一、IDEA使用

1、简介

```
1.1、IDEA全称IntelliJ IDEA
1.2、在业界被公认为最好的Java开发工具
1.3、IDEA是JetBrains公司的产品
1.4、除了支持Java开发，还支持HTML、CSS、PHP、MySQL、Python等
```

2、快捷键

```
2.1、删除当前行：ctrl + d
2.2、复制当前行：ctrl + alt + v
2.3、补全代码：alt + /
2.4、添加/取消注释：ctrl + /
2.5、导入类：先配置auto import，然后alt + enter
2.6、格式化代码：ctrl + alt + L
2.7、运行：alt + R
2.8、生成构造器：alt + insert
2.9、查看类层级关系：ctrl + H
2.10、定位方法：ctrl + B
2.11、自动分配变量名：在后面加.var
```

3、其他

```
3.1、自定义模板：file -> settings -> editor -> Live templates
```

二、包

1、简介

```
包，实际上就是创建不同的文件夹来保存类文件。
```

2、特点

```
2.1、区分相同名字的类
2.2、当类很多时，可以很好的管理类
2.3、控制访问范围
```

3、语法

```
package com.test;
说明：
    - package：关键字，表示打包
    - com.test：表示包名
```

4、命名

```
4.1、命名规则：只能包含数字、字母、下划线、小圆点.，但不能以数字开头，不能是关键字或保留字
4.2、命名规范：一般是小写字母+小圆点
    - com.公司名.项目名.业务模块名
    - com.hspedu.oa.model;
    - com.hspedu.oa.controller;
```

5、常用包

```
java.lang.*     // lang包是基本包，默认引入，不需要手动引入
java.util.*     // util包，系统提供的工具包、工具类
java.net.*      // 网络包，用于网络开发
java.awt.*      // 用于java的界面开发，GUI
```

6、引入包

```
import 包;
```

7、细节

```
7.1、package的作用是声明当前类所在的包，需要放在类的最上面，一个类中最多只有一句package
7.2、import指令，位置放在package的下面，在类定义前面，可以有多句且没有顺序要求
```

三、访问修饰符

1、简介

```
Java提供4种访问控制修饰符，用于控制方法和属性(成员变量)的访问权限(范围)
```

2、分类

访问级别	访问控制修饰符	同类	同包	子类	不同包
公开	public	√	√	√	√
受保护	protected	√	√	√	×
默认	没有修饰符	√	√	×	×
私有	private	√	×	×	×

3、细节

```
3.1、修饰符可以用来修饰类中的属性、成员方法以及类
3.2、只有默认的和public才能修饰类
3.3、成员方法的访问规则和属性完全一样
```

四、OOP三大特征（封装、继承、多态）

1、封装-简介

封装(encapsulation)，就是把抽象出的数据[属性]和对数据的操作[方法]封装在一起，数据被保护在内部，程序的其他部分只有通过被授权的操作[方法]，才能对数据进行操作。

2、封装-功能

2.1、隐藏实现细节
2.2、可以对数据进行验证，保证安全合理

3、封装-实现步骤

3.1、将属性进行私有化private
3.2、提供一个公共的(public)set方法，用于对属性判断并赋值
3.3、提供一个公共的(public)get方法，用于获取属性的值

4、继承-简介

继承可以解决代码复用，让我们的编程更加靠近人类思维。当多个类存在相同的属性(变量)和方法时，可以从这些类中抽象出父类，在父类中定义这些相同的属性和方法，所有的子类不需要重新定义这些属性和方法，只需要通过extends来声明继承父类即可。

5、继承-语法

```
class 子类 extends 父类 {
}
```

5.1、子类就会自动拥有父类定义的属性和方法
5.2、父类又叫超类、基类
5.3、子类又叫派生类

6、继承-功能

6.1、提高代码的复用性
6.2、提高代码的扩展性和维护性

7、继承-细节

7.1、子类继承了父类所有的属性和方法，但是父类的私有属性和方法不能在子类中直接访问，要通过公共的方法去访问

7.2、子类必须调用父类的构造器，完成父类的初始化

7.3、当创建子类对象时，不管使用子类的哪个构造器，默认情况下总会去调用父类的无参构造器，如果父类没有提供无参构造器，则必须在子类的构造器中用super去指定使用父类的哪个构造器完成对父类的初始化工作，否则，编译不会通过

