CS 542 Design Pattern and Object-Oriented Analysis

Lab 5

	Student Name	Student CSUSM ID	Contribution percentage
1	Dhruval shah	200361439	33%
2	Niki Patel	200360048	33%
3	Yifan Wu	200379249	33%
4			
5			

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

REVIEWED
By Simon at 3:09 pm, Oct 02, 2019

Total Grade (100)

Problems:

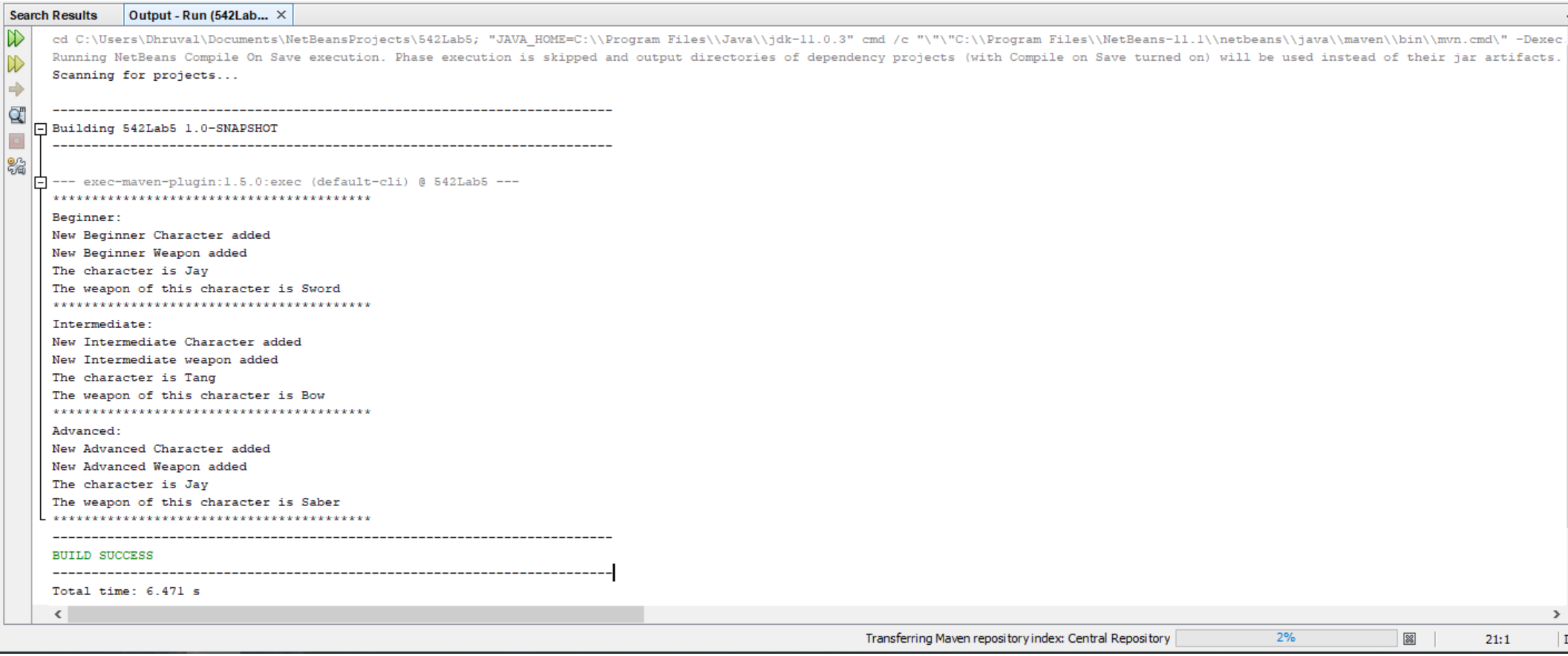
A video game has three modes: beginner, intermediate and advanced. For each mode chosen by a player, the game GUI shows two control objects: a character selection panel and a weapon selection panel. Note that under (a) different modes the system displays different character selection panels and weapon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

