继 承（下）

毕向东

4 继承（下）

4.7 抽象类

4.8 接口

4.7 抽象类

4.7.1 抽象类概述
4.7.2 抽象类的特点
4.7.3 抽象类举例代码讲解
4.7.4 抽象类相关问题

4.7.1 抽象类概述

- 抽象定义：
  - 抽象就是从多个事物中将共性的，本质的内容抽取出来。
  - 例如：狼和狗共性都是犬科，犬科就是抽象出来的概念。
- 抽象类：
  - Java中可以定义没有方法体的方法，该方法的具体实现由子类完成，该方法称为抽象方法，包含抽象方法的类就是抽象类。
- 抽象方法的由来：
  - 多个对象都具备相同的功能，但是功能具体内容有所不同，那么在抽取过程中，只抽取了功能定义，并未抽取功能主体，那么只有功能声明，没有功能主体的方法称为抽象方法。
  - 例如：狼和狗都有吼叫的方法，可是吼叫内容是不一样的。所以抽象出来的犬科虽然有吼叫功能，但是并不明确吼叫的细节。

4.7.2 抽象类的特点

- 抽象类和抽象方法必须用abstract关键字来修饰。
- 抽象方法只有方法声明，没有方法体，定义在抽象类中。
  - 格式：修饰符 abstract 返回值类型 函数名(参数列表);
- 抽象类不可以被实例化，也就是不可以用new创建对象。原因如下：
  - 抽象类是具体事物抽取出来的，本身是不具体的，没有对应的实例。例如：犬科是一个抽象的概念，真正存在的是狼和狗。
  - 而且抽象类即使创建了对象，调用抽象方法也没有意义。
- 抽象类通过其子类实例化，而子类需要覆盖掉抽象类中所有的抽象方法后才可以创建对象，否则该子类也是抽象类。

4.7.3 抽象类举例代码讲解

- 老师示例，根据给出内容设计继承体系
  - 具体事物：基础班老师，就业班老师
  - 共性：姓名，所属教室，讲课。
- 学员示例(练习)
  - 具体事物：基础班学员，就业班学员
  - 共性：姓名，学习，休假。
- 雇员示例：
  - 需求：公司中程序员有姓名，工号，薪水，工作内容。
  - 项目经理除了有姓名，工号，薪水，还有奖金，工作内容。
  - 对给出需求进行数据建模。

4.7.4 抽象类相关问题

- 抽象类中是否有构造函数？
- 抽象关键字abstract不可以和哪些关键字共存？
- 抽象类中可不可以没有抽象方法？

4.8 接口

- 格式：
  ```
  interface {}
  ```
- 接口中的成员修饰符是固定的。
  - 成员常量：public static final
  - 成员函数：public abstract
- 接口的出现将“多继承”通过另一种形式体现出来，即“多实现”。

4.8.1 接口的特点

- 接口是对外暴露的规则。
- 接口是程序的功能扩展。
- 接口可以用来多实现。
- 类与接口之间是实现关系，而且类可以继承一个类的同时实现多个接口。
- 接口与接口之间可以有继承关系。

4.9 多态

定义：某一类事物的多种存在形态。

- 例：动物中猫，狗。
- 猫这个对象对应的类型是猫类型
  - 猫 x = new 猫();
- 同时猫也是动物中的一种，也可以把猫称为动物。
  - 动物 y = new 猫();
  - 动物是猫和狗具体事物中抽取出来的父类型。
  - 父类型引用指向了子类对象。

4.9 多态

- 体现：

  父类或者接口的引用指向或者接收自己的子类对象。

- 作用：

  多态的存在提高了程序的扩展性和后期可维护性

- 前提：
  - 需要存在继承或者实现关系
  - 要有覆盖操作

多态的特点

- 成员函数：
  - 编译时：要查看引用变量所属的类中是否有所调用的成员。
  - 在运行时：要查看对象所属的类中是否有所调用的成员。
- 成员变量：
  - 只看引用变量所属的类。

内部类

- 将一个类定义在另一个类的里面，对里面那个类就称为内部类（内置类，嵌套类）。
- 访问特点：
  - 内部类可以直接访问外部类中的成员，包括私有成员。
  - 而外部类要访问内部类中的成员必须要建立内部类的对象。

