

Abstract Factory Homework

What is Abstract Factory?

Abstract Factory is a creational pattern that creates a group of objects that relate to themselves using only a type of object.

How pattern work

1 create an interface of the object that we are going to build and make abstract method to make those object

2 complete the real object class that use in the application

3. When the user need those objects, then we use the abstract method to build the object

What problem that abstract factory solved

1 it solves the complexity of the group of the object

2 it reduces the complexity of the system

3. It also prevents the god class (one class do everything)

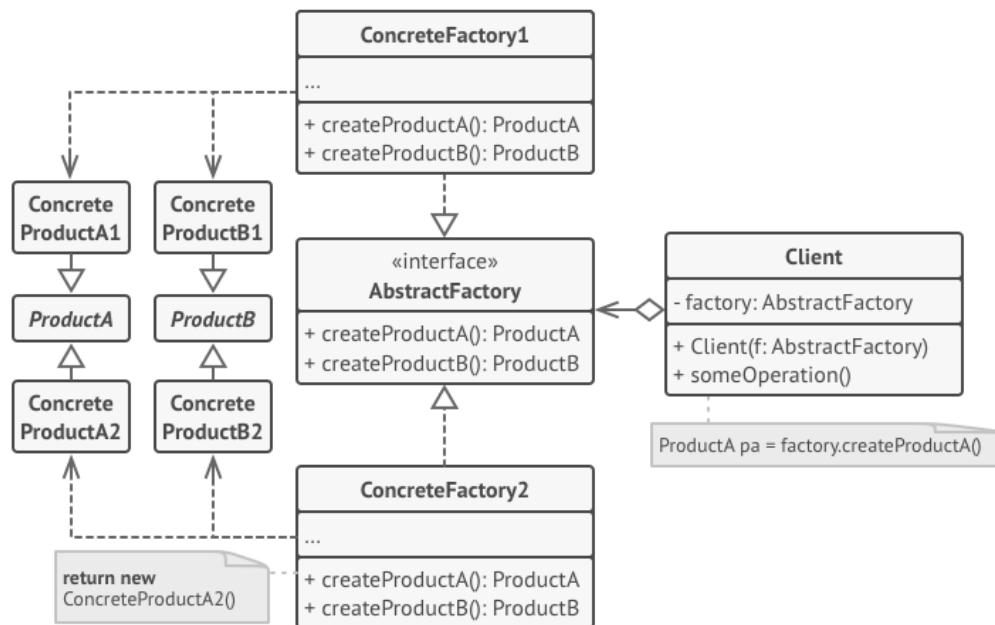
Advantage

- The object will match the type of the factory
- Clean code

Disadvantage

- Even if it solves the complexity of the system but it very hard to read and take time to understand the code

This is an Abstract factory class diagram



This is my example Example background

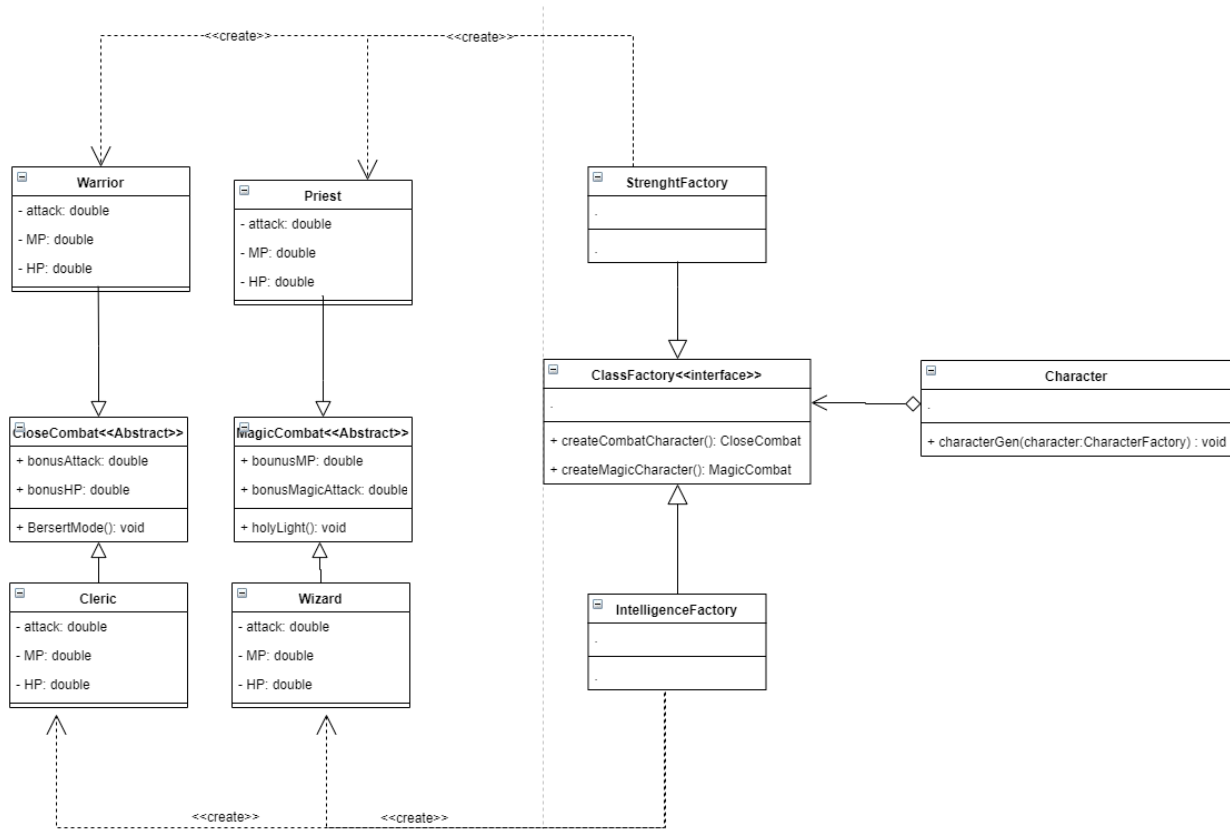
My RPG world has 2 types of power and 2 types of combat. The Character in my world will have 4 classes by following.

