



SE 471 Software Architecture

Lab 3

Student Name		Student CSUSM ID	Contribution percentage
1	Lauren Gonzalez	gonza823	50
2	Sirena Murphree	murph135	50

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 <i>correctness</i>	0-9	10-14	15-19	20
Program: functionality <i>Behavior Testing</i>	0-9	10-14	15-19	20
Program: quality -> <i>Readability</i>	0-2	3-5	6-9	10
Program: quality -> <i>Modularity</i>	0-2	3-5	6-9	10
Program: quality -> <i>Simplicity</i>	0-2	3-5	6-9	10
Total Grade (100)				

SE 471 Software Architecture

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.

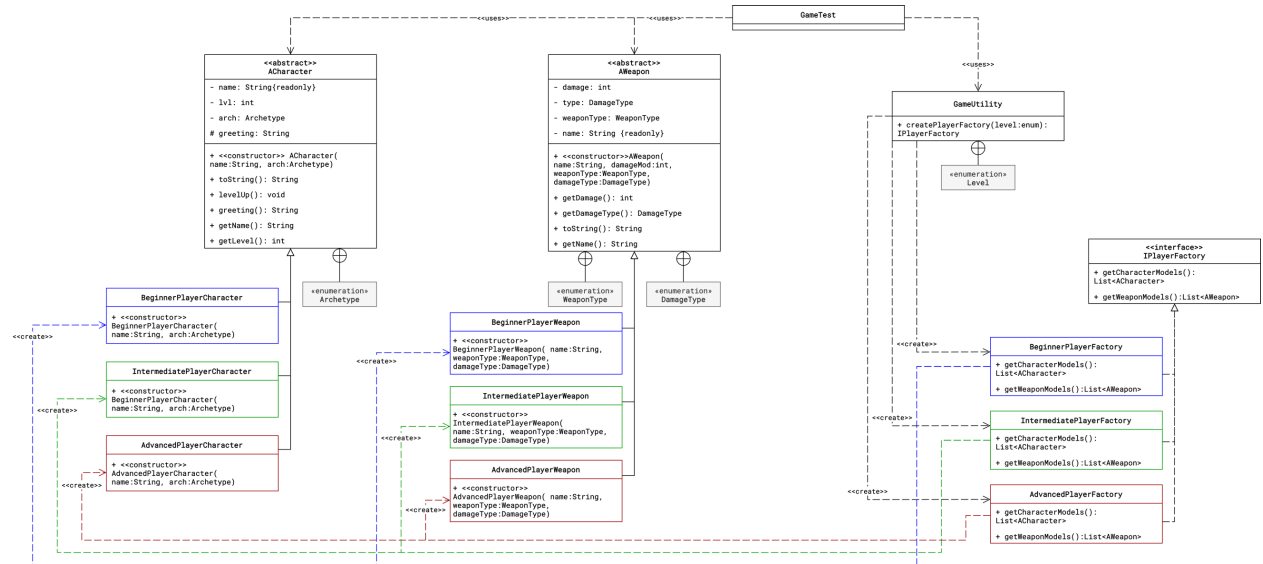


1. Apply a design pattern to design the system such that the model can be easily extended to cover future changes without affecting the code on the client side. You should use a UML class diagram to document your design.
2. Write Java code to implement your design. You should have a simple test class 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 your source code here.
- Save this report in PDF, then **each student** needs to submit the pdf report to the graded assignment named “**Lab3ReportSubmission**”.

SE 471 Software Architecture



Screenshots

```

Select a level: [0] Exit
               [1] Beginner  [3] Advanced
               [2] Intermediate
    
```

Selection: 3

```

Select a Character: [0] Exit
                   [1] Arya Stark - Level 1 Rouge
                   [2] Jon Snow - Level 1 Knight
                   [3] Pyat Pree - Level 1 Wizard
                   [4] Melisandre - Level 1 Clairic
    
```

Selection: 4

```

Select a Weapon: [0] Exit
                [1] Dragonglass [4hp +1 Dagger ]
                [2] Crossbow    [8hp +1 Gun   ]
                [3] Heartsbane  [8hp +1 Sword ]
                [4] Ruby Necklace [8hp +1 Spell Focus ]
                [5] Longbow     [6hp +1 Bow and Arrow ]
    
```

Selection: 4
Evening. I am Melisandre , first of my name, of Game of Thrones.

```

Select a level: [0] Exit
               [1] Beginner  [3] Advanced
               [2] Intermediate
    
```

