# 面向对象基础

## 一、类和对象

（1）类：类是模子，确定对象将会拥有的特征(属性)和行为(方法)。

类的特点：类是具有相同属性和方法的一组对象的集合。

### （2）对象

创建对象：类名 对象名 = new 类名()

使用对象：对象.属性   /   对象.方法名()

### （3）成员变量与局部变量

成员变量：在类中定义，用来描述对象将要有什么。（作用域在整个类内部都是可见的）

局部变量：在类的方法中定义，在方法中临时保存数据。（作用域仅限于定义它的方法）

区别：

①作用域不用

②两类变量同名时，局部变量的优先级更高

③初始值不同：Java会给成员变量一个初始值，不会给局部变量赋予初始值

（4）构造方法：定义在Java中的一个用来初始化对象的方法。

①使用new+构造方法 创建一个新的对象。

②名称与类名相同且没有返回值。

③当没有指定构造方法时，系统会自动添加无参的构造方法。

④当有指定构造方法时，无论是有参、无参的构造方法，都不会自动添加无参的构造方法。

⑤构造方法可重载：方法名相同，但参数不同，调用时会自动根据不同的参数选择相应的方法。

⑥构造方法不但可以给对象的属性赋值，还可以保证给对象的属性赋一个合理的值（在构造函数中增加判断）。

### （5）static静态变量（也称类成员）

①它属于整个类所有，而不是某个对象所有，被类的所有对象所共享。

②静态成员可使用类名直接访问，也可以使用对象名进行访问。

③静态成员属于整个类，当系统第一次使用该类时，就会为其分配内存空间直到该类被卸载才会进行资源回收。

### （6）static静态方法（也称类方法）

①静态方法中可以直接调用同类中的静态成员，但不能直接调用非静态成员。

②如果系统在静态方法中调用非静态变量，可通过创建类的对象，然后通过对象来访问非静态变量。

③在普通成员方法中，则可以直接访问同类的非静态变量和静态变量。

④静态方法中不能直接调用非静态方法，需要通过对象来访问非静态方法。

### （7）初始化块

①普通初始化块

{

xxx = xxx;

}

②静态初始化块

static {

xxx = xxx;

}

③执行顺序：静态初始化块最先被执行-->普通初始化块-->构造方法

## 二、封装

（1）概念：将类的某些信息隐藏在类内部，不允许外部程序直接访问，而是通过该类提供的方法类实现对隐藏信息的操作和访问。

（2）好处：只能同规定的方法访问数据，隐藏类的实现细节，方便修改和实现。

### （3）实现步骤

① 修改属性的可见性：设为private

② 创建getter/setter方法：用于属性的读写

③ 在getter/setter方法中加入属性控制语句：对属性值的合法性进行判断

### （4）Java中的包

①作用：管理Java文件，解决同名文件的冲突

②必须放在Java源程序的非注释性语句第一行，包名间用"."号隔开

③包的使用：可以通过import关键字在某个文件中使用其它文件中的类，Java包的命名是全小写字母。

（5）Java中的访问修饰符：可以修饰属性和方法的访问范围

①private：只能在本类中访问和使用。

②默认：能在本类和同包中访问和使用。

③protected：能在本类、同包、子类中使用。

④public：在本类、同包、子类、其它类中使用。

### （6）Java中this关键字

代表当前对象的一个引用

this.属性：操作当前对象的属性

this.方法：调用当前对象的方法

在构造方法中第一行用this(实参列表)调用其他构造方法

## 三、继承

（1）继承时类与类的一种关系，是一种"is a"的关系

（2）Java中的继承时单继承，只有一个父类

1. class 子类 extends 父类
2. 子类的构造方法

① 如果没有用super显式调用父类的有参构造方法，也没有通过this显式调用自身的其他构造方法，则系统会默认先调用父类的无参构造方法。在这种情况下，是否写”super()；”语句，效果是一样的

② 如果通过super显式调用了父类的有参构造方法，将执行父类相应的构造方法

③ 如果通过this显式调用了自身的其他构造方法，在相应的构造方法中遵循以上两条规则

（5）方法的重写

① 子类可重写从父类继承的方法，当调用方法时会优先调用子类的方法。

② 返回值类型、方法名、参数类型及个数 都要与父类继承的方法相同，才叫方法的重写。

（6）继承的初始化顺序

① 初始化父类再初始化子类

② 属性的初始化在构造方法之前执行。

③ 父类的属性初始化-->父类的构造函数-->子类的属性初始化-->子类的构造函数

（7）final

① 可修饰类、方法、属性和变量

② final修饰类，则该类不允许被继承

③ final修饰方法，则该方法不允许被覆盖(重写)

