Homework：

1、 **请举例说明克隆模式的其他应用。**

答：在游戏开发中，使用克隆模式可以非常有效地管理和复制游戏中的角色。假设需要创建大量的小兵，每个兵种都有自己独特的技能和属性。通过使用克隆模式，可以创建一个兵种的原型，然后根据需要快速地克隆出大量小兵。

首先，定义一个 Manager 类，用于存储和克隆原型：

```java
import java.util.HashMap;
import java.util.Map;

2 个用法
public class Manager {
    2 个用法
    private final Map<String, GameCharacter> showcase = new HashMap<>();

    3 个用法
    public void register(String name, GameCharacter proto) {
        showcase.put(name, proto);
    }

    4 个用法
    public GameCharacter create(String characterName) {
        GameCharacter character = showcase.get(characterName);
        return character.createClone();
    }
}
```

接下来，定义 GameCharacter 接口和 Character 类：

```java
public interface GameCharacter extends Cloneable {
    4 个用法  3 个实现
    void attack();
    1 个用法  1 个实现
    GameCharacter createClone();
}
```

```java
public abstract class Character implements GameCharacter {
    3 个用法
    private String name;
    2 个用法
    private int HP;
    3 个用法
    private int ATK;
    3 个用法
    public Character(String name, int HP, int ATK) {
        this.name = name;
        this.HP = HP;
        this.ATK = ATK;
    }
```

```java
    public void attack() {
        System.out.println(getName() + " 发动攻击，攻击力为 " + getATK());
    }
    1 个用法
    public GameCharacter createClone() {
        try {
            return (GameCharacter) super.clone();
        } catch (CloneNotSupportedException e) {
            e.printStackTrace();
            return null;
        }
    }
```

然后，定义具体的兵种类（近战兵、法师、弓箭手）：

```java
public class Warrior extends Character {
    1 个用法
    public Warrior(String name, int HP, int ATK) {
        super(name, HP, ATK);
    }
}
```

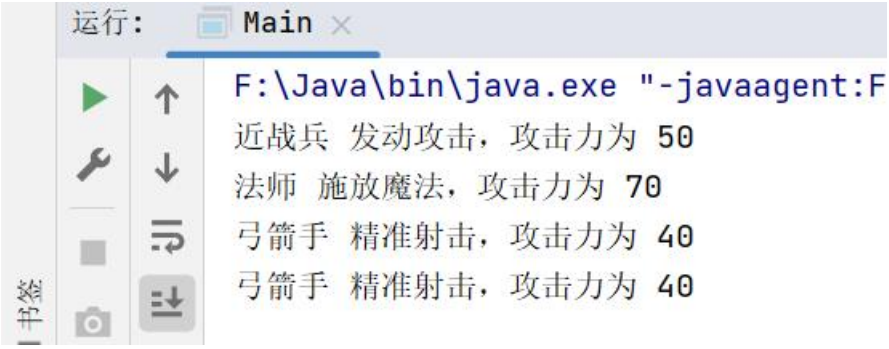
```java
public class Archer extends Character {
    1 个用法
    public Archer(String name, int HP, int ATK) {
        super(name, HP, ATK);
    }

    4 个用法
    @Override
    public void attack() {
        System.out.println(getName() + " 精准射击，攻击力为 " + getATK());
    }
}
```

```java
public class Mage extends Character {
    1 个用法
    public Mage(String name, int HP, int ATK) {
        super(name, HP, ATK);
    }

    4 个用法
    @Override
    public void attack() {
        System.out.println(getName() + " 施放魔法，攻击力为 " + getATK());
    }
}
```

