Lab7: Interface & Abstract

> 在本实验中，将创建一个简单的游戏，其中包含不同类型的角色，例如骑士、弓箭手和法师。 每种角色类型都有一些共同的行为，例如移动和攻击，但它们也有一些独特的行为来区分它们。
>
> 要创建这些角色，将使用抽象和接口的概念以更灵活和可扩展的方式实现相同的目标。

注意：本lab中提供的代码结构仅供大家参考，大家根据需求合理进行修改或者自己重新进行设计。示例里类的各种属性和方法仅作为参考进行使用，可能存在不合理的地方，大家在实现过程中要仔细辨别。本次lab不要求代码鲁棒性，主要查看大家代码设计风格，为下一次代码重构奠定基础！！！

1. 创建一个抽象类：

   创建一个名为“Character”的抽象类，包含所有角色的通用行为，如移动和攻击，并定义一些子类必须实现的抽象方法。 下面是“Character”类的示例：

```java
public abstract class Character {
    protected int health;
    protected int strength;
    // Abstract methods that subclasses will implement
    public abstract void takeDamage(int damage);
    public abstract void specialAttack(Position pos);
    public abstract void move(Command order)

    public void attack(Position pos) {
        // code for attacking the other character
    }
    // Getter and setter methods
    public int getHealth() {
        return health;
    }

    public void setHealth(int health) {
        this.health = health;
    }

    public int getStrength() {
        return armor;
    }
```

```
24
25      public void setStrength(int strength) {
26          this.strength = strength;
27      }
28  }
```

请注意，我们已经创建了一个名为“specialAttack()”的抽象方法。 这个方法是抽象的，因为每个角色类型都会有不同的特殊攻击，我们希望子类实现自己的特殊攻击逻辑。

2. 创建子类：

   现在，创建扩展“Character”类并实现它们自己独特行为的子类，例如“Knight”、“Archer”和“Wizard”。 这是“Knight”类的示例：

```
1  public class Knight extends Character {
2      private int armor;
3      public Knight(int health, int strength, int armor) {
4          this.health = health;
5          this.strength = strength;
6          this.armor = armor;
7      }
8      public void takeDamage(int damage) {
9          // code for the knight's damage
10     }
11     public void specialAttack(Character other) {
12         // code for the knight's special attack
13     }
14 }
```

   同样，创建其他子类“Archer”和“Wizard”，它们扩展“Character”类并实现它们自己的独特行为。

3. 创建接口：

   创建一个名为“RangedAttack”的接口，它定义了某些角色具有的行为，例如远距离攻击的能力。 以下是“RangedAttack”的示例：

```
1  public interface RangedAttack {
2      void rangedAttack(Position pos);
3  }
```

   任何实现“RangedAttack”接口的类都必须提供“rangedAttack()”方法的实现。

4. 实现接口：

现在，创建一个名为“Archer”的类，它实现“RangedAttack”接口并添加自己独特的行为。 以下是“Archer”类的示例：

```
public class Archer extends Character implements RangedAttack {
    private int range;
    public Archer(int health, int strength, int range) {
        this.health = health;
        this.strength = strength;
        this.range = range;
    }

    @override
    public void attack(Position pos) {
        // code for attacking the other character
    }

    public void takeDamage(int damage) {
        // code for the knight's damage
    }
    public void specialAttack() {
        // archers don't have a special attack
    }
    public void rangedAttack(Position pos) {
        // code for the archer's ranged attack
    }
}
```

同样，创建其他实现“RangedAttack”接口的类并添加它们自己的独特行为。

5. 测试你的游戏：

   最后，通过创建不同角色类型的实例并调用它们的方法来测试您的游戏。 确保每个字符类型的行为都符合预期，并且正确实现了其独特的行为。

要求：

1. 本次实验主要考察大家的对interface和abstract关键字的理解，因此不着重于代码逻辑和正确性，只需要能够通过文档中提供的sample即可。
2. ~~本实验需要提交一份代码设计说明文档，标明代码的设计逻辑和层次结构。例如，将xxx功能设计为interface，原因是xxx。将xxx方法/属性定义在abstract类中，原因是xxx。要求说明文档能够大致描述代码的设计结构。~~（考虑了下，为了简化lab工作量，就不写文档了）
3. 角色仅包括“Knight”、“Archer”和“Wizard”，其中：
   a. 骑士

i. 武器：盔甲，在承受任何伤害时可以减免盔甲数值的伤害，仅可对周围一格的一个目标造成伤害
ii. 移动：一次移动一格
iii. 特殊攻击：损失20血，对周围一格内的一个目标造成50伤害
iv. 治愈：可以恢复自己40血
v. 第一次死亡后可以半血原地复活
vi. 属性
    - health: 100
    - strength: 30
    - armor: 5

b. 法师
i. 武器：法杖，可进行远程普通攻击，范围为2
ii. 范围攻击：可以对范围3内目标周围1格的所有角色造成30点伤害
iii. 特殊攻击：对范围为2内的一个目标造成50伤害，恢复自己20血
iv. 移动：一次移动一格
v. 治愈：可以恢复自己30血
vi. 属性
    - health: 40
    - strength: 10

c. 弓箭手
i. 武器：弓箭，可进行远程普通攻击，范围为3
ii. 受到1.5倍伤害
iii. 范围攻击：可以对范围为2内目标周围1格的所有角色造成20点伤害
iv. 特殊攻击：对范围6内的一个目标造成20伤害
v. 移动：可以最多一次移动两格
vi. 属性：
    - health: 60
    - strength: 20

4. 普通攻击造成strength值的伤害
5. 地图为二维地图方格阵
6. 范围定义为目标位置与角色当前位置的横纵坐标差之和

普通攻击：1

特殊攻击：2

范围攻击：3

治愈：4

复活：5

Sample：

```
5 5（5x5的地图）
Knight 0 0（坐标）
Witch 4 4
Archer 4 0
Knight l（移动到0 1）
Archer k（移动到2 0）
Witch k（移动到3 4）
Archer 1 0 1（攻击0 1位置的骑士，骑士掉20-5=15血，剩余85血）
Witch 2 2 3（特殊攻击2 3位置，未命中任何目标）
Knight j（移动到1 1）
Archer 2 3 4（特殊攻击3 4位置的witch，掉20血，剩余20血）
Witch h（移动到3 3）
Knight j（移动到2 1）
Witch 3 2 1（对位置2 1坐标为中心一格内的所有角色造成30伤害，K剩60血，A剩15血）
Archer 1 2 1（普通攻击2 1坐标的角色，K剩45血）
Witch h （移动到3 2）
Witch 2 2 1（K死亡，W生命值40）
Knight 5 （K原地复活，剩余生命值50）
Knight 4（K剩余90血）
Archer 3 2 2（对2 2周围1格内所有角色造成20伤害，K剩余75，W剩余20）
Witch 4 （W剩余40，血量不能溢出）
Knight 2 2 0（K对2 0位置造成50伤害，A死亡，K剩余55血）
End（End是截止指令）
Knight 2 1 55（输出各自的位置与剩余血量）
Witch 3 2 40
Archer 2 0 0
```