④ final修饰属性，则该类的属性不会进行隐式初始化（类的初始化属性必须有值）或在构造方法中赋值（只能任选其一）

⑤ final修饰变量，则该变量的值只能赋一次值，即变为常量

（8）super关键字

代表父类对象的一个引用

1. super.属性：可调用父类所有非私有属性
2. super.方法: 可调用父类所有非私有非抽象方法
3. 在构造方法中第一行用super(实参列表)调用父类的构造方法

（9）Object类是所有类的父类，如果一个类没有使用extends关键字明确标识继承另一个类，那么这个类默认继承Object类。

## 四、多态

继承是多态实现的基础。

（1）引用多态：父类的引用可以指向本类的对象/父类的引用可以指向子类的对象。

（2）方法多态：创建本类对象时，调用的方法为本类方法/创建子类对象时，调用的方法为子类重写的方法或继承的方法。

（3）引用类型转换

① 向上类型转换：隐式/自动类型转换，小类型到大类型的转换。（无风险）

② 向下类型转换：强制类型转换，是大类型到小类型。（有风险）

## 异常

① 异常：

是什么？是对问题的描述，将问题进行对象的封装

② 异常体系：

Throwable

|--Error

|--Exception

|--RuntimeException

③ 异常体系的特点：异常体系中的所有类以及建立的对象都具备可抛性

也就是说可以被throw和throws关键字所操作

只有异常体系具备这个特点

④ throw和throws的用法：

throw定义在函数内，用于抛出异常对象

throws定义在函数上，用于抛出异常类，可以抛出多个用逗号隔开

但函数内部有throw抛出异常对象，并未进行try处理，必须要在函数上声明，都在编译失败

注意：RuntimeException除外，也就说，函数内如果抛出的是RuntimeException异常，函数上可以不用声明

如果函数声明了异常，调用者要进行处理，处理方法可以throws可以try

⑤ 异常有两种：

1、编译时被检测异常

该异常在编译时，如果没有处理(没有抛也没有try)，编译失败

2、运行时异常(编译时不检测)

在编译时，不需要处理，编译器不检测

该异常的发生，建议不处理，让程序停止，需要对代码进行修正

异常处理语句：

try

{

需要被检测的代码;

}

catch()

{

处理异常的代码;

}

finally

{

一定会执行的代码;

}

有三种结合格式：

1, try

{

}

catch()

{

}

2, try

{

}

finally

{

}

3, try

{

}

catch()

{

}

finally

{

}

注意：

1，finally中定义的通常是 关闭资源代码，因为资源必须释放

2，finally只有一种情况不会执行，当执行到System.exit(0);fianlly不会执行

⑥自定义异常：

定义类继承Exception或者RuntimeException

1，为了让该自定义类具备可抛性

2，让该类具备操作异常的共性方法

当要定义自定义异常的信息时，可以使用父类已经定义好的功能

异常异常信息传递给父类的构造函数

class MyException extends Exception

{

MyException(String message)

{

super(message);

}

}

自定义异常：按照java的面向对象思想，将程序中出现的特有问题进行封装

--------------------

异常的好处：

1，将问题进行封装

2，将正常流程代码和问题处理代码相分离，方便于阅读

异常的处理原则：

1，处理方式有两种：try或者throws

2，调用到抛出异常的功能时，抛出几个，就处理几个

一个try对应多个catch

3，多个catch，父类的catch放到最下面

4，catch内，需要定义针对性的处理方式，不要简单的定义PrintStackTrace，输出语句也不要不写

当捕获到的异常，本功能处理不了时，可以继续在catch中抛出

try

{

throw new AException();

}

catch(AException e)

{

throw e;

}

如果该异常处理不了，但并不属于该功能出现的异常

可以将异常转换后，再抛出和该功能相关的异常

或者异常可以处理，当需要将异常产生的和本功能相关的问题提供出去

当调用者知道，并处理，也可以将捕获异常处理后，转换新的异常

try

{

throw new AException();

}

catch(AException e)

{

//对AException处理

throw new Bexception();

}

比如，汇款的例子

异常的注意事项：

在子父类覆盖时：

1，子类抛出的异常必须是父类异常的子类或者子集

2，如果父类或者接口没有异常抛出时，子类覆盖出现异常，只能try不能抛