class	Strenght	Intelligence
Close	Warrior	priest
Magic	Cleric	Wizard

And to put this to the abstract factory by building the big factory class call.

CharacterFactory relates to two significant types: StrenghtFactory and MagicFactory Then, Create AbstractMethod name closeCombat and MagicCombat to create those 4-following classes.

Here my example class diagram.



And this is a core code in java.

Character.java This class generates the Character by type of factory.

```
1 public class Character {
2     public static void main(String[] args) {
3         System.out.println("create strength character\n");
4         characterGen(new StrengthFactory());
5         System.out.println("\n===== \n");
6         System.out.println("create magic character\n");
7         characterGen(new MagicFactory());
8     }
9
10    public static void characterGen(CharacterFactory character){
11        CloseCombat sahachan = character.createCombatCharacter();
12        MagicCombat sahachan2 = character.createMagicCharacter();
13
14        sahachan.BersertMode();
15        sahachan2.holyLight();
16    }
17 }
```

On the picture above, I pass StrengthFactory to create a CharacterStrType

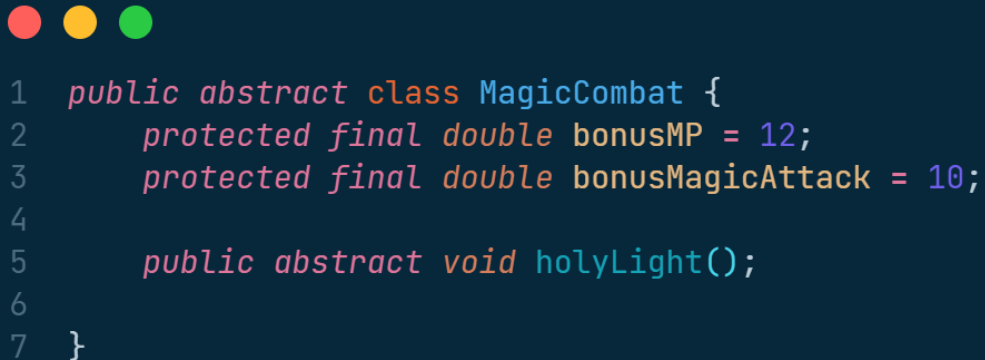
And Another is MagicFactory for CharaterMagicType

In the StrengthFactory and MagicFacotry in java (actually, it the same)

```
1 public class MagicFactory implements CharacterFactory {
2
3     @Override
4     public CloseCombat createCombatCharacter() {
5         System.out.println("Cleric is created");
6         return new Cleric();
7     }
8
9     @Override
10    public MagicCombat createMagicCharacter() {
11        System.out.println("Wizard is summoned");
12        return new Wizard();
13    }
14 }
```

As you can see, it going to create the closeCombat and Magictype

And here this is the CloseCombat, and MagicCombat looks like in java.



```
1 public abstract class MagicCombat {  
2     protected final double bonusMP = 12;  
3     protected final double bonusMagicAttack = 10;  
4  
5     public abstract void holyLight();  
6  
7 }
```

In my example, when I pass the MagicType(MagicFactory) to the generate Character and then choose to create the Magic combat, the java program will start a wizard.

In short, I just create a wizard by just passing MagicFactory and choose to createMagicCombat without instance, the real wizard class.

ref:

[Abstract Factory - Saladpuk.com](http://Saladpuk.com)

[Abstract Factory Pattern - GeeksforGeeks](http://GeeksforGeeks)

[Design Pattern - Abstract Factory Pattern - Tutorialspoint](http://TutorialsPoint)

My code

[oat431/DesignPatternExample: This repo aim to collect the all of 23 design pattern Example and summarize explanation \(github.com\)](https://github.com/oat431/DesignPatternExample)

Sahachan Tippimwong 622115039