day11_inner_王世杰

1.简答题

1. 总结成员内部类和静态内部类的成员特点和访问特点。（尽量简洁的用一句话描述）

```
(1)成员内部类
    (a)成员特点:和普通类基本一样除了不能有静态声明,但可以有静态全局常量.
    (b)访问特点:内部类和外围类访问不受限制,内部类和外部类之间依赖外围类对象.
(2)静态内部类的成员特点和普通类的成员一样，如果想要访问一个静态内部类。首先需要外围类权限，然后还需要静态内部类权限。
```

2. 什么是功能接口?
    1. 功能接口中只能有一个方法吗?

    ```
    非也,可以有实现好的方法.
    ```

    2. 功能接口中只能有一个抽象方法吗?

    ```
    非也,可以有Object已经实现的方法的抽象方法.
    ```

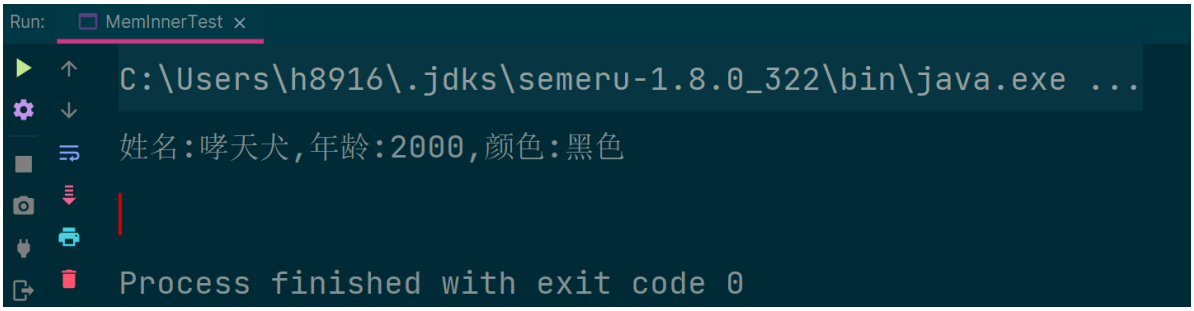
2.成员内部类练习

成员内部类对象依赖于它的外围类对象

> 定义一个类Dog
> 属性：age,name
> 除此之外,Dog类中需要定义一个成员内部类Body,Body类中有属性color
> 请私有化该成员内部类,然后将该成员内部类对象作为外围类的成员变量私有化。
> 最后在Dog类提供一个方法,展示Dog类的全部属性（age、name和color）
> 注意：
> 　既然定义成员内部类，就应该让外部感受不到这个内部类
> 　思考，如何在外部创建Dog类对象且给属性color赋值的情况下，实现这一点?

```java
package com.cskaoyan.homework.day11;

public class MemInnerTest {
```

```java
    public static void main(String[] args) {
        new Dog("哮天犬",2000,"黑色").show();

    }
}

class Dog {
    String name;
    int age;
    Body body = new Body();

    private class Body {
        String color;

        public void setColor(String color) {
            this.color = color;
        }
    }

    public Dog(String name, int age,String color) {
        this.name = name;
        this.age = age;
        body.setColor(color);
    }

    public void show() {
        System.out.println("姓名:" + name + ",年龄:" + age + ",颜色:" +
body.color);
    }
}
```



