面向对象学习笔记二

一、面向对象中的类及其成员的修饰符和作用

权限修饰符：private 、默认(什么都不写)、protected 、public(权限从小到大)

状态修饰符：static final

抽象修饰符：abstract

类：

权限修饰符：默认和 public 两种，常用的是 public，作用对外提供访问权限。

状态修饰符：final，表示该类是最终类不能被其他类继承，但可以创建对象。

注意：在 java 的 JDK 中被 final 修饰的类不能被继承!!!

抽象修饰符：abstract，表示该类是抽象类，不能直接创建对象，只能被继承，通过子类或多态的方式来创建对象。抽象类可以不包含抽象方法，但包含抽象方法的类一定是抽象类。

成员变量：

权限修饰符：private 、默认 、protected 、public，最常用的是 private，用于封装成员变量，保护数据的安全，且该变量只在本类中才能访问。

状态修饰符：static、final。被 static 修饰的成员变量也叫类变量可以由类名.变量名的方式直接访问，此时该变量被这个类的全体对象所共享；被 final 修饰的变量，其值不可以被改变。二者可以同时使用。

构造方法：

权限修饰符：private、默认、protected 、public，最常用的是 public，当构造方法被 private 修饰时表示该构造方法不能用来 new 出对象。

成员方法：

权限修饰符：private、默认、protected、public，最常用的是 public，对外提供访问权限。被 private 修饰的成员方法只在本类中才能访问。

状态修饰符：static、final，被 static 修饰的成员变量也叫类变量可以由类名.方法名的方式直接访问；被 final 修饰的方法不能被子类重写，二者可以同时使用。

抽象修饰符：abstract，被 abstract 修饰的成员方法称为抽象方法，该方法只有声明没有方法体，只能被子类重写。

注意：抽象修饰符 abstract 不能和 private、final、static 一起使用。

二、成员变量和局部变量的区别

在类中的位置不同

成员变量：在类中方法外

局部变量：在方法定义中或者方法声明上

在内存中的位置不同

成员变量：在堆内存分配存储空间

局部变量：在栈内存分配存储空间

生命周期不同

成员变量：随着对象的创建而存在，随着对象的消失而消失

局部变量：随着方法的调用而存在，随着方法的调用完毕而消失

初始化值不同

成员变量：有默认初始化值。

局部变量：没有默认初始化值，必须定义，赋值，然后才能使用。

注意：定义变量的时候，尽可能将这个变量的使用范围最小化，能定义成局部变量，就不定义成成员变量

三、静态变量和成员变量的区别

所属不同：

静态变量属于类，所以也称为为类变量

成员变量属于对象，所以也称为实例变量(对象变量)

内存中位置不同：

静态变量存储于内存中的 data 数据区。

成员变量存储于堆内存

内存出现时间不同：

静态变量随着类的加载而加载，随着类的消失而消失

成员变量随着对象的创建而存在，随着对象的消失而消失

调用不同：

静态变量可以通过类名调用，也可以通过对象调用

成员变量只能通过对象名调用

四、静态的作用，特点？为什么静态不能访问非静态?

static 关键字可以修饰成员变量和成员方法，方便我们使用，无需创建对象，直接用类名.静态方法名/变量名即可调用，节约堆区的开销。

static 关键字特点：

静态所修饰的成员，随着类的加载而加载，优先于对象存在，会被该类的所有对象共享。

static 关键字注意事项：

在静态方法中是没有 this 关键字，静态方法只能访问静态的成员变量和静态的成员方法，简而言之静态只能访问静态。

五、静态代码块和构造代码块的区别，静态代码块、构造代码块、构造方法的执行顺序

静态代码块：只执行一次，以后创建再创建多少个对象，都不会执行；

构造代码块：每创建一个对象，就会执行一次；

执行顺序：先执行所有的静态代码块，再执行所有的构造代码块，最后执行构造方法。

六、this 和 super 的区别

this 代表本类对应的引用，用来解决成员变量与局部变量重名问题。

this 关键字代表所在类的对象引用，方法被哪个对象调用，this 就代表那个对象。

Super：代表父类引用。

Java 里的子类用 super 调用父类构造函数时，调用的函数必须放在子类的第一条语句的位置。可以在子类中用 super 关键字调用父类中非私有的成员变量和成员方法。

七、重写和重载的区别？

重写 Override：指子类中出现了和父类中一模一样的方法声明，也被称为方法覆盖，方法复写，与返回值类型无关。

注意：父类中私有方法不能被重写，子类重写父类方法时，访问权限不能更低。如果父类的方法是 private 类型，那么，子类则不存在覆盖的限制，相当于子类中增加了一个全新的方法。

重载 Overload：表示同一个类中可以有多个名称相同的方法，但这些方法的参数列表各不相同(即参数个数或类型不同)。

八、抽象类和接口的区别

抽象类：abstract 修饰的类即为抽象类，抽象类不能创建实例对象。含有 abstract 方法的类必须定义为抽象类，抽象类中可以没有抽象方法。抽象类中定义抽象方法必须在具体子类中实现，所以，不能有抽象构造方法或抽象静态方法。如果子类没有实现抽象父类中的所有抽象方法，那么子类也必须定义为 abstract 类型。

