**哈尔滨工业大学（威海）**

**Java语言实验报告**

|  |  |  |  |
| --- | --- | --- | --- |
| 课程名称 | Java程序设计 | 课程编号 | SE33601 |
| 实验名称 | 实验3：类的关联及其应用 | | |
| 实验类型 | 验证与编程 | 实验学时 | 2学时 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 班号 | 21911103 | 学号 | 2191110314 | 姓名 | 刘浩然 |
| 开始 | 2020.10.23 | 截止 | 2019.10.27 | 成绩 |  |
| 批语 |  | | | | |

# 1 实验目的、内容和要求

本次实验着重训练学生面向对象的灵活应用，体现在两个方面：类的水平关联关系和垂直关联关系。同时，继续训练利用UML设计后再进行编码的软件开发过程。因此本次实验的主要目的：

1. 训练UML中类的关系的表达
2. 掌握类中关联关系

# 2 实验内容

在下面图1中，你将看到一堆杂乱的类与接口，这是取自一个交互式的冒险游戏。你将看到代表游戏角色的类，以及武器行为的类。每个角色一次只能使用一个武器，但是可以在游戏的过程中换武器。

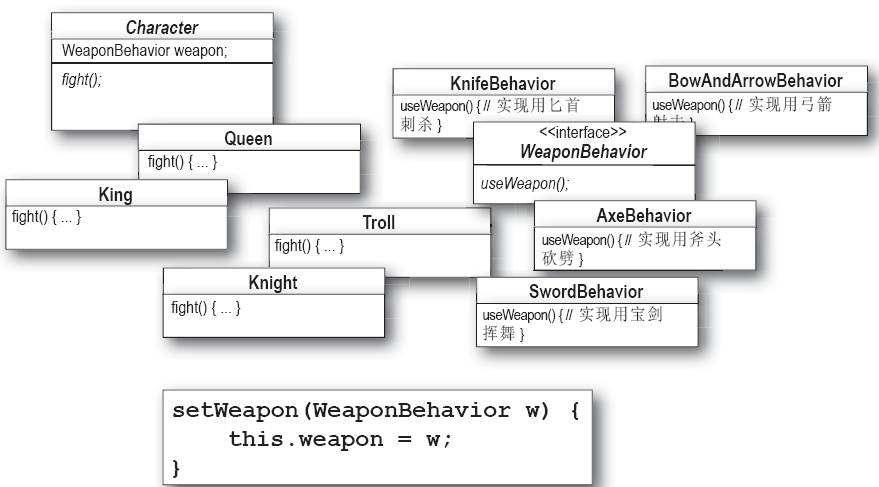


图1 某冒险游戏部分类

* 实验内容：参考讲义相关案例，对上述的杂乱的类与接口进行整理，形成完整的设计。要求利用StarUML工具绘制出整理完毕的类图。类图绘制过程中，需要把上图中的setWeapon()方法放到上图中某类中，从而实现武器运行时更换功能，不用画客户端。绘制时特别注意abstract class和interface的绘制。类图绘制如下：

|  |
| --- |
|  |

实验内容2根据绘制的类图，实现各个类，请编写代码。调通后，粘贴在下方相应位置

Character类的代码：

|  |
| --- |
| public abstract class Character {  WeaponBehavior weapon;  abstract void fight();  abstract void setWeapon(WeaponBehavior w);  } |

King类的代码

|  |
| --- |
| public class King extends Character {  public King() {  weapon = new SwordBehavior();  }  public void setWeapon(WeaponBehavior w) {  this.weapon = w;  }  public void fight() {  System.out.println("King fighting");  }  } |

Queen类的代码

|  |
| --- |
| public class Queen extends Character {  public Queen() {  weapon = new BowAndArrowBehavior();  }  public void setWeapon(WeaponBehavior w) {  this.weapon = w;  }  public void fight() {  System.out.println("Queen fighting");  }  } |

Knight类的代码

|  |
| --- |
| public class Knight extends Character {  public Knight() {  weapon = new AxeBehavior();  }  public void setWeapon(WeaponBehavior w) {  this.weapon = w;  }  public void fight() {  System.out.println("Knight fighting");  }  } |

Troll类的代码

|  |
| --- |
| public class Troll extends Character {  public Troll() {  weapon = new KnifeBehavior();  }  public void setWeapon(WeaponBehavior w) {  this.weapon = w;  }  public void fight() {  System.out.println("Troll fighting");  }  } |

WeaponBehavior接口的代码

|  |
| --- |
| public interface WeaponBehavior {  public void useWeapon();  } |

KnifeBehavior类的代码

|  |
| --- |
| public class KnifeBehavior implements WeaponBehavior {  public void useWeapon() {  System.out.println("using knife");  }  } |

BowAndArrowBehavior类的代码

|  |
| --- |
| public class BowAndArrowBehavior implements WeaponBehavior {  public void useWeapon() {  System.out.println("using bow and arrow");  }  } |

AxeBehavior类的代码

|  |
| --- |
| public class AxeBehavior implements WeaponBehavior {  public void useWeapon() {  System.out.println("using axe");  }  } |

SwordBehavior类的代码

|  |
| --- |
| public class SwordBehavior implements WeaponBehavior {  public void useWeapon() {  System.out.println("using sword");  }  } |

测试类的代码：

|  |
| --- |
| public class Test {  public static void main(String args[]) {  Character king = new King();  Character queen = new Queen();  Character knight = new Knight();  Character troll = new Troll();  WeaponBehavior axe = new AxeBehavior();  WeaponBehavior sword = new SwordBehavior();  WeaponBehavior knife = new KnifeBehavior();  WeaponBehavior bowAndArrow = new BowAndArrowBehavior();  king.weapon.useWeapon();  king.setWeapon(axe);  king.weapon.useWeapon();  king.fight();  queen.weapon.useWeapon();  queen.setWeapon(sword);  queen.weapon.useWeapon();  queen.fight();  knight.weapon.useWeapon();  knight.setWeapon(knife);  knight.weapon.useWeapon();  knight.fight();  troll.weapon.useWeapon();  troll.setWeapon(bowAndArrow);  troll.weapon.useWeapon();  troll.fight();  }  } |

运行结果截图：

|  |
| --- |
|  |