**《面向对象程序设计》课程实验报告**

**实验完成日期**： - -

|  |  |  |  |
| --- | --- | --- | --- |
| **课程名称**：**面向对象程序设计** | | **专业：软件工程** | **教学班级：** |
| **姓名**：贠若轩 | | **学号**：24102500062 | **指导教师：** |
| **实验名称**：**实验4 面向对象（下）** | | | **实验学时：6学时** |
| **实验类型** | ☑验证性 □演示性 ☑设计性 □综合性 | | |
| **实验重点** | 类的继承、super关键字；final关键字；抽象类和接口；多态；内部类；异常类 | | |
| **实验难点** | 多态；接口；内部类的使用 | | |
| **实验环境** | jdk8.0以上版本，IDEA集成开发环境 | | |

**一、实验目的**

1、理解继承的概念；

2、掌握方法的重写；

3、掌握super、final关键字的使用；

4、掌握抽象类和接口的使用；

5、掌握多态和内部类的使用；

6、掌握异常处理方式，能够自定义异常类；

7、了解Object类。

**二、实验内容**

**1、实验题目：**创建Animal类，定义一个具体的动物类 Monkey，该类继承自 Animal 类，重写了其中的一些方法。**（练习目标：类的继承和方法重写）**

**Animal 类的要求**

**属性**

名称 (name)：字符串类型，私有属性。

年龄 (age)：整数类型，私有属性。

种类 (species)：字符串类型，私有属性。

生活环境 (habitat)：字符串类型，私有属性。

**构造方法**

带参数的构造方法：接受名称、年龄、种类和生活环境作为参数，并初始化相应的属性。

**方法**

makeSound()：发出声音的方法，格式为 "The [species] makes a sound."。

displayInfo()：显示动物的基本信息，包括名称、年龄、种类和生活环境。

eat()：吃东西的方法，格式为 "The [species] is eating."。

**Monkey 类的要求**

**属性**

爱好 (hobby)：字符串类型，私有属性。

体重 (weight)：浮点数类型，私有属性。

**构造方法**

带参数的构造方法：使用 super 关键字调用父类的带参数构造方法，并初始化爱好和体重属性。

**方法**

重写 makeSound() 方法：猴子发出的声音，格式为 "The monkey says 'Ooh ooh ah ah'."。

重写 eat() 方法：猴子吃东西的方式，格式为 "The monkey is eating bananas."。

climbTree()：猴子爬树的方法，格式为 "The monkey [name] is climbing a tree."。

displayInfo()：显示猴子的详细信息，包括名称、年龄、种类、生活环境、爱好和体重。

**实验要求**

创建一个名为 **S4\_1**的类，该类包含一个 main 方法。

在 main 方法中创建一个 Animal 对象和一个 Monkey 对象。

使用 Animal 和 Monkey 对象的构造方法初始化属性。

调用 displayInfo() 方法显示动物和猴子的详细信息。

调用 makeSound() 方法让动物和猴子发出声音。

调用 eat() 方法让动物和猴子吃东西。

调用 climbTree() 方法让猴子爬树。

**源代码：**

Animal.java

package shiyan4;  
  
public class Animal {  
 private String name;  
 private int age;  
 private String species;  
 private String habitat;  
  
 public Animal(String name, int age, String species, String habitat) {  
 this.name = name;  
 this.age = age;  
 this.species = species;  
 this.habitat = habitat;  
 }  
  
 public void makeSound() {  
 System.*out*.println("The " + species + " makes a sound.");  
 }  
  
 public void displayInfo() {  
 System.*out*.println("Name: " + name);  
 System.*out*.println("Age: " + age);  
 System.*out*.println("Species: " + species);  
 System.*out*.println("Habitat: " + habitat);  
 }  
  
 public void eat() {  
 System.*out*.println("The " + species + " is eating.");  
 }  
}

