDamage());
   }
}
```