3.用内部类来实现接口

定义一个接口Compute，用来完成计算器的功能，给出四个抽象方法：加减乘除。
然后请用以下三种方式测试：

1，编写实现类进行测试
2，用局部内部类进行测试
3，使用匿名内部类进行测试

```java
package com.cskaoyan.homework.day11;

public class InterfaceInnerTest {
    public static void main(String[] args) {
        System.out.println("============================");
        System.out.println("1，编写实现类进行测试");
        ComputerImpl com = new ComputerImpl();
        System.out.println("10 + 10 = " + com.plus(10, 10));
        System.out.println("10 - 10 = " + com.subtract(10, 10));
        System.out.println("10 x 10 = " + com.multiply(10, 10));
        System.out.println("10 ÷ 10 = " + com.divide(10, 10));
        System.out.println("============================");
        System.out.println("2.用局部内部类进行测试");
        class ComputerImpl1 implements Computer {
            @Override
            public double plus(double a, double b) {
                return a + b;
            }

            @Override
            public double subtract(double a, double b) {
                return a - b;
            }

            @Override
            public double multiply(double a, double b) {
                return a * b;
            }

            @Override
            public double divide(double a, double b) {
                return a / b;
            }
        }
        ComputerImpl1 computerImpl1 = new ComputerImpl1();
        System.out.println("10 + 10 = " + computerImpl1.plus(10, 10));
        System.out.println("10 - 10 = " + computerImpl1.subtract(10,
10));
        System.out.println("10 x 10 = " + computerImpl1.multiply(10,
10));
        System.out.println("10 ÷ 10 = " + computerImpl1.divide(10,
10));

        System.out.println("============================");
        System.out.println("3，使用匿名内部类进行测试");
        System.out.println("10 + 10 = " + new Computer() {
            @Override
            public double plus(double a, double b) {
                return a + b;
```

```
        }

        @Override
        public double subtract(double a, double b) {
            return a - b;
        }

        @Override
        public double multiply(double a, double b) {
            return a * b;
        }

        @Override
        public double divide(double a, double b) {
            return a / b;
        }

    }.plus(10, 10));

    System.out.println("10 - 10 = " + new Computer() {
        @Override
        public double plus(double a, double b) {
            return a + b;
        }

        @Override
        public double subtract(double a, double b) {
            return a - b;
        }

        @Override
        public double multiply(double a, double b) {
            return a * b;
        }

        @Override
        public double divide(double a, double b) {
            return a / b;
        }

    }.subtract(10, 10));

    System.out.println("10 x 10 = " + new Computer() {
        @Override
        public double plus(double a, double b) {
            return a + b;
        }

        @Override
```

```java
            public double subtract(double a, double b) {
                return a - b;
            }

            @Override
            public double multiply(double a, double b) {
                return a * b;
            }

            @Override
            public double divide(double a, double b) {
                return a / b;
            }

        }.multiply(10, 10));

        System.out.println("10 ÷ 10 = " + new Computer() {
            @Override
            public double plus(double a, double b) {
                return a + b;
            }

            @Override
            public double subtract(double a, double b) {
                return a - b;
            }

            @Override
            public double multiply(double a, double b) {
                return a * b;
            }

            @Override
            public double divide(double a, double b) {
                return a / b;
            }

        }.divide(10, 10));
    }
}

class ComputerImpl implements Computer {

    @Override
    public double plus(double a, double b) {
        return a + b;
    }

    @Override
```

```java
    public double subtract(double a, double b) {
        return a - b;
    }

    @Override
    public double multiply(double a, double b) {
        return a * b;
    }

    @Override
    public double divide(double a, double b) {
        return a / b;
    }
}

interface Computer {
    double plus(double a, double b);

    double subtract(double a, double b);

    double multiply(double a, double b);

    double divide(double a, double b);
}
```

```
Run:  InterfaceInnerTest ×
C:\Users\h8916\.jdks\semeru-1.8.0_322\bin\java.exe ...
============================
1. 编写实现类进行测试
10 + 10 = 20.0
10 - 10 = 0.0
10 x 10 = 100.0
10 ÷ 10 = 1.0
============================
2.用局部内部类进行测试
10 + 10 = 20.0
10 - 10 = 0.0
10 x 10 = 100.0
10 ÷ 10 = 1.0
============================
3. 使用匿名内部类进行测试
10 + 10 = 20.0
10 - 10 = 0.0
10 x 10 = 100.0
10 ÷ 10 = 1.0
```

4.填代码题

成员内部类，静态内部类的区别实际上时非常明显的，对象的创建和使用都有很大差异。请完成下列题目：根据注释填写（1），（2），（3）处的代码：

```java
public class Test{
    public static void main(String[] args){
        //(1)创建并初始化Bean1类对象bean1
        Test test = new Test();
        Test.Bean1 bean1 =test.new Bean1();
        bean1.i++;
        //(2)创建并初始化Bean2类对象bean2
        Bean2 bean2 = new Bean2();
        bean2.j++;
        //(3)创建并初始化Bean3类对象bean3
        Bean.Bean3 bean3 = new Bean().new Bean3();
        bean3.k++;
    }
```

```
    class Bean1{
        public int i = 0;
    }
    static class Bean2{
        public int j = 0;
    }
}

class Bean{
    class Bean3{
        public int k = 0;
    }
}
```

```
package com.cskaoyan.homework.day11;

public class Test{
    public static void main(String[] args){
        System.out.println("=================");
        //(1)创建并初始化Bean1类对象bean1
        Test test = new Test();
        Test.Bean1 bean1 =test.new Bean1();
        System.out.println("bean1.i开始为:"+bean1.i);
        bean1.i++;
        System.out.println("bean1.i++后为:"+bean1.i);
        System.out.println("=================");
        //(2)创建并初始化Bean2类对象bean2
        Bean2 bean2 = new Bean2();
        System.out.println("bean2.j开始为:"+bean2.j);
        bean2.j++;
        System.out.println("bean2.j++后为:"+bean2.j);
        System.out.println("=================");
        //(3)创建并初始化Bean3类对象bean3
        Bean.Bean3 bean3 = new Bean().new Bean3();
        System.out.println("bean3.k开始为:"+bean3.k);
        bean3.k++;
        System.out.println("bean3.k++后为:"+bean3.k);
    }
    class Bean1{
        public int i = 0;
    }
    static class Bean2{
        public int j = 0;
    }
}

class Bean{
    class Bean3{
```

```
        public int k = 0;
    }
}
```

```
Run:  Test ×
C:\Users\h8916\.jdks\semeru-1.8.0_322\bin\j
=================
bean1.i开始为:0
bean1.i++后为:1
=================
bean2.j开始为:0
bean2.j++后为:1
=================
bean3.k开始为:0
bean3.k++后为:1
```