内部类的位置

- 内部类定义在成员位置上
  - 可以被private static成员修饰符修饰。
  - 被static修饰的内部类只能访问外部类中的静态成员。
- 内部类定义在局部位置上
  - 也可以直接访问外部类中的成员。
  - 同时可以访问所在局部中的局部变量，但必须是被final修饰的。

匿名内部类

- 就是内部类的简化写法。
- 前提：
  - 内部类可以继承或实现一个外部类或者接口。
- 格式为：
  - new 外部类名或者接口名(){覆盖类或者接口中的代码，(也可以自定义内容。)}
- 简单理解：
  - 就是建立一个建立一个带内容的外部类或者接口的子类匿名对象。

异常

- 异常的体系
  - Throwable
    - Error
      - 通常出现重大问题如：运行的类不存在或者内存溢出等。
      - 不编写针对代码对其处理
    - Exception
      - 在运行时运行出现的一起情况，可以通过try catch finally
- Exception和Error的子类名都是以父类名作为后缀。

Throwable中的方法

- getMessage()
  - 获取异常信息，返回字符串。
- toString()
  - 获取异常类名和异常信息，返回字符串。
- printStackTrace()
  - 获取异常类名和异常信息，以及异常出现在程序中的位置。返回值void。
- printStackTrace(PrintStream s)
  - 通常用该方法将异常内容保存在日志文件中，以便查阅。

throws和throw

- throws用于标识函数暴露出的异常。
- throw用于抛出异常对象。
- throws与throw的区别：
  - thorws用在函数上，后面跟异常类名。
  - throw用在函数内，后面跟异常对象。

异常处理

```
try
{
    需要检测的代码；
}
catch(异常类  变量)
{
    异常处理代码；
}
finally
{
    一定会执行的代码；
}
```

Finally代码块只有一种情况不会被执行。就是在之前执行了System.exit(0)。

自定义异常

- 自定义类继承Exception或者其子类。
- 通过构造函数定义异常信息。

例：

```
Class DemoException extends Exception
{
    DemoException(String message)
    {
        super(message);
    }
}
```

- 通过throw将自定义异常抛出。

异常细节

- RuntimeException以及其子类如果在函数中被throw抛出，可以不用在函数上声明。
- 一个方法被覆盖时，覆盖它的方法必须抛出相同的异常或异常的子类。
- 如果父类抛出多个异常，那么重写（覆盖）方法必须抛出那些异常的一个子集，不能抛出新的异常。
- 介绍异常在分层设计时的层内封装。
- 例程。

包(package)

- 对类文件进行分类管理。
- 给类提供多层命名空间。
- 写在程序文件的第一行。
- 类名的全称的是 包名.类名。
- 包也是一种封装形式。

classpath

- 给JVM提供的一个环境变量。
- 指定类或者包所在的路径。
- classpath变量值的最后有分号与无分号的区别。
- 思考：在开发时,分号是否需要呢？

包之间的访问

- 被访问的包中的类权限必须是public的。
- 类中的成员权限：public或者protected
- protected是为其他包中的子类提供的一种权限
- 例程

四种权限

	public	protected	default	private
同一类中	√	√	√	√
同一包中	√	√	√	
子类	√	√		
不同包中	√			

import

- 简化类名。
- 一个程序文件中只有一个package，可以有多个import。
- 用来导包中的类，不导入包中的包。
- 通常写import mypack.Demo;
  而不写import mypack.*;为什么？

Jar包

- Java的压缩包
  - 方便项目的携带。
  - 方便于使用，只要在classpath设置jar路径即可。
  - 数据库驱动，SSH框架等都是以jar包体现的。

Jar包的操作

通过jar.exe工具对jar的操作。

- 创建jar包
  - jar -cvf mypack.jar packa packb
- 查看jar包
  - jar -tvf mypack.jar [>定向文件]
- 解压缩
  - jar -xvf mypack.jar
- 自定义jar包的清单文件
  - jar –cvfm mypack.jar mf.txt packa packb