数学141 王涛 24

**5-3.**

父类SuperClass:

域：data

方法：setData(int newData)、getData()

子类SubClass：

域：data、subData

方法：setData(int newData)、getData()、setSubData(int newData)

**5-5.**

域的隐藏是指子类重新定义所继承父类中存在的同名域，导致所继承的同名域被隐藏。

覆盖后子类拥有域；SuperClass.data 、SubClass.data、subData

5-6.

1）方法的覆盖：子类中定义与父类同名的方法，实现方法的覆盖。

2）与域的隐藏不同：域隐藏后仍然占有内存，仍然可以被子类中继承自父类的方法访问，而方法覆盖后将不再占有内存，无法再调用。

3）与重载的不同：覆盖需要与父类方法中方法有完全相同的方法头声明。

重载的父类方法不会被覆盖仍然可以被调用。

**5-7.**

5-3中getData方法已经实现方法的覆盖，对继承的setData方法的覆盖可以在子类SubClass中定义如下方法：

setData(int newData)

{

subData=newData;

}

**5-8.**

this：表示当前对象本身，当前对象的一个引用。用来利用this可以调用当前对象的方法或者域，或将当前对象引用作为参数传递给其他对象或方法。

super：代表本类的直接父类。用于在类中引用它的父类。

**5-11.**

java实现多态的方式：方法覆盖实现多态、方法重载实现多态、对象引用的多态

**5-12.**

**public** **class** ComplexNumber {

**double** m\_dRealPart;

**double** m\_dImaginPart;

**public** ComplexNumber()

{

**this**.=0.0;

**this**.m\_dRealPart=0.0;

}

**public** ComplexNumber (**double** r,**double** i)

{

**this**.m\_dRealPart=r;

**this**.m\_dImaginPart=i;

}

**public** **double** getRealPart()

{

**return** **this**.m\_dRealPart;

}

**public** **double** getImaginaryPart()

{

**return** **this**.m\_dImaginPart;

}

**public** **void** setRealPart(**double** d)

{

**this**.m\_dRealPart=d;

}

**public** **void** setImaginaryPart(**double** d)

{

**this**.m\_dImaginPart=d;

}

**public** ComplexNumber complexAdd(ComplexNumber c)

{

**this**.m\_dImaginPart+=c.getRealPart();

**this**.m\_dImaginPart+=c.getImaginaryPart();

**return** **this**;

}

**public** ComplexNumber complexAdd(**double** c)

{

**this**.m\_dImaginPart+=c;

**return** **this**;

}

**public** ComplexNumber complexMinus(ComplexNumber c)

{

**this**.m\_dImaginPart-=c.getRealPart();

**this**.m\_dImaginPart-=c.getImaginaryPart();

**return** **this**;

}

**public** ComplexNumber complexMinus(**double** c)

{

**this**.m\_dImaginPart-=c;

**return** **this**;

}

**public** ComplexNumber complexMulti(ComplexNumber c)

{

**this**.m\_dRealPart=**this**.m\_dRealPart\*c.getRealPart()-**this**.m\_dImaginPart\*c.getImaginaryPart();

**this**.m\_dImaginPart=**this**.m\_dRealPart\*c.getImaginaryPart()+c.getRealPart()\***this**.m\_dImaginPart;

**return** **this**;

}

**public** ComplexNumber complexMulti(**double** c)

{

**this**.m\_dRealPart=**this**.m\_dRealPart\*c;

**this**.m\_dImaginPart=**this**.m\_dImaginPart\*c;

**return** **this**;

}

**public** ComplexNumber complexDiv(ComplexNumber c)

{

**this**.m\_dRealPart=**this**.m\_dRealPart\*c.getRealPart()+**this**.m\_dImaginPart\*c.getImaginaryPart();

**this**.m\_dImaginPart=c.getRealPart()\***this**.m\_dImaginPart-**this**.m\_dRealPart\*c.getImaginaryPart();

**return** **this**;

}

**public** ComplexNumber complexDiv(**double** c)

{

**this**.m\_dRealPart=**this**.m\_dRealPart/c;

**this**.m\_dImaginPart=**this**.m\_dRealPart/c;

**return** **this**;

}

**public** String toString()

{

**return** **this**.m\_dRealPart+"+"+**this**.m\_dImaginPart+"i";

}

}

使用了方法的重载，使类的设计和方法的调用更简便

**5-15.**

包是一种松散的类的集合，用来组织和管理类。

**5-17.**

1)引用包中的类：import <包名>.<包中类名>;

2)引用整个包：import <包名>.\*;

3) import java.applet .Applet.

**5-18.**

CLASSPATH 指明查找包的路径。系统将环境变量CLASSPATH包含的一个或多个目录作为起始目录，查找包目录，并在找到的包目录下查找类文件。

设置CLASSPATH环境变量：

set CLASSPATH=.;d:\temp

**5-19.**

接口用来组织应用中各类并调节它们的相互关系的一种结构，用来实现类间多重继承功能的结构。

java不支持多重继承，需要接口来实现这一功能。

接口是一种特殊的类，它只有抽象方法和常量。

**5-21.**

在类的定义时 implements <接口名>

如 public class TestClass implements ActionListener{};

如果实现接口的类不是抽象类，则必须重载（实现）接口的所有抽象方法。