Selection: 1

```

Select a Character: [0] Exit
                   [1] Bender - Level 1 Rouge
                   [2] Leela - Level 1 Knight
                   [3] Dr. Farnsworth - Level 1 Wizard
                   [4] Dr. Zoidburge - Level 1 Clairic
    
```

Selection: 1

```

Select a Weapon: [0] Exit
                [1] Broken Beer Bottle [4hp -1 Dagger
                [2] Blaster             [8hp -1 Gun
                [3] Norwal Horn          [8hp -1 Sword
                [4] 7 Leaf Clover        [8hp -1 Spell Focus
                [5] Sling Shot           [6hp -1 Bow and Arrow
    
```

Selection: 1
Good News Everyone! I'm Bender from Futurama.

```

Select a level: [0] Exit
               [1] Beginner  [3] Advanced
               [2] Intermediate
    
```

Selection: 2

```

Select a Character: [0] Exit
                   [1] Amos Burton - Level 1 Rouge
                   [2] James Holden - Level 1 Knight
                   [3] Naomi Nagata - Level 1 Wizard
                   [4] Alex Kamal - Level 1 Clairic
    
```

Selection: 3

```

Select a Weapon: [0] Exit
                [1] Wrench [4hp +0 Dagger
                [2] Rail Gun [8hp +0 Gun
                [3] Broken Beam [8hp +0 Sword
                [4] Communicator [8hp +0 Spell Focus
                [5] Standard UN Pistol [6hp +0 Bow and Arrow
    
```

Selection: 4
Hello, this is Naomi Nagata from Expanse.

ACharacter.java

```

public abstract class ACharacter {
    /**
     * Enumerated Types of Character Class Archetypes
     */
    public enum Archetype {
        KNIGHT("Knight"),
        CLAIRIC("Clairic"),
        WIZARD("Wizard"),
        ROUGE("Rouge");

        String charclass;
    }
}
    
```

SE 471 Software Architecture

```
        Archetype(String c){ charclass = c;}
        public String toString() {return charclass;}
    }

    /**
     * The name of the character
     */
    private String name;

    /**
     * the level of the character
     */
    private int lvl;

    /**
     * the Class archetype of the character
     */
    private Archetype arch;

    /**
     * string that holds standard greeting
     */
    protected String greeting = "Greeting Not Defined";

    /**
     * constructor
     * @param name
     * @param charClass
     */
    public ACharacter(String name, Archetype charClass){
        this.name = name;
        this.arch = charClass;
        this.lvl = 1;
    }

    /**
     * returns a string that describes the character
     * @return string
     */
    public String toString() {
        return String.format("%s - Level %d %s", name, lvl, arch.toString());
    }

    /**
     * level up the character
     */
    public void levelUp() {
        lvl++;
    }

    /**
     * returns a string of the character's greeting
     * @return null if undefined
     */
    public String greeting() {
        return greeting;
    }
}
```

SE 471 Software Architecture

```
/**
 * return the name of the character
 * @return string
 */
public String getName() { return name;}

/**
 * return the current level of the character
 * @return int
 */
public int getLevel() {return lvl;}
}
```

AdvancedPlayerCharacter.java

```
public class AdvancedPlayerCharacter extends ACharacter {

    /**
     * constructor for an advanced Game of Thrones themed character
     * @param name
     * @param charClass
     */
    public AdvancedPlayerCharacter(String name, Archetype charClass) {
        super(name, charClass);
        this.greeting = String.format("%s I am %s , first of my name, of Game of
Thrones. \n", "Evening.", this.getName());
    }

}
```

AdvancedPlayerFactory.java

```
import java.util.ArrayList;
import java.util.List;

public class AdvancedPlayerFactory implements IPlayerFactory {

    /**
     * @return List<ACharacter> a list of pre-made Game of Thrones themed
    Characters
     */
    @Override
    public List<ACharacter> getCharacterModels() {
        List<ACharacter> futuramaCharacters = new ArrayList<ACharacter>();
    }
}
```