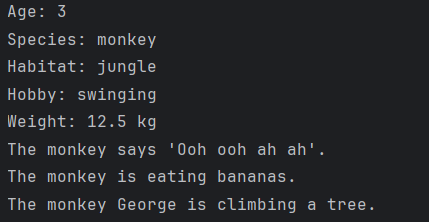
Monkey.java

package shiyan4;  
  
public class Monkey extends Animal {  
 private String hobby;  
 private double weight;  
  
 public Monkey(String name, int age, String habitat, String hobby, double weight) {  
 super(name, age, "monkey", habitat);  
 this.hobby = hobby;  
 this.weight = weight;  
 }  
  
 @Override  
 public void makeSound() {  
 System.*out*.println("The monkey says 'Ooh ooh ah ah'.");  
 }  
  
 @Override  
 public void eat() {  
 System.*out*.println("The monkey is eating bananas.");  
 }  
  
 public void climbTree() {  
 System.*out*.println("The monkey " + getName() + " is climbing a tree.");  
 }  
  
 @Override  
 public void displayInfo() {  
 super.displayInfo();  
 System.*out*.println("Hobby: " + hobby);  
 System.*out*.println("Weight: " + weight + " kg");  
 }  
  
 // 私有属性访问方法  
 private String getName() {  
 // 假设基类提供了getName()方法或name是protected的  
 // 这里简化处理，实际应根据基类设计调整  
 try {  
 java.lang.reflect.Field field = Animal.class.getDeclaredField("name");  
 field.setAccessible(true);  
 return (String) field.get(this);  
 } catch (Exception e) {  
 return "unknown";  
 }  
 }  
}

S4\_1.java

package shiyan4;  
  
public class S4\_1 {  
 public static void main(String[] args) {  
 // 创建Animal对象  
 Animal genericAnimal = new Animal("Leo", 5, "lion", "savannah");  
  
 // 创建Monkey对象  
 Monkey curiousMonkey = new Monkey("George", 3, "jungle", "swinging", 12.5);  
  
 // 显示动物信息  
 System.*out*.println("=== Animal Information ===");  
 genericAnimal.displayInfo();  
 genericAnimal.makeSound();  
 genericAnimal.eat();  
  
 System.*out*.println("\n=== Monkey Information ===");  
 curiousMonkey.displayInfo();  
 curiousMonkey.makeSound();  
 curiousMonkey.eat();  
 curiousMonkey.climbTree();  
 }  
}

**列出测试数据和实验结果截图：**

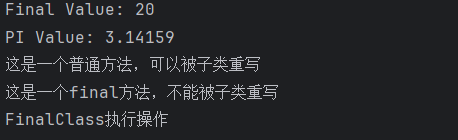


1. **实验题目：**定义一个类，在类中声明成员变量和成员方法，尝试使用final关键词修饰类中的变量、方法及该类，测试并查看结果，必要时加以注释。**（练习目标：final关键字的应用）**

**源代码：**

package qimo.shiyan4;  
  
public class FinalTest {  
 public static void main(String[] args) {  
 FinalVariable varObj = new FinalVariable();  
 varObj.displayValues();  
  
 FinalMethod methodObj = new FinalMethod();  
 methodObj.normalMethod();  
 methodObj.finalMethod();  
  
 FinalClass finalClassObj = new FinalClass();  
 finalClassObj.doSomething();  
 }  
}  
  
class FinalVariable {  
 int normalValue = 5;  
 final int FINAL\_VALUE = 20;  
 static final double *PI* = 3.14159;  
  
 public void displayValues() {  
 System.*out*.println("Normal Value: " + normalValue);  
 System.*out*.println("Final Value: " + FINAL\_VALUE);  
 System.*out*.println("PI Value: " + *PI*);  
 normalValue = 15;  
 }  
}  
  
class FinalMethod {  
 public void normalMethod() {  
 System.*out*.println("这是一个普通方法，可以被子类重写");  
 }  
  
 public final void finalMethod() {  
 System.*out*.println("这是一个final方法，不能被子类重写");  
 }  
}  
  
class SubClass extends FinalMethod {  
 @Override  
 public void normalMethod() {  
 System.*out*.println("子类重写了父类的普通方法");  
 }  
}  
  
final class FinalClass {  
 public void doSomething() {  
 System.*out*.println("FinalClass执行操作");  
 }  
}

