面向对象上

一、类和对象

1.定义类

定义类的语法格式：

```
[修饰符] class 类名{
零到多个构造器
零到多个成员变量
零到多个方法
}
```

- 修饰符：public、final、abstract，或者省略
- 类名由一到多个有意义单词连缀而成，每个单词首字母大写，其余字母小写，单词间无分隔符
- 类里各成员之间的定义顺序没有区别，可以相互调用，但static修饰的成员不能访问无static修饰的成员

（1）定义构造器

构造器是一种特殊的方法，如果类中没有定义构造器，系统自动提供默认构造器（默认构造器无参数），如果显式定义了构造器，则系统不提供默认构造器

定义构造器格式：

```
[修饰符] 构造器名(形参列表){
//构造器执行体
}
```

- 修饰符：public、private、protected，或者省略
- 构造器名与类名相同
- 形参列表与方法中形参列表格式相同
- 不能定义返回值类型，也不能用void声明无返回值，否则系统会当作方法来处理

（2）定义成员变量

定义成员变量格式：

```
[修饰符] 类型 成员变量名 [=默认值]
```

- 修饰符：可以省略，也可以是public、private、protected三者之一，final、static，它们能够相互组合
- 类型：Java允许的任何数据类型、
- 成员变量名：由一到多个有意义单词连缀而成，第一个单词首字母小写，其余首字母大写，其余字母小写，单词间无分隔符
- 默认值：可以指定一个默认值

（3）**定义方法**
定义方法格式：

```
[修饰符] 方法返回值类型 方法名（形参列表）{
//方法体
}
```

- 修饰符：可以省略，也可以是public、private、protected三者之一，final、abstract两者之一，static，它们能够相互组合
- 方法返回值类型：Java允许的任何数据类型
- 方法名：由一到多个有意义单词连缀而成，第一个单词首字母小写，其余首字母大写，其余字母小写，单词间无分隔符
- 形参列表：由一到多组"参数类型 形参名"组成

2.定义对象

（1）**对象的产生和使用**
创建对象的根本途径是构造器，通过new关键字来调用类的构造器来创建类的实例，对象可以调用类的成员变量和方法
类或实例访问方法或成员变量的语法是：类.类变量|类方法，或者实例.实例变量|实例方法
static修饰的方法和成员变量可以通过类或实例来调用，无static修饰的方法和成员变量只能通过实例来调用

（2）**对象的this引用**
this关键字总是指向调用该方法的对象，this作为对象的默认引用有两种使用情形：

- 构造器中引用该构造器正在初始化的对象
- 方法中引用调用该方法的对象

this关键字的最大作用是让类中一个方法访问类中另一个方法或成员变量Java允许对象的一个成员直接访问另一个成员，无需使用this引用，实际上this依然存在
static修饰的方法中不能使用this引用，如果需要在静态方法中访问另一个普通方法，则只能重新创建一个对象

二、构造器

1.使用构造器执行初始化

构造器最大的用处是在创建对象时执行初始化
如果用户希望保留无参数构造器，或者希望有多个初始化过程，则可以提供多个构造器，这就构成了构造器重载

2.构造器重载

同一个类里具有多个构造器，多个构造器的形参列表不同，即构成了构造器重载
如果系统中包含了多个构造器，其中一个构造器的执行体完全包含了另一个构造器的执行体，则可以通过this调用另一个重载的构造器的初始化代码，this调用只能作为构造器执行体第一条语句

三、成员变量和局部变量

1.成员变量和局部变量是什么

- 成员变量包括：类成员变量（static修饰，属于类）、实例成员变量（无static修饰，属于实例）
- 局部变量包括：形参、方法局部变量、代码块局部变量

成员变量与局部变量的区别:

- 成员变量无需显式初始化，定义成员变量时，系统会默认执行初始化，而局部变量（除形参）必须显式初始化后才能使用
- 成员变量作用域为类或实例，而局部变量作用域为方法或代码块
- 同一个类里不能同时定义两个同名的成员变量，同一个方法里不能同时定义两个同名的局部变量，形参与方法局部变量同名也不行
- 成员变量可以与局部变量同名，此时局部变量会覆盖成员变量，如果要在方法中访问成员变量，可以使用this或类名来调用