接口：可以说成是抽象类的一种特例，接口中的所有方法都必须是抽象的。接口中的方法定义默认为 public abstract 类型，接口中的成员变量类型默认为 public static final。

两者的语法区别

1. 抽象类可以有构造方法，接口中不能有构造方法。

2. 抽象类中可以有普通成员变量，接口中没有普通成员变量

3. 抽象类中可以包含非抽象的普通方法，接口中的所有方法必须都是抽象的，不能有非抽象的普通方法。

4. 抽象类中的抽象方法的访问类型可以使 public、protected 和默认类型，但接口中的抽象方法只能是 public 类型的，并且默认修饰即为 public abstract 类型。

5. 抽象类中可以包含静态方法，接口中不能包含静态方法

6. 抽象类和接口中都可以包含静态成员变量，抽象类中的静态成员变量的访问类型可以任意，但接口中定义的变量只能是 public static final 类型，并且默认即为 public static final 类型。

7. 一个类可以实现多个接口，但只能继承一个抽象类。

九、内部类和匿名内部类

内部类：

把一个类定义在其他类的内部，这个类就被称为内部类。例如：

```
class Outer{
    class Inner{}//内部类
}
```

特点：

内部类可以直接访问外部类的成员，包括私有。

外部类要访问内部类的成员，必须创建对象。

内部类的分类：

成员内部类：类中方法外

局部内部类：方法内

静态内部类：

成员内部类的使用格式：

外部类名.内部类名 对象名 = new 外部类名().new 内部类名();

成员内部类的修饰符：

private: 提高安全性

static : 方便内部类的访问

局部内部类可以访问方法中的局部变量，但该变量必须被 final 修饰，因为局部变量会随着方法的调用完毕而消失，这个时候，局部对象并没有立马从堆内存中消失，还要使用那个变量。为了让数据还能继续被使用，就用 fianl 修饰该变量，这样在堆内存里面存储的其实是一个常量值。

匿名内部类：

就是没有名字的内部类，它是内部类的简化写法，匿名内部类本质就是一个继承了类或者实现了接口的子类匿名对象。

匿名内部类理解为继承了一个类或者是实现了一个接口的子类，并且完成了子类对象的创建，同时子类对象没有名字，是对象的一种简化表示形式。

前提：存在一个类或者接口，这里的类可以是具体类也可以是抽象类。

格式：

```
new 类名或者接口名() {
    重写方法;
};
```

匿名对象的两种使用情况:

对象调用方法仅仅一次的时候

```
new Student().study();
```

好处：当方法调用完毕后，这个对象就成为了垃圾，会被 JVM 的垃圾回收器清理

作为实际参数传递

```
t.method( new Student() );
```

十、什么是包 package，什么是 import；package，import 和 class 之间有何关系？

包就是一个文件夹，用来存储多个.class 类的，对类进行分类管理。

包的格式：package 包名，多级包之间，使用.分隔。

注意：package 语句必须是程序的第一条可执行的代码。

package 语句在一个 java 文件中只能有一个，如果没有 package，默认表示无包名。

可以有多个 import，可以有多个 class，通常一个 java 文件中对应一个类。

Import 表示同包下类之间的访问

导包格式：import 包名.类名(类名可以用通配符*代替，不建议这样使用)

三者之间的的关系是：package --> import --> class

十一、类与类、类和接口、接口和接口的关系

类与类：java 中的类是单继承关系，可以多层继承。

类与接口：实现关系，可以实现一个接口，也可以同时实现多个接口。

接口与接口：继承，单继承，多继承

十二、java 中的多态

概述：某一个事物，在不同时刻表现出来的不同状态

多态前提和体现

1： 要有继承关系

2: 要进行方法重写(可以不从写，但那样就失去继承的意义)

3： 父类的引用 指向 子类对象

```
Fu   fu = new Zi();
```

成员访问特点

成员变量：编译看左边，运行看左边。

构造方法：给对象中的成员进行初始化操作使用。

成员方法：(方法重写,方法覆盖)编译看左边，运行看右边。

静态方法：静态方法没有方法重写，静态方法是类方法，编译看左边，运行看左边。

注意：成员方法的运行看右边， 其他情况都是看左边

多态的好处：提高了程序的维护性(由继承保证)，提高了程序的扩展性(由多态保证)。

多态的弊端：不能访问子类特有功能

多态中的转型问题

向上转型：

```
Animal an = new Cat();
```

父类引用 指向 子类对象

左边 (父类型) = 右边 (子类型)

向下转型：

```
Cat cat = (Cat)an;
```

父类引用转为子类对象,需要通过强制转换来完成

左边(子类类型) = 右边(父类类型)

方法的重写区别：

子类没有重写父类方法：编译看左边的父类方法，运行是父类方法(方法从父类继承过来的)。

子类重写了父类方法：编译看左边的父类方法，运行看右边的子类方法。

本文总结与 java 基础班课堂笔记和毕老师的视频，如有描述不清或错误的地方，还请大家指出，以便大家共同学习，共同进步，谢谢！