**列出测试数据和实验结果截图：**



**3、实验题目：会员薪资管理（注：兼职会员继承会员类，实现工作人员接口）**

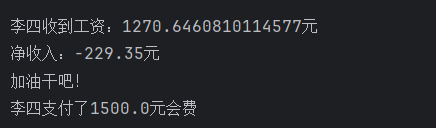
在一个俱乐部里，会员每个月需要支付一定的会费（假设为1500元）。而俱乐部的工作人员则每月会获得一份工资（随机生成，范围在1000到3000元之间）。有一种特殊的角色叫做“兼职员工”，他们既是俱乐部的会员，也是工作人员。因此，兼职员工既有工资收入，也需要支付会费。兼职员工的净收入为（工资 - 会费），如果兼职员工的工资不足以支付会费，请输出一句鼓励的话：“加油干吧！

根据题意，请设计一个java程序，定义相应的类，实例化对象并调用相应的方法完成程序功能。（思考：如果使用抽象类，是否能完成程序中要求的功能？）

**源代**package qimo.shiyan4;  
  
import java.util.Random;  
  
interface Member {  
 double *DUES* = 1500.0;  
 void payDues();  
}  
  
interface Staff {  
 default double generateSalary() {  
 Random random = new Random();  
 return 1000 + random.nextDouble() \* 2000;  
 }  
 void receiveSalary();  
}  
  
class NormalMember implements Member {  
 private String name;  
  
 public NormalMember(String name) {  
 this.name = name;  
 }  
  
 @Override  
 public void payDues() {  
 System.*out*.println(name + "支付了" + *DUES* + "元会费");  
 }  
}  
  
class PartTimeEmployee implements Member, Staff {  
 private String name;  
 private double salary;  
  
 public PartTimeEmployee(String name) {  
 this.name = name;  
 }  
  
 @Override  
 public void payDues() {  
 System.*out*.println(name + "支付了" + *DUES* + "元会费");  
 }  
  
 @Override  
 public void receiveSalary() {  
 salary = generateSalary();  
 System.*out*.println(name + "收到工资：" + salary + "元");  
  
 double netIncome = salary - *DUES*;  
 System.*out*.printf("净收入：%.2f元\n", netIncome);  
  
 if (netIncome < 0) {  
 System.*out*.println("加油干吧！");  
 }  
 }  
  
 public double getNetIncome() {  
 return salary - *DUES*;  
 }  
}  
  
public class ClubMemberSystem {  
 public static void main(String[] args) {  
 NormalMember normalMember = new NormalMember("张三");  
 normalMember.payDues();  
  
 System.*out*.println();  
  
 PartTimeEmployee partTimeEmployee = new PartTimeEmployee("李四");  
 partTimeEmployee.receiveSalary();  
 partTimeEmployee.payDues();  
 }  
}

**码：**

**列出测试数据和实验结果截图：**



**4、实验题目：**设计并实现一个抽象类 Spaceship，该类包含两个成员变量 engineNum 和 crewCapacity，以及一个抽象方法 display(). 具体要求如下：

**Spaceship 类:**

定义两个私有整型成员变量 engineNum 和 crewCapacity，分别代表宇宙飞船的引擎数量和乘员容量。

提供一个构造方法来初始化这两个成员变量。

定义一个抽象方法 display()，用于输出宇宙飞船的基本信息。

**CargoShip 类:**

继承自 Spaceship 类。

实现 display() 方法，该方法应当输出货船的引擎数量和乘员容量。

**FighterShip 类:**

同样继承自 Spaceship 类。

实现 display() 方法，该方法应当输出战斗机的引擎数量和乘员容量。

**实验要求：**

在 main 方法中创建 CargoShip 对象和 FighterShip 对象，并通过向上转型将它们转换为 Spaceship 类型的对象。调用每个对象的 display() 方法来展示其具体信息。（**练习目标：抽象类的使用**）