SE 471 Software Architecture

```
futuramaCharacters.add(new AdvancedPlayerCharacter("Arya Stark",
ACharacter.Archetype.ROUGE));

futuramaCharacters.add(new AdvancedPlayerCharacter("Jon Snow",
ACharacter.Archetype.KNIGHT));

futuramaCharacters.add(new AdvancedPlayerCharacter("Pyat Pree",
ACharacter.Archetype.WIZARD));

futuramaCharacters.add(new AdvancedPlayerCharacter("Melisandre",
ACharacter.Archetype.CLAIRIC));

return futuramaCharacters;
}

/**
 * @return List<AWeapon> a list of pre-made Game of Thrones themed Weapons
 */
@Override
public List<AWeapon> getWeaponsModels() {
    List<AWeapon> weapons = new ArrayList<AWeapon>();

    weapons.add(new AdvancedPlayerWeapon("Dragonglass",
AWeapon.WeaponType.DAGGER, AWeapon.DamageType.PIERCING));

    weapons.add(new AdvancedPlayerWeapon("Crossbow", AWeapon.WeaponType.GUN,
AWeapon.DamageType.NECROTIC));

    weapons.add(new AdvancedPlayerWeapon("Heartsbane",
AWeapon.WeaponType.SWORD, AWeapon.DamageType.SLASHING));

    weapons.add(new AdvancedPlayerWeapon("Ruby Necklace",
AWeapon.WeaponType.SPELLFOCUS, AWeapon.DamageType.RADIANT));

    weapons.add(new AdvancedPlayerWeapon("Longbow", AWeapon.WeaponType.BOW,
AWeapon.DamageType.PIERCING));

    return weapons;
}
}
```

AdvancedPlayerWeapon.java

SE 471 Software Architecture

```
public class AdvancedPlayerWeapon extends AWeapon {  
  
    /**  
     * Constructor for Advanced Weapon  
     * @param name  
     * @param weaponType  
     * @param damageType  
     */  
    public AdvancedPlayerWeapon(String name, WeaponType weaponType, DamageType  
damageType) {  
        super(name, 1, weaponType, damageType);  
    }  
  
}
```

AWeapon.java

```
public abstract class AWeapon {  
  
    /**  
     * Enumeration of damage that a weapon can cause  
     */  
    public enum DamageType {  
        SLASHING,  
        BLUDGENING,  
        PIERCING,  
        NECROTIC,  
        FIRE,  
        RADIANT  
    }  
  
    /**  
     * Weapon types  
     */  
    public enum WeaponType{  
        SWORD("Sword", 8, 5),  
        SPELLFOCUS("Spell Focus", 8, 90),  
        DAGGER("Dagger", 4, 25),  
        BOW("Bow and Arrow", 6, 120),  
        GUN("Gun", 8, 90);  
  
        String weapon;  
        int damage;  
        int range;  
        WeaponType(String w, int d, int r){  
            weapon = w;  
            damage = d;  
            range = r;  
        }  
        public String toString(){ return String.format("%dD %s", damage,  
weapon);}  
    }  
  
    /**  
     * the damage modifier of the weapon  
     */  
    private int damage;
```

SE 471 Software Architecture

```
/**
 * the type of damage the weapon imposes
 */
private DamageType type;

/**
 * the type of weapon as defined by enumeration
 */
private WeaponType weaponType;

/**
 * the name of the weapon
 */
private String name;

/**
 * constructor for abstract weapon
 * @param name
 * @param damageMod
 * @param weaponType
 * @param damageType
 */
public AWeapon(String name, int damageMod, WeaponType weaponType, DamageType
damageType) {
    this.name = name;
    this.damage = damageMod;
    this.weaponType = weaponType;
    this.type = damageType;
}

/**
 * @return int - how much damage this weapon causes
 */
public int getDamage() {
    return weaponType.damage + damage;
}

/**
 * @return DamageType - return the type of damage that the weapon
 */
public DamageType getDamageType() {
    return type;
}

/**
 * @return a string that describes the weapon
 */
public String toString() {
    return String.format("%-20s [%dhp %2d %15s]", name, weaponType.damage,
damage, weaponType.weapon);
}

/**
 * @return the name of the weapon
 */
public String getName() { return name;}
```


SE 471 Software Architecture

```
}
```

BeginnerPlayerCharacter.java

```
public class BeginnerPlayerCharacter extends ACharacter {

    /**
     * constructor for a beginner Futurama themed character
     * @param name
     * @param charClass
     */
    public BeginnerPlayerCharacter(String name, Archetype charClass) {
        super(name, charClass);
        this.greeting = String.format("%s I'm %s from Futurama. \n", "Good News
Everyone!", this.getName());
    }
}
```

BeginnerPlayerFactory.java

```
import java.util.ArrayList;

import java.util.List;

public class BeginnerPlayerFactory implements IPlayerFactory {

    /**
     * @return List<ACharacter> a list of pre-made Futurama themed Characters
     */
    @Override
    public List<ACharacter> getCharacterModels() {

        List<ACharacter> futuramaCharacters = new ArrayList<ACharacter>();

        futuramaCharacters.add(new BeginnerPlayerCharacter("Bender",
ACharacter.Archetype.ROUGE));

        futuramaCharacters.add(new BeginnerPlayerCharacter("Leela",
ACharacter.Archetype.KNIGHT));

        futuramaCharacters.add(new BeginnerPlayerCharacter("Dr. Farnsworth",
ACharacter.Archetype.WIZARD));
    }
}
```



