1. 请举例说明克隆模式的其他应用

答：在游戏开发中，玩家创建角色时 often 需要基于预设模板（如 “战士”“法师”）快速生成角色实例，通过克隆模板并修改个性化属性（如姓名、等级）来提高效率。

Code：

// 抽象原型类

class GameCharacter implements Cloneable {

protected String role; // 角色职业（模板属性）

protected String name; // 角色姓名（个性化属性）

protected int level; // 角色等级（个性化属性）

// 克隆方法（浅克隆）

@Override

protected GameCharacter clone() {

GameCharacter clone = null;

try {

clone = (GameCharacter) super.clone();

} catch (CloneNotSupportedException e) {

e.printStackTrace();

}

return clone;

}

// 通用方法

public void display() {

System.out.println("职业：" + role + ", 姓名：" + name + ", 等级：" + level);

}

}

// 具体原型类：战士

class Warrior extends GameCharacter {

public Warrior() {

this.role = "战士";

this.level = 1; // 初始等级

}

}

// 具体原型类：法师

class Mage extends GameCharacter {

public Mage() {

this.role = "法师";

this.level = 1; // 初始等级

}

}

// 客户端代码

public class PrototypeExample {

public static void main(String[] args) {

// 预设原型模板

GameCharacter warriorTemplate = new Warrior();

GameCharacter mageTemplate = new Mage();

// 克隆战士并设置个性化属性

GameCharacter player1 = warriorTemplate.clone();

player1.name = "亚瑟";

player1.level = 5;

// 克隆法师并设置个性化属性

GameCharacter player2 = mageTemplate.clone();

player2.name = "吉安娜";

player2.level = 3;

// 显示角色信息

player1.display(); // 输出：职业：战士, 姓名：亚瑟, 等级：5

player2.display(); // 输出：职业：法师, 姓名：吉安娜, 等级：3

}

}

1. 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成员和引用地址，不复制引用指向的对象（即引用类型成员共享内存地址），我们可以通过实现Cloneable接口并重写clone()方法，或者使用super.clone()完成默认克隆；**深克隆**：深克隆递归复制对象的所有层级成员，包括基本类型和引用类型指向的对象，确保克隆后的对象与原对象完全独立。