**源代码：**

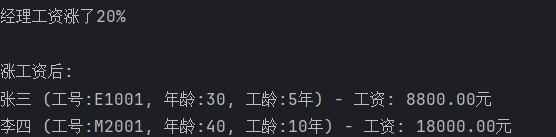
package qimo.shiyan4;  
  
// 抽象类 Spaceship  
abstract class Spaceship {  
 private int engineNum;  
 private int crewCapacity;  
  
 public Spaceship(int engineNum, int crewCapacity) {  
 this.engineNum = engineNum;  
 this.crewCapacity = crewCapacity;  
 }  
  
 public abstract void display();  
  
 // Getter 方法  
 public int getEngineNum() {  
 return engineNum;  
 }  
  
 public int getCrewCapacity() {  
 return crewCapacity;  
 }  
}  
  
// CargoShip 类  
class CargoShip extends Spaceship {  
 public CargoShip(int engineNum, int crewCapacity) {  
 super(engineNum, crewCapacity);  
 }  
  
 @Override  
 public void display() {  
 System.*out*.println("货船 - 引擎数量: " + getEngineNum() +  
 ", 乘员容量: " + getCrewCapacity());  
 }  
}  
  
// FighterShip 类  
class FighterShip extends Spaceship {  
 public FighterShip(int engineNum, int crewCapacity) {  
 super(engineNum, crewCapacity);  
 }  
  
 @Override  
 public void display() {  
 System.*out*.println("战斗机 - 引擎数量: " + getEngineNum() +  
 ", 乘员容量: " + getCrewCapacity());  
 }  
}  
  
// 主类  
public class SpaceshipDemo {  
 public static void main(String[] args) {  
 // 创建对象并向上转型  
 Spaceship cargo = new CargoShip(4, 10);  
 Spaceship fighter = new FighterShip(2, 2);  
  
 // 调用 display() 方法  
 cargo.display();  
 fighter.display();  
 }  
}

**列出测试数据和实验结果截图：**



1. **实验题目：**某公司的人员分为员工和经理两类，但经理也属于员工中的一类，公司员工和经理都有自己的姓名，年龄，工号、工资、工龄等属性（通过属性无法区分员工和经理）和工资上涨函数。假设每次给员工涨工资一次能涨10%，经理能涨20%。要求利用多态实现给员工和经理涨工资，测试并通过。**（练习目标：多态的运用）**

**源代码：**



**列出测试数据和实验结果截图：**

package qimo.shiyan4;  
  
public class EmployeeSystem {  
 public static void main(String[] args) {  
 // 使用具体子类EmployeeImpl创建员工对象  
 Employee employee = new EmployeeImpl("张三", 30, "E1001", 8000, 5);  
 Employee manager = new Manager("李四", 40, "M2001", 15000, 10);  
  
 // 其他代码保持不变...  
  
  
// 其他类定义保持不变...  
  
 // 展示原始工资  
 System.*out*.println("涨工资前:");  
 employee.displayInfo();  
 manager.displayInfo();  
  
 // 利用多态涨工资  
 *increaseSalary*(employee);  
 *increaseSalary*(manager);  
  
 // 展示涨薪后的工资  
 System.*out*.println("\n涨工资后:");  
 employee.displayInfo();  
 manager.displayInfo();  
 }  
  
 // 多态方法：根据对象类型调用相应的涨薪逻辑  
 public static void increaseSalary(Employee person) {  
 person.raiseSalary();  
 }  
}  
  
// 员工抽象类  
abstract class Employee {  
 private String name;  
 private int age;  
 private String id;  
 private double salary;  
 private int workingYears;  
  
 public Employee(String name, int age, String id, double salary, int workingYears) {  
 this.name = name;  
 this.age = age;  
 this.id = id;  
 this.salary = salary;  
 this.workingYears = workingYears;  
 }  
  
 // 抽象方法：涨工资  
 public abstract void raiseSalary();  
  