2.成员变量使用规则

以下几种情况，考虑使用成员变量:

- 需要定义的变量是用于描述某个类或某个对象的固有信息
- 需要定义的变量是用于保存该类或者实例的状态信息
- 需要定义的变量是用于在类中多个方法间共享

四、方法

1.方法的所属性

方法要么属于类本身，要么属于对象

2.方法参数的传递

Java中参数传递方式只有一种：值传递。引用类型的参数传递也是值传递
值传递：ValueTrans.java

3.形参个数可变的方法

Java允许定义形参个数可变的参数。如果在定义方法时，在最后一个形参的类型后增加三个点（...），则表明该形参可接受多个参数值，多个参数值被当作数组传入。
个数可变的形参只能处于形参列表的最后。其本质就是一个数组类型的形参，因此其既可以接受多个参数，也可以接受一个数组

4.递归方法

一个方法体内调用它自身，被称为方法递归

5.方法重载

同一个类中可以定义多个同名方法，只要方法名相同且形参列表不同，就会构成方法重载。

五、隐藏和封装

1.理解封装

封装是面向对象的三大特征之一（另外两个是继承和多态），它指将对象的状态信息隐藏在对象内部，不允许外部程序直接访问对象内部信息，而是通过该类所提供的方法来实现对信息的操作和访问。

2.使用访问控制符

Java提供了3个访问控制符：public、private、protected，分别代表3个访问控制级别，另外还提供了一个不加任何访问控制符的访问控制级别：default，一共4个访问控制级别。
访问控制级别由小到大：
private ----> default ----> protected ----> public

- private：当前类访问权限
- default：当前包访问权限
- protected：子类访问权限，既可以被当前包中其他类访问，也可以被其他包中的子类访问
- public：公共访问权限

局部变量不能使用访问控制符修饰
外部类只能使用public和default修饰
类中大部分成员变量都应该使用private修饰，只有一些类似全局变量的成员变量才考虑使用public修饰

3.导包

(1)package

- Java允许将一组功能相关的类放在同一个包下，组成逻辑上的类库单元。将一个类放在某个包下的格式（源程序的第一个非注释行）： `package packageName` ，位于包中的每个类的完整类名为包名和类名的组合： `packageName.className` 。
- 包名由多个有意义的单词连缀而成，且全部是小写字母
- 同一个包下的类可以自由访问，无需添加包名前缀

(2)import
为简化编程，Java引入了import关键字导入类
import关键字用于向Java文件中导入指定包层次下某个类或全部类，import语句位于package语句后，类定义之前。一个类只能有一条package语句，可以有多条import语句

- 导入单个类： `import package.subpackage... .ClassName`
- 导入全部类： `import package.subpackage... .*`

一旦用import导入了指定类，在程序中使用该类就可以省略包名前缀

(3)import static
静态导入import static用于导入类的静态成员变量和静态方法
import static用于向Java文件中导入指定包层次下某个类的某个静态变量和静态方法或全部静态变量和静态方法，import static语句位置和import语句一样，与import没有先后顺序

- 导入某个静态成员或静态方法： `import static package.subpackage... .className.fieldName|methodName`
- 导入全部静态成员和静态方法： `import static package.subpackage... .className.*`

4.Java常用包

Java的核心类都放在java包及其子包下

- java.lang:核心类，系统默认导入该包下所有类
- java.util:工具类/接口、集合框架类/接口
- java.net:网络相关
- java.io:输入输出相关
- java.text:格式化相关
- java.sql:数据库相关
- java.awt:抽象窗口工具集相关
- java.swing:用户图形界面相关

六、继承

1.理解继承

继承是面向对象的三大特征之一，也是实现软件复用的重要手段。Java的继承是单继承，每个子类最多只有一个父类
子类继承父类格式：

```
[修饰符] class Subclass extends SuperClass{
//类体
}
```

- 每个子类最多只有一个直接父类，可以有无限多个间接父类
- java.lang.Object类是所有类的父类，要么是直接父类，要么是间接父类
- 子类可以得到父类的所有成员变量和方法

