### JAVA 第一阶段—DAY11-JAVA案例

1. 使用模板设计模式，完成一块猪肉和大象放入冰箱的操作。

**分析：1、定义一个抽象类，提供几个共同的且无需子类详细实现的方法；**

1. **在抽象类中，再定义一些需要子类详细实现的方法；**
2. **提供一个对外访问实现的模板方法；**
3. **子类继承抽象类，只需重写需要详细实现的方法即可。**

* 参考答案

// 1、抽象类

public abstract class AbstractClass{

public final void execute(){

open();

put();

close();

}

private void open(){

System.out.println("打开冰箱");

}

private void close(){

System.out.println("关闭冰箱");

}

protected abstract void put();

}

// 子类实现

public class Pork extends AbstractClass{

@Override

protected void put(){

System.out.println("将一块猪肉装进冰箱：直接装啊");

}

}

public class Elephant extends AbstractClass{

@Override

protected void put(){

System.out.println("将大象装入冰箱：你必须剁碎再装入");

}

}

// 测试类

public class Demo {

public static void main(String[] args){

AbstractClass abstractClass = new Pork();

abstractClass.execute();

System.out.println("-------------------");

abstractClass = new Elephant();

abstractClass.execute();

}

}

1. 以接口形参的形式传入对应的子类对象。

* 参考答案

// 接口A

public abstract class Animal {

public abstract void eat();

}

// 实现类

public class Cat extends Animal {

public void eat() {

System.out.println("吃鱼");

}

}

public class Dog extends Animal {

public void eat() {

System.out.println("吃骨头");

}

}

// 测试

public class MyTest2 {

public static void main(String[] args) {

// 多态形式，创建对象

Cat c = new Cat();

Dog d = new Dog();

// 调用showCatEat

showCatEat(c);

// 调用showDogEat

showDogEat(d);

// 以上两个方法, 均可以被showAnimalEat(Animal a)方法所替代 而执行效果一致

showAnimalEat(c);

showAnimalEat(d);

}

public static void showCatEat (Cat c){

c.eat();

}

public static void showDogEat (Dog d){

d.eat();

}

public static void showAnimalEat (Animal a){

a.eat();

}

}

1. 对上述接口参数改用匿名内部类的形式实现。

* 参考答案

// 测试类

public class MyTest2 {

public static void main(String[] args) {

// 匿名内部类调用

showAnimalEat(new Animal() {

@Override

public void eat() {

System.out.println("动物吃东西");

}

});

}

public static void showAnimalEat(Animal a){

a.eat();

}

}