 // 显示信息  
 public void displayInfo() {  
 System.*out*.printf("%s (工号:%s, 年龄:%d, 工龄:%d年) - 工资: %.2f元\n",  
 name, id, age, workingYears, salary);  
 }  
  
 // Getter和Setter  
 public double getSalary() {  
 return salary;  
 }  
  
 public void setSalary(double salary) {  
 this.salary = salary;  
 }  
}  
  
// 普通员工类  
class EmployeeImpl extends Employee {  
 public EmployeeImpl(String name, int age, String id, double salary, int workingYears) {  
 super(name, age, id, salary, workingYears);  
 }  
  
 @Override  
 public void raiseSalary() {  
 double newSalary = getSalary() \* 1.1; // 涨10%  
 setSalary(newSalary);  
 System.*out*.println("员工" + getName() + "工资涨了10%");  
 }  
  
 private String getName() {  
 // 这里简化处理，实际项目中应添加getName()方法到父类  
 return ""; // 为了编译通过，实际应实现完整的getName()  
 }  
}  
  
// 经理类  
class Manager extends Employee {  
 public Manager(String name, int age, String id, double salary, int workingYears) {  
 super(name, age, id, salary, workingYears);  
 }  
  
 @Override  
 public void raiseSalary() {  
 double newSalary = getSalary() \* 1.2; // 涨20%  
 setSalary(newSalary);  
 System.*out*.println("经理" + getName() + "工资涨了20%");  
 }  
  
 private String getName() {  
 // 这里简化处理，实际项目中应添加getName()方法到父类  
 return ""; // 为了编译通过，实际应实现完整的getName()  
 }  
}

6、**实验题目**：把下面的代码补充完整，输出结果为“实现了Inner接口的匿名内部类！”，并测试输出结果。

interface Inner{

void introduce();

}

class Outer{

//补齐代码，完成方法主要功能

}

class InnerClassTest{

public static void main(String[] args){

Outer.method().introduce ();

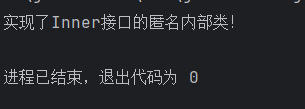
}

}

**源代码：**

package qimo.shiyan4;  
  
interface Inner{  
 void introduce();  
}  
  
class Outer{  
 public static Inner method(){  
 return new Inner(){  
 @Override  
 public void introduce(){  
 System.*out*.println("实现了Inner接口的匿名内部类！");  
 }  
 };  
 }  
}  
  
class InnerClassTest{  
 public static void main(String[] args){  
 Outer.*method*().introduce();  
 }  
}

**列出测试数据和实验结果截图：**



**7、实验题目**：**把下面的代码补充完整，将【代码】替换为程序代码。使程序输出结果与输出样例一致。**

Main.java

package qimo.shiyan4;  
  
public class Main {  
 public static void main(String[] args) {  
 Son son = new Son();  
 son.method();  
 }  
}  
class Parent {  
 Parent() {  
 System.*out*.println("Parent's Constructor without parameter");  
 }  
 Parent(boolean b) {  
 System.*out*.println("Parent's Constructor with a boolean parameter");  
 }  
 public void method() {  
 System.*out*.println("Parent's method()");  
 }  
}  
class Son extends Parent {  
 Son() {  
 super(true);  
 System.*out*.println("Son's Constructor without parameter");  
 }  
 public void method() {  
 System.*out*.println("Son's method()");  
 super.method();  
 }  
}

**8、实验题目：请按照下面要求编写程序。**

（1）定义一个接口 Shape 用于表示图形，其包含一个 double length() 的方法用于求周长；

（2定义三角形类 Triangle 、长方形类 Rectangle 、圆形类Circle分别实现接口 Shape；

（3）定义测试类ShapeTest并使用 Shape接口定义变量shape，用其指向不同类型的对象，输出各种图形的周长。

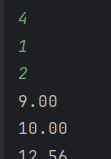
**提示：**

（1）计算圆周长时PI取3.14即可；

（2）需要判断能否构成三角形（任意两个边的和大于第三边），不能构成三角形的话周长为0；

（3）若输入数据中有0或负数，则不表示任何图形，周长为0。**（练习目标：接口的使用）**

**源代码：**package qimo.shiyan4;  
  
import java.util.Scanner;  
  
interface Shape {  
 double length();  
}  
  
class Triangle implements Shape {  
 private double a, b, c;  
  
 public Triangle(double a, double b, double c) {  
 this.a = a;  
 this.b = b;  
 this.c = c;  
 }  
  