SE 471 Software Architecture

```
futuramaCharacters.add(new BeginnerPlayerCharacter("Dr. Zoidburge",
ACharacter.Archetype.CLAIRIC));

return futuramaCharacters;

}

/**
 * @return List<AWeapon> a list of pre-made Futurama themed Weapons
 */
@Override
public List<AWeapon> getWeaponsModels() {

    List<AWeapon> weapons = new ArrayList<AWeapon>();

    weapons.add(new BeginnerPlayerWeapon("Broken Beer Bottle",
AWeapon.WeaponType.DAGGER, AWeapon.DamageType.BLUDGENING));

    weapons.add(new BeginnerPlayerWeapon("Blaster", AWeapon.WeaponType.GUN,
AWeapon.DamageType.NECROTIC));

    weapons.add(new BeginnerPlayerWeapon("Norwal Horn",
AWeapon.WeaponType.SWORD, AWeapon.DamageType.SLASHING));

    weapons.add(new BeginnerPlayerWeapon("7 Leaf Clover",
AWeapon.WeaponType.SPELLFOCUS, AWeapon.DamageType.RADIANT));

    weapons.add(new BeginnerPlayerWeapon("Sling Shot",
AWeapon.WeaponType.BOW, AWeapon.DamageType.PIERCING));

    return weapons;

}

}
```

BeginnerPlayerWeapon.java

```
public class BeginnerPlayerWeapon extends AWeapon {

    /**
     * constructor for beginner weapon
     * @param name
     * @param weaponType
     */
}
```



SE 471 Software Architecture

```
        * @param damageType
        */
        public BeginnerPlayerWeapon(String name, WeaponType weaponType, DamageType
damageType) {
            super(name, -1, weaponType, damageType);
        }
    }
}
```

Driver.java

```
import java.util.List;
import java.util.Scanner;

public class Driver {

    public static void main(String[] args) {

        IPlayerFactory pf = null;

        /**
         * while selection is null determine player selection
         */

        Scanner scanner = new Scanner(System.in);
        int menuselection;
        do {
            printLevelMenu();
            menuselection = scanner.nextInt();
            switch(menuselection){
                case 1: pf =
GameUtility.createPlayerFactory(GameUtility.Level.BEGINNER);
```



SE 471 Software Architecture

```
                break;

                case 2: pf =
GameUtility.createPlayerFactory(GameUtility.Level.INTERMEDIATE);

                break;

                case 3: pf =
GameUtility.createPlayerFactory(GameUtility.Level.ADVANCED);

                break;

                default:

                break;

        }
}while(pf == null);

/**
 * get player character model from list
 */

List<ACharacter> charOptions = pf.getCharacterModels();
ACharacter myCharacter = null;
do {

    printCharMenu(charOptions);

    menuselection = scanner.nextInt();

    if(menuselection >0 && menuselection <= charOptions.size()) {

        myCharacter = charOptions.get(menuselection - 1);

    }

}while(myCharacter == null);

/**
 * get player weapon model from list
```



SE 471 Software Architecture

```
*/

List<AWeapon> wepOptions = pf.getWeaponsModels();
AWeapon myWeapon = null;
do {
    printWeaponMenu(wepOptions);
    menuselection = scanner.nextInt();
    if(menuselection >0 && menuselection <= wepOptions.size()) {
        myWeapon = wepOptions.get(menuselection - 1);
    }
}while(myWeapon == null);

/**
 * print selected greeting based on player input
 */

System.out.println(myCharacter.greeting());
}

/**
 * print select level menu
 */

public static void printLevelMenu() {

    System.out.println("\n-----");
    System.out.printf("%-19s [%d] %-20s\n", "Select a level:", 0, "Exit");
```



SE 471 Software Architecture

```
        System.out.printf("[%d] %-15s [%d] %-20s\n", 1, "Beginner", 3,
"Advanced");

        System.out.printf("[%d] %-15s \n", 2, "Intermediate");

        System.out.println("-----");

        System.out.print("Selection: ");

    }

    /**
     * @param charOptions prints all options in chosen level characters
     */

    public static void printCharMenu(List<ACharacter> charOptions) {

        System.out.println("\n-----");

        System.out.printf("%-19s [%d] %-20s\n", "Select a Character:", 0, "Exit");

        int i = 0;

        for(ACharacter c : charOptions) {

            i++;

            System.out.printf("[%d]\t%s\n", i, c.toString());

        }

        System.out.println("-----");

        System.out.print("Selection: ");

    }

    /**
     * @param wepOptions prints all options in chosen level weapons
     */
```