2.重写父类方法

子类可以重写父类方法，要求是子类方法与父类方法（1）方法名相同（2）形参列表相同（3）子类方法返回值类型比父类小或相等（4）子类方法抛出的异常类型比父类小或相等（5）子类方法访问权限比父类大或相等

- 子类方法与被重写方法要么都是类方法，要么都是实例方法
- 子类方法重写父类方法后，子类对象将无法直接访问被重写方法，可以通过使用super调用（实例方法）或父类类名（类方法）调用该方法
- 如果父类方法使用private修饰，那么子类不能重写该方法，只能重新定义一个同名的新方法

3.super调用

super关键字用于限定对象调用它从父类继承得到的实例变量或实例方法
类方法中不能使用super调用

super调用的两种使用情形：

- 构造器中，super限定该构造器初始化的是从父类中继承来的实例变量
- 方法中，super限定该方法使用的是从父类中继承来的实例变量

4.调用父类构造器

子类不会获得父类的构造器，但是子类构造器中可以通过super关键字调用父类构造器中的初始化代码（super调用父类构造器必须是子类构造器的第一条语句，因此this调用和super调用不会同时出现）

- 当调用子类构造器时，父类构造器总是在子类构造器之前执行，因此创建任何对象，最先执行的总是java.lang.Object类的构造器

七、多态

1.理解多态

多态是面向对象的三大特征之一。Java引用变量有两个类型：一个是编译时类型，一个是运行时类型。编译时类型由声明该变量时使用的类型确定，运行时类型由实际赋给该变量的对象确定，如果两个类型不一致，就会出现多态。

当把子类对象直接赋给父类引用变量时：

- 方法具有多态性：调用该对象的方法时，总是表现出子类方法的行为特征
- 成员变量不具有多态性：调用该对象的成员变量时，总是表现出父类成员变量的特征
- 引用变量在编译阶段只能调用其编译时类型所具有的方法，但运行时则执行其运行时类型所具有的方法。因此引用变量只能调用其声明时类型的方法

2.引用变量的强制类型转换

引用变量只能调用其编译时类型声明的方法，为了调用其运行时类型声明的方法，可以强制类型转换

强制类型转换有两类：

- 基本类型间的强制类型转换只能在数值类型之间转换
- 引用类型间的强制类型转换只能在具有继承关系的两个类型之间转换。两个类型无继承关系强制转换编译时会报错，且父类对象实际是子类实例时强制转换才能成功，否则运行时会报错

3.instanceof 运算符

instanceof 运算符的前一个操作数通常是一个引用变量，后一个操作数通常是一个类，用于判断前面的对象是否是后面的类、其子类、其实现类的实例。

- instanceof 运算符前一个操作数要么与后面的类同类型，要么与后面的类具有父子继承关系，否则编译时会报错

八、继承与组合

1.继承的问题

继承是实现类复用的重要手段，但是继承会破坏封装。
从父类派生新的子类，不仅需要保证子类是一种特殊的父类，还需要保证以下两个条件之一：

- 子类需要增加额外属性
- 子类需要增加自己独有的行为方式（包括增加新的方法或重写父类方法）

2.使用组合实现复用

- 继承实现复用：子类可以直接访问从父类继承而来的成员变量和方法
- 组合实现复用：将旧类对象作为新类的成员变量组合起来，实现复用

继承与组合的使用：ExtendsTest.java、ConbinationTest.java

初始化块

初始化块用于对类或对象执行初始化操作
初始化块格式：

```
[修饰符] {
//初始化体
}
```

- 初始化块的修饰符：static或省略，静态初始化块初始化静态成员变量，实例初始化块初始化实例成员变量
- 初始化块没有标识，不能通过类或对象来调用，其只在创建对象时隐式执行。其作用是将多个构造器中重复的代码提取出来，实现软件复用，实际上在编译过程中，初始化块代码会被还原到构造器中
- 第一次创建某个类的实例时，执行顺序为Object类的静态初始化块、... 、直接父类的静态初始化块、该类的静态初始化块、Object类的普通初始化块、Object类的构造器、... 、直接父类的普通初

始化块、直接父类的构造器、该类的普通初始化块、该类的构造器。后面再创建该类的实例时，就不需要再执行静态初始化块了