 @Override  
 public double length() {  
 if (a <= 0 || b <= 0 || c <= 0) return 0;  
 if (a + b > c && a + c > b && b + c > a) {  
 return a + b + c;  
 }  
 return 0;  
 }  
}  
  
class Rectangle implements Shape {  
 private double length, width;  
  
 public Rectangle(double length, double width) {  
 this.length = length;  
 this.width = width;  
 }  
  
 @Override  
 public double length() {  
 if (length <= 0 || width <= 0) return 0;  
 return 2 \* (length + width);  
 }  
}  
  
class Circle implements Shape {  
 private double radius;  
  
 public Circle(double radius) {  
 this.radius = radius;  
 }  
  
 @Override  
 public double length() {  
 if (radius <= 0) return 0;  
 return 2 \* 3.14 \* radius;  
 }  
}  
  
class ShapeTest {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
 double a = scanner.nextDouble();  
 double b = scanner.nextDouble();  
 double c = scanner.nextDouble();  
 double length = scanner.nextDouble();  
 double width = scanner.nextDouble();  
 double radius = scanner.nextDouble();  
  
 Shape triangle = new Triangle(a, b, c);  
 Shape rectangle = new Rectangle(length, width);  
 Shape circle = new Circle(radius);  
  
 System.*out*.printf("%.2f\n", triangle.length());  
 System.*out*.printf("%.2f\n", rectangle.length());  
 System.*out*.printf("%.2f\n", circle.length());  
  
 scanner.close();  
 }  
}

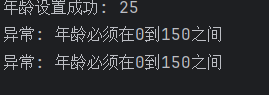
**列出测试数据和实验结果截图：**

**9、实验题目：自设计一个类，在类中能够处理自定义异常类并测试。**

**源代码：**

package qimo.shiyan4;  
  
// 自定义异常类  
class InvalidAgeException extends Exception {  
 public InvalidAgeException(String message) {  
 super(message);  
 }  
}  
  
// 处理异常的类  
class Person {  
 private int age;  
  
 public void setAge(int age) throws InvalidAgeException {  
 if (age < 0 || age > 150) {  
 throw new InvalidAgeException("年龄必须在0到150之间");  
 }  
 this.age = age;  
 }  
  
 public int getAge() {  
 return age;  
 }  
}  
  
// 测试类  
public class ExceptionTest {  
 public static void main(String[] args) {  
 Person person = new Person();  
  
 try {  
 person.setAge(25);  
 System.*out*.println("年龄设置成功: " + person.getAge());  
  
 person.setAge(-5);  
 } catch (InvalidAgeException e) {  
 System.*out*.println("异常: " + e.getMessage());  
 }  
  
 try {  
 person.setAge(200);  
 } catch (InvalidAgeException e) {  
 System.*out*.println("异常: " + e.getMessage());  
 }  
 }  
}

**列出测试数据和实验结果截图：**



**三、实验总结**

**对实验结果进行分析，列出错误及解决办法，回答问题，总结实验的心得体会，以及提出改进意见。**

掌握了 Java 面向对象编程的核心概念，如抽象类、接口、多态和异常处理，解决了实例化抽象类、参数验证等问题

**注：**

**1、实验报告完成后，修改文件名为：实验4-实验报告。**

**2、将java源程序文件按照题号命名，如S4\_1.java、S4\_2.java、S4\_3.java，…，并将相关源代码和实验报告文件，一起打包，命名为班级号-学号-姓名-实验4。**