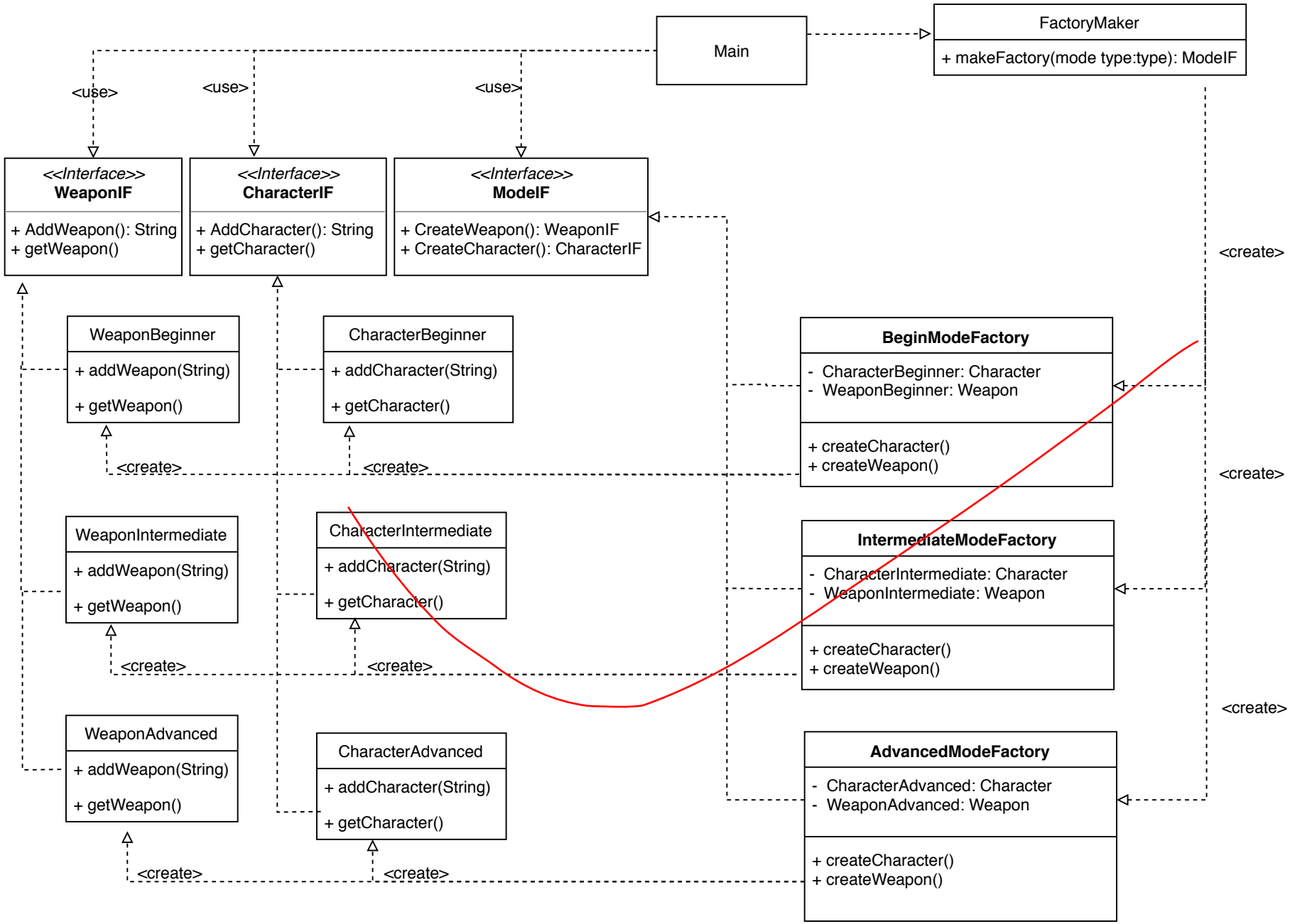
1. Apply a design pattern to design the system such that the model can be easily extended to cover future changes without affecting the code on the client side. You should use a UML class diagram to document your design.

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 "Lab5CodeSubmission". 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 "Lab5ReportSubmission".


Search Results
Output - Run (542Lab...
cd C:\Users\Dhruval\Documents\NetBeansProjects\542Lab5; "JAVA_HOME=C:\\Program Files\\Java\\jdk-11.0.3" cmd /c "\"\"C:\\Program Files\\NetBeans-11.1\\netbeans\\java\\maven\\bin\\mvn.cmd\" -Dexec
Running NetBeans Compile On Save execution. Phase execution is skipped and output directories of dependency projects (with Compile on Save turned on) will be used instead of their jar artifacts.
Scanning for projects...
Building 542Lab5 1.0-SNAPSHOT
--- exec-maven-plugin:1.5.0:exec (default-cli) @ 542Lab5 ---
Beginner:
New Beginner Character added
New Beginner Weapon added
The character is Jay
The weapon of this character is Sword
Intermediate:
New Intermediate Character added
New Intermediate weapon added
The character is Tang
The weapon of this character is Bow
Advanced:
New Advanced Character added
New Advanced Weapon added
The character is Jay
The weapon of this character is Saber
BUILD SUCCESS
Total time: 6.471 s
Transferring Maven repository index: Central Repository
2%
21:1


