Java 程序设计 LAB04

JDK 版本	jdk-8u221-windows-64bit jdk1.8.0_221	代码文件 编码方式	UTF-8
IDE 种类 （如果有）	Eclipse	IDE 版本 （如果有）	jdk-8u131-windows-x64_8.0.1310.11.exe

1. package

你在 A.java 添加的语句

```
package com.oo.aa;
```

你在 B.java 添加的语句

```
package com.oo.bb;
import com.oo.aa.A;
```

你在 C.java 添加的语句

```
package com.oo.cc;
import com.oo.bb.B;
```

你在 Main.java 添加的语句

```
package com.oo;
import com.oo.cc.C;
```

你在 BTest.java 添加的语句

```
package test;
import com.oo.bb.B;
```

2. 权限

LAB04 中的单例模式，有可能有子类吗？

不可能

final 类可以被视为所有构造方法都是 private 的类吗？

不能，虽然 final 类和所有构造方法都是 private 的类都不能有子类继承，但是 final 类可以创建该类的对象，但是所有构造方法都是 private 的类不能创建该类的对象。

如下：

```
final class Cxy1{
    int num=1;
    public Cxy1(){
        System.out.println("我是 final 类 Cxy1");
        System.out.println("num"+num);
    }
}
class Cxy2{
    private int num1=1;
    private Cxy2(){
        System.out.println("我是默认类 Cxy2");
        System.out.println("num1"+num1);
    }
```

```
}
public class CxyTest {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Cxy1 cxy1=new Cxy1();
        //Cxy2 cxy2=new Cxy2();
    }
}
```

Cxy2 cxy2=new Cxy2();会报错，而 Cxy1 cxy1=new Cxy1();没有报错。

请不要忘记把不能编译的代码注释掉并提交

3. Hide&Override

执行 java Test，程序的运行结果是什么？

```
4
4
5
foo() of Parent
foo() of Child
foo() of Child
bar() of Parent
bar() of Parent
bar() of Child
```

如果将子类的 protected void foo()改为默认访问权限，程序还能通过编译吗？

不能

如果将子类的 protected void bar()改为默认访问权限，程序还能通过编译吗？

不能，foo 和 bar 的测试输出结果表现一致

属性、静态方法、非静态方法，哪些可以覆盖（重写、override），哪些可以隐藏？

属性、静态方法、非静态方法可以隐藏，非静态方法可以覆盖

4. Shape

在不添加新的属性的情况下，子类想要实现自己的 calcArea()该怎么办？

在父类中增加 getter 和 setter 方法，子类中可以通过调用该方法获取 a,b，并在子类中重写 calcArea()方法，该方法可以使用获取的 a,b。

```
class Rectangle extends Shape {
    public Rectangle(double a,double b){
        this.setA(a);
        this.setB(b);
    }
    public double calcArea() {
        return this.getA()*this.getB();
    }
}
```

7. 初始化 II

结合代码描述你的证明过程

证明（1）

```
class Fu {
    int num=1;
    static String p="parents";
    {
        System.out.println("我是父亲！ ");
        System.out.println("父亲的编号是"+num);
    }
    static {
        System.out.println("我是父亲！ ");
        System.out.println("我是"+p);
    }
}
class Zi extends Fu{
    int num1=2;
    int num2=3;
    static int num3=4;
    {
        System.out.println("我是孩子"+s+"！ ");
        System.out.println("孩子的编号是"+num2);
    }
    static String s="cxy";
    static {
        System.out.println("我是孩子"+s+"！ ");
        System.out.println("孩子的编号是"+num3);
    }
}
public class TestFuZi{
    public static void main(String[] args) {
        Zi zi=new Zi();
    }
}
```

输出：

```
我是父亲!
我是 parents
我是孩子 cxy!
孩子的编号是 4
我是父亲!
父亲的编号是 1
我是孩子 cxy!
孩子的编号是 3
```

在代码中可以看出，Zi zi=new Zi();时其父类还没有被装载过，所以会优先完成父类的装载，即初始化父类中的静态属性或方法，首先输出的：
我是父亲!
我是 parents
和第三四行输出的：
我是孩子 cxy!
孩子的编号是 4
可以看出，首先执行父类中 static String p="parents";的静态变量的初始化，紧接着执行父类中的静态代码块

```
static {
        System.out.println("我是父亲! ");
        System.out.println("我是"+p);
}
```

之后才回到了子类中，加载子类中的静态属性、执行静态代码块。
综上所述：JVM 执行类装载时，在加载某类的.class 文件后，如果其父类还没有被装载过，会先完成父类的装载（加载文件和静态初始化），之后才执行本类的静态初始化。

证明（2）

```
class Fu {
    int num=1;
    public Fu(){
        System.out.println("我是父亲! ");
        System.out.println("父亲的编号是"+num);
    }
}
class Zi extends Fu{
    public Zi() {
        System.out.println("我是孩子"+s+"! ");
        System.out.println("孩子的编号是"+num3);
    }
    int num1=2;
    int num2=3;
    int num3=4;
    String s="cxy";
}
public class TestFuZi{
    public static void main(String[] args) {
        Zi zi=new Zi();
    }
}
```

输出：
我是父亲!
父亲的编号是 1
我是孩子 cxy!

孩子的编号是 4

由输出可以看出，在 Zi zi=new Zi();后构造器构造对象，该对象有父类，所以先调用父类构造器，所以首先输出:
我是父亲!
父亲的编号是 1
等父类构造器执行结束后，再按照子类中定义的显式赋值的实例属性的顺序进行初始化，最后执行子类构造器的其他代码，因为在定义中显式赋值的实例属性在子类构造器之后定义，若按照定义的顺序应该先执行子类构造器中的代码，在执行实例属性的赋值，但是由输出：孩子的编号是 4，可以知道，int num3=4;在执行子类构造器中的代码之前执行。综上可以证明：调用构造器构造对象时的顺序：先父类，再属性，最后其他代码。