最后，在 Main 类中使用 Manager 来创建和管理士兵：

```java
public class Main {
    0 个用法
    public static void main(String[] args) {
        Manager manager = new Manager();
        manager.register( name: "warrior", new Warrior( name: "近战兵",  HP: 150,  ATK: 50));
        manager.register( name: "mage", new Mage( name: "法师",  HP: 120,  ATK: 70));
        manager.register( name: "archer", new Archer( name: "弓箭手",  HP: 100,  ATK: 40));

        GameCharacter warrior = manager.create("warrior");
        warrior.attack();

        GameCharacter mage = manager.create("mage");
        mage.attack();

        GameCharacter archer1 = manager.create("archer");
        archer1.attack();

        GameCharacter archer2 = manager.create("archer");
        archer2.attack();
    }
}
```

运行效果：



2、 试描述浅克隆和深克隆。

答：

- 浅克隆

  浅克隆仅复制对象的基本数据类型的值和引用类型的引用，但不复制引用对象本身。如果原型对象包含引用其他对象的字段，那么克隆出的新对象中这些字段引用的仍然是同一个对象。

  在第一题的示例中，如果 Character 类包含了引用类型的字段（如装备、技能等），那么使用浅克隆方法只会复制这些字段的引用，不会创建这些对象的副本。

- 深克隆

  深克隆不仅复制对象的基本数据类型的值，还会递归地复制它所引用的所有对象，从而创建所有层次的独立副本。这意味着克隆出的对象和原始对象在引用类型的数据上完全独立，修改一个不会影响到另一个。

  在第一题的示例中，如果需要实现每个角色的装备和技能在被克隆时也是独立的，那么就需要实现深克隆。

3、 附录

1） Maneger

```
import java.util.HashMap;
import java.util.Map;

public class Manager {
    private final Map<String, GameCharacter> showcase = new HashMap<>();

    public void register(String name, GameCharacter proto) {
        showcase.put(name, proto);
    }

    public GameCharacter create(String characterName) {
        GameCharacter character = showcase.get(characterName);
        return character.createClone();
    }
}
```

2） Character

```
public abstract class Character implements GameCharacter {
    private String name;
    private int HP;
    private int ATK;
    public Character(String name, int HP, int ATK) {
        this.name = name;
        this.HP = HP;
        this.ATK = ATK;
    }
    public void attack() {
        System.out.println(getName() + " 发动攻击，攻击力为 " +
getATK());
    }
    public GameCharacter createClone() {
        try {
            return (GameCharacter) super.clone();
        } catch (CloneNotSupportedException e) {
            e.printStackTrace();
            return null;
        }
    }

    public String getName() {
        return name;
    }
```

```java
    public void setName(String name) {
        this.name = name;
    }

    public void setHP(int HP) {
        this.HP = HP;
    }

    public void setATK(int ATK) {
        this.ATK = ATK;
    }

    public int getATK() {
        return ATK;
    }
}
```

3) GameCharacter

```java
public interface GameCharacter extends Cloneable {
    void attack();
    GameCharacter createClone();
}
```

4) Warrior

```java
public class Warrior extends Character {
    public Warrior(String name, int HP, int ATK) {
        super(name, HP, ATK);
    }
}
```

5) Mage

```java
public class Mage extends Character {
    public Mage(String name, int HP, int ATK) {
        super(name, HP, ATK);
    }

    @Override
    public void attack() {
        System.out.println(getName() + " 施放魔法，攻击力为 " +
getATK());
    }
}
```

6) Archer

```java
public class Archer extends Character {
    public Archer(String name, int HP, int ATK) {
        super(name, HP, ATK);
    }

    @Override
    public void attack() {
        System.out.println(getName() + " 精准射击，攻击力为 " +
getATK());
    }
}
```

7) Main

```java
public class Main {
    public static void main(String[] args) {
        Manager manager = new Manager();
        manager.register("warrior", new Warrior("近战兵", 150, 50));
        manager.register("mage", new Mage("法师", 120, 70));
        manager.register("archer", new Archer("弓箭手", 100, 40));

        GameCharacter warrior = manager.create("warrior");
        warrior.attack();

        GameCharacter mage = manager.create("mage");
        mage.attack();

        GameCharacter archer1 = manager.create("archer");
        archer1.attack();

        GameCharacter archer2 = manager.create("archer");
        archer2.attack();
    }
}
```