Main
FactoryMaker
+ makeFactory(mode type:type): ModeIF
<use>
<use>
<use>
<<Interface>>
WeaponIF
+ AddWeapon(): String
+ getWeapon()
<<Interface>>
CharacterIF
+ AddCharacter(): String
+ getCharacter()
<<Interface>>
ModeIF
+ CreateWeapon(): WeaponIF
+ CreateCharacter(): CharacterIF
<create>
WeaponBeginner
+ addWeapon(String)
+ getWeapon()
CharacterBeginner
+ addCharacter(String)
+ getCharacter()
BeginModeFactory
- CharacterBeginner: Character
- WeaponBeginner: Weapon
+ createCharacter()
+ createWeapon()
<create>
<create>
<create>
WeaponIntermediate
+ addWeapon(String)
+ getWeapon()
CharacterIntermediate
+ addCharacter(String)
+ getCharacter()
IntermediateModeFactory
- CharacterIntermediate: Character
- WeaponIntermediate: Weapon
+ createCharacter()
+ createWeapon()
<create>
<create>
<create>
WeaponAdvanced
+ addWeapon(String)
+ getWeapon()
CharacterAdvanced
+ addCharacter(String)
+ getCharacter()
AdvancedModeFactory
- CharacterAdvanced: Character
- WeaponAdvanced: Weapon
+ createCharacter()
+ createWeapon()
<create>
<create>

AdvancedModeFactory.java

```
package pkg542Lab5;

/**
 *
 * @author Dhruval
 */
public class AdvancedModeFactory implements ModeIF {

    @Override
    public CharacterIF createCharacter() {
        String add = "New Advanced Character added";
        System.out.println(add);
        return new CharacterAdvanced();

    }
    @Override
    public WeaponIF createWeapon() {
        String add = "New Advanced Weapon added";
        System.out.println(add);
        return new WeaponAdvanced();
    }
}
```

CharacterAdvanced.java

```
package pkg542Lab5;

/**
 *
 * @author Dhruval
 */
public class CharacterAdvanced implements CharacterIF {

    String name;

    @Override
    public void addCharacter(String c) {
        this.name = c;
    }

    @Override
    public String getCharacter() {
        return "The character is " + this.name;
    }

}
```

BeginModeFactory.java

```
package pkg542Lab5;

/**
 *
 * @author Dhruval
 */
public class BeginModeFactory implements ModeIF {

    @Override
    public CharacterIF createCharacter() {
        String add = "New Beginner Character added";
        System.out.println(add);
        return new CharacterBeginner();
    }

    @Override
    public WeaponIF createWeapon() {
        String add = "New Beginner Weapon added";
        System.out.println(add);
        return new WeaponBeginner();
    }
}
```

CharacterBeginner.java

```
package pkg542Lab5;

/**
 *
 * @author Dhruval
 */
public class CharacterBeginner implements CharacterIF {

    String name;

    @Override
    public void addCharacter(String c) {
        this.name = c;
    }

    @Override
    public String getCharacter() {
        return "The character is " + this.name;
    }
}
```

CharacterIF.java

```
package pkg542Lab5;

/**
 *
 * @author Dhruval
 */
public interface CharacterIF {

    public void addCharacter(String c);

    public String getCharacter();

}
```

CharacterIntermediate.java

```
package pkg542Lab5;

/**
 *
 * @author Dhruval
 */
public class CharacterIntermediate
                    implements CharacterIF {String name;

    @Override
    public void addCharacter(String c) {
        this.name = c;
    }

    @Override
    public String getCharacter() {
        return "The character is " + this.name;
    }

}
```

FactoryMaker.java

```
package pkg542Lab5;

/**
 *
 * @author Dhruval
 */
public class FactoryMaker {

    public enum ModeType{
        BEGINNER, INTERMEDIATE, ADVANCED
    }

    /**
     * makeFactory method to create a factory
     * @param type -- the mode
     * @return a modeFactory
     */
    public static ModelF makeFactory(ModeType type){
        switch(type){
            case BEGINNER:
                return new BeginModeFactory();
            case INTERMEDIATE:
                return new IntermediateModeFactory();
            case ADVANCED:
                return new AdvancedModeFactory();
            default:
                throw new IllegalArgumentException("ModeType is not supported");
        }
    }



}
```

