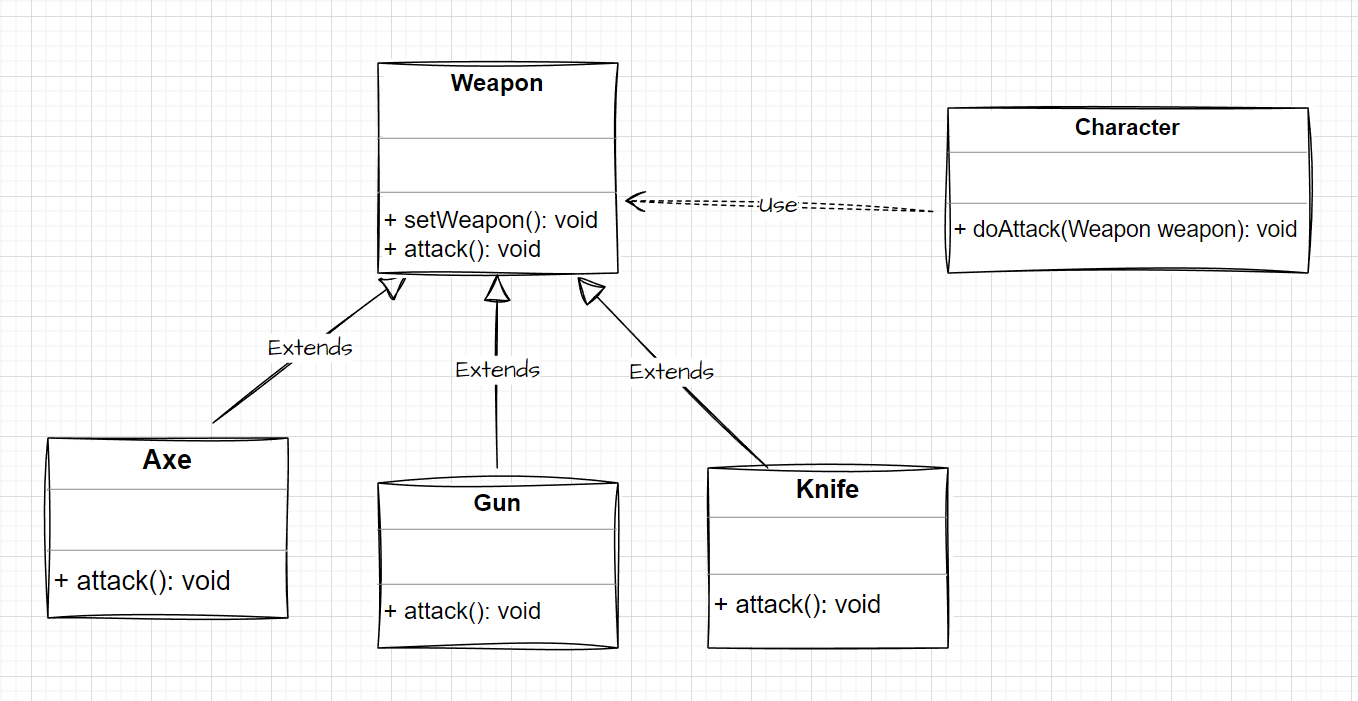
# Design Pattern

## 一.Chapter 1

### 1.OOP\_REVIEW

题目描述：有一动作冒险游戏， 游戏角色（Character类）可以使用多种武器（Weapon抽象类）战斗（fight方法），武器包括刀Knife、弓箭Bow、斧头Axe等等，战斗时首先拿起武器，然后使用武器攻击（每种武器攻击方法不同），每个游戏角色一次只能使用一个武器，但是可以在游戏的过程中切换武器.

一. **由题意可画出如下UML类图**：有一个抽象类Weapon,由三个子类Axe,Gun,Knife**继承**Weapon并实现了抽象类的 attack()方法，然后游戏角色类Character有一个doAttack()方法，消息传递是通过它的参数类型是Weapon，传递一个Weapon类的子类,作为角色的武器。所以 Character和 Weapon的关系为**依赖**关系。



二.**代码实现如下**：

1. 抽象类**Weapon**的代码

* public abstract class Weapon {
    
   public void setWeapon() {
    
   System.out.print("拿起武器: ");
    
   attack();
    
   }
    
   public abstract void attack();
    
  }

1. **Weapon**的子类的代码
   * 刀Knife
   * public class Knife extends Weapon{  
      @Override  
      public void attack() {  
      System.out.println("用刀砍");  
      }  
     }
   * 枪Gun
   * public class Gun extends Weapon{  
      @Override  
      public void attack() {  
      System.out.println("用枪射击");  
      }  
     }
   * 斧头Axe
   * public class Axe extends Weapon{  
      @Override  
      public void attack() {  
      System.out.println("用斧头劈");  
      }  
     }
2. 游戏角色类**Character**

* public class Character {  
   public Character() {  
   System.out.println("角色创建");  
   }  
   public void doAttack(Weapon weapon) {  
   weapon.setWeapon();  
   }  
  }

1. 测试结果Main类

* public class Main {  
   public static void main(String[] args) {  
   Character role1 = new Character();  
     
   //动态绑定，编译看左边，运行看右边  
   Weapon axe = new Axe();  
   role1.doAttack(axe);  
    
   Weapon gun = new Gun();  
   role1.doAttack(gun);  
    
   Weapon knife = new Knife();  
   role1.doAttack(knife);  
    
   }  
  }
* 运行结果为：
* 