SE 471 Software Architecture

```
public static void printWeaponMenu(List<AWeapon> wepOptions) {

    System.out.println("\n-----");

    System.out.printf("%-19s [%d] %-20s\n", "Select a Weapon:", 0, "Exit");

    int i = 0;

    for(AWeapon w : wepOptions) {

        i++;

        System.out.printf("[%d]\t%s\n", i, w.toString());

    }

    System.out.println("-----");

    System.out.print("Selection: ");

}

}
```

GameUtility.java

```
public class GameUtility {

    /**
     * Enumeration of Levels
     */
    public enum Level {
        BEGINNER,
        INTERMEDIATE,
        ADVANCED;
    }

    /**
     * Creates and returns the appropriate PlayerFactory
     * @param l the level the player wants to play at
     * @return IPlayerFactory
     */
    public static IPlayerFactory createPlayerFactory(Level l) {
        switch(l) {
            case BEGINNER : return new BeginnerPlayerFactory();
            case INTERMEDIATE : return new IntermediatePlayerFactory();
            case ADVANCED : return new AdvancedPlayerFactory();
            default: return null;
        }
    }

}
```



SE 471 Software Architecture

IntermediatePlayerCharacter.java

```
public class IntermediatePlayerCharacter extends ACharacter {

    /**
     * constructor for intermediate Expanse themed character
     * @param name
     * @param charClass
     */
    public IntermediatePlayerCharacter(String name, Archetype charClass) {
        super(name, charClass);
        this.greeting = String.format("%s this is %s from Expanse. \n", "Hello,",
this.getName());
    }
}
```

IntermediatePlayerFactory.java

```
import java.util.ArrayList;

import java.util.List;

public class IntermediatePlayerFactory implements IPlayerFactory {

    /**
     * @return List<ACharacter> a list of pre-made Expanse themed Characters
     */
    @Override
    public List<ACharacter> getCharacterModels() {

        List<ACharacter> futuramaCharacters = new ArrayList<ACharacter>();

        futuramaCharacters.add(new IntermediatePlayerCharacter("Amos Burton",
ACharacter.Archetype.ROUGE));

        futuramaCharacters.add(new IntermediatePlayerCharacter("James Holden",
ACharacter.Archetype.KNIGHT));

        futuramaCharacters.add(new IntermediatePlayerCharacter("Naomi Nagata",
ACharacter.Archetype.WIZARD));
    }
}
```


SE 471 Software Architecture

```
futuraCharacters.add(new IntermediatePlayerCharacter("Alex Kamal",
ACharacter.Archetype.CLAIRIC));

return futuraCharacters;

}

/**
 * @return List<AWeapon> a list of pre-made Expanse themed Weapons
 */
@Override
public List<AWeapon> getWeaponsModels() {

    List<AWeapon> weapons = new ArrayList<AWeapon>();

    weapons.add(new IntermediatePlayerWeapon("Wrench",
AWeapon.WeaponType.DAGGER, AWeapon.DamageType.BLUDGENING));

    weapons.add(new IntermediatePlayerWeapon("Rail Gun",
AWeapon.WeaponType.GUN, AWeapon.DamageType.NECROTIC));

    weapons.add(new IntermediatePlayerWeapon("Broken Beam",
AWeapon.WeaponType.SWORD, AWeapon.DamageType.SLASHING));

    weapons.add(new IntermediatePlayerWeapon("Comunicator",
AWeapon.WeaponType.SPELLFOCUS, AWeapon.DamageType.RADIANT));

    weapons.add(new IntermediatePlayerWeapon("Standard UN Pistol",
AWeapon.WeaponType.BOW, AWeapon.DamageType.PIERCING));

    return weapons;

}

}
```

IntermediatePlayerWeapon.java

```
public class IntermediatePlayerWeapon extends AWeapon {

    /**
     * constructor for intermediate weapon
     * @param name
     * @param weaponType
     * @param damageType
     */
    public IntermediatePlayerWeapon(String name, WeaponType weaponType, DamageType
damageType) {
```



SE 471 Software Architecture

```
        super(name, 0, weaponType, damageType);  
    }  
}
```

IPlayerFactory.java

```
import java.util.List;  
  
public interface IPlayerFactory {  
    public List<ACharacter> getCharacterModels();  
    public List<AWeapon> getWeaponsModels();  
}
```