ModelF.java

```
package pkg542Lab5;

/**
 *
 * @author Dhruval
 */

public abstract interface ModelF {
    public abstract CharacterIF createCharacter();
    public abstract WeaponIF createWeapon();
}
```

```
package pkg542Lab5;

/**
 *
 * @author Dhruval
 */

/**
 * Main class to test this factory design pattern
 * This project is a simple demo that using abstract factory design pattern
 * As you see in the code, we just simply using String to represent character
 * and weapon. In further development, we may use character objects to give more
 * attributes of a character like name, hair, eye color,HP,MP,etc; using weapon
 * object to give more attributes like Attack power, speed and so on.
 */
public class Main {

    private CharacterIF character;
    private WeaponIF weapon;

    /**
     * createMode to create a mode with character name and weapon
     * @param mode -- the mode to build
     * @param name -- the character name
     * @param weapon -- the weapon
     */
    public void createMode(final ModeIF mode, final String name, final String weapon) {
        setCharacter(mode.createCharacter(), name);
        setWeapon(mode.createWeapon(), weapon);
    }

    public CharacterIF getCharacter() {
        return character;
    }

    public WeaponIF getWeapon() {
        return weapon;
    }

    public void setCharacter(final CharacterIF character, final String name) {
        this.character = character;
        character.addCharacter(name);
    }

    public void setWeapon(final WeaponIF weapon, final String weaponName) {
        this.weapon = weapon;
        weapon.addWeapon(weaponName);
    }

    public static void main(String[] args) {
        Main test = new Main();
        System.out.println("****************************************");
        System.out.println("Beginner: ");
        test.createMode(FactoryMaker.makeFactory(FactoryMaker.ModeType.BEGINNER),
                "Jay", "Sword");
        System.out.println(test.getCharacter().getCharacter());
        System.out.println(test.getWeapon().getWeapon());
        System.out.println("****************************************");
        System.out.println("Intermediate: ");
        test.createMode(FactoryMaker.makeFactory(FactoryMaker.ModeType.INTERMEDIATE),
                "Tang", "Bow");
        System.out.println(test.getCharacter().getCharacter());
        System.out.println(test.getWeapon().getWeapon());
        System.out.println("****************************************");
        System.out.println("Advanced: ");
        test.createMode(FactoryMaker.makeFactory(FactoryMaker.ModeType.ADVANCED),
                "Jay", "Saber");
        System.out.println(test.getCharacter().getCharacter());
        System.out.println(test.getWeapon().getWeapon());
        System.out.println("****************************************");
    }
}
```

IntermediateModeFactory.java

```
package pkg542Lab5;

/**
 *
 * @author Dhruval
 */
public class IntermediateModeFactory implements ModeIF {

    @Override
    public CharacterIF createCharacter() {
        String add = "New Intermediate Character added";
        System.out.println(add);
        return new CharacterIntermediate();
    }

    @Override
    public WeaponIF createWeapon() {
        String add = "New Intermediate weapon added";
        System.out.println(add);
        return new WeaponIntermediate();
    }

}
```

WeaponIF.java

```
package pkg542Lab5;

/**
 *
 * @author Dhruval
 */
public interface WeaponIF {

    public void addWeapon(String name);

    public String getWeapon();

}
```

WeaponBeginner.java

```
package pkg542Lab5;

/**
 *
 * @author Dhruval
 */
public class WeaponBeginner implements WeaponIF{

    private String name;

    //add a weapon
    @Override
    public void addWeapon(String n){
        this.name = n;
    }

    //get this weapon
    @Override
    public String getWeapon(){
        return "The weapon of this character is " + this.name;
    }

}
```

WeaponIntermediate.java

```
package pkg542Lab5;

/**
 *
 * @author Dhruval
 */
public class WeaponIntermediate implements WeaponIF {

    private String name;

    @Override
    public void addWeapon(String n){
        this.name = n;
    }

    @Override
    public String getWeapon(){
        return "The weapon of this character is " + this.name;
    }
}
```

WeaponAdvanced.java

```
package pkg542Lab5;

/**
 *
 * @author Dhruval
 */
public class WeaponAdvanced implements WeaponIF{

    private String name;

    @Override
    public void addWeapon(String n){
        this.name = n;
    }

    @Override
    public String getWeapon(){
        return "The weapon of this character is " + this.name;
    }

}
```