7.4、如果希望指定去调用父类的某个构造器，则显式的调用一下：super(参数列表)

7.5、super在使用时，必须放在构造器第一行(super只能在构造器中使用)

7.6、super()和this()都只能放在构造器第一行，因此这两个方法不能共存在一个构造器

7.7、Java所有类都是Object类的子类

7.8、父类构造器的调用不限于直接父类，将一直往上追溯到Object类（顶级父类）

7.9、子类最多只能继承一个父类（指直接继承），即Java中是单继承机制。如果需要A类同时继承B类和C类，可以让A继承B，B继承C

7.10、不能滥用继承，子类和父类之间必须满足 is-a 的逻辑关系

8、多态-简介

方法或对象具有多种形态。是面向对象的第三大特征，多态是建立在封装和继承基础之上的。

9、多态-分类

9.1、方法的多态

- 重写和重载体现多态

9.2、对象的多态

- 一个对象的编译类型和运行类型可以不一致
- 编译类型在定义对象时就确定了，不能改变
- 运行类型是可以变化的
- 编译类型看定义时，也就是=号左边；运行类型看=号右边

10、多态-细节

```
# 前提
    两个对象(类)存在继承关系
```

```
# 向上转型
    1) 本质：父类的引用指向了子类的对象
    2) 语法：父类类型 引用名 = new 子类类型();
    3) 特点：
        - 编译类型看左边，运行类型看右边
        - 可以调用父类中的所有成员(遵守访问原则)
        - 不能调用子类中特有成员
        - 最终运行效果看子类的具体实现
```

```
# 向下转型
    1) 语法：子类类型 引用名 = (子类类型) 父类引用;
    2) 只能强转父类的引用，不能强转父类的对象
    3) 要求父类的引用必须指向的是当前目标类型的对象
    4) 可以调用子类类型中所有的成员
```

```
# 其他细节
    1) 属性没有重写之说，属性的值看编译类型
    2) instatnceof 比较操作符，用于判断对象的运行类型是否为XX类型或XX类型的子类型
    3) 属性看编译，方法看运行
```

11、多态-动态绑定

```
    11.1、调用对象方法时，方法会和该对象的内存地址/运行类型绑定
    11.2、调用对象属性时，属性没有动态绑定，哪里声明哪里使用
```

12、多态-数组

```
    多态数组：数组的定义类型为父类类型，里面保存的实际元素类型为子类类型
```

五、Super关键字

1、简介

```
    super代表父类的引用，用于访问父类的属性、方法、构造器
```

2、语法

```
    2.1、访问父类的属性，不能访问父类的private属性
        - super.属性名;
    2.2、访问父类的方法，不能访问父类的private方法
        - super.方法名(参数列表);
    2.3、访问父类的构造器，只能放在构造器的第一句，且只能出现一句
        - super(参数列表);
```

3、细节

```
    3.1、调用父类的构造器的好处
        - 分工明确
        - 父类属性由父类初始化
        - 子类属性由子类初始化
    3.2、当子类中有父类中的成员（属性和方法）重名时，为了访问父类的成员，必须通过super。如果没有重名，使用super、this、直接访问是一样的效果
    3.3、super的访问不限于直接父类，如果爷爷类和本类中有同名的成员，也可以使用super去访问爷爷类的成员；如果多个基类（上级类）中都有同名的成员，使用super访问就会遵循就近原则。 A->B->C
```

4、super与this区别

No.	关键点	this	super
1	访问属性	访问本类中的属性，如果本类没有此属性，则从父类中继续查找	从父类开始查找属性

No.	关键点	this	super
2	调用方法	访问本类中的方法，如果本类没有此方法，则从父类中继续查找	从父类开始查找方法
3	调用构造器	调用本类构造器，必须放在构造器的首行	调用父类构造器，必须放在子类构造器的首行
4	特殊	表示当前对象	子类中访问父类对象

六、方法重写（overwrite）

1、简介

方法覆盖（重写）就是子类有一个方法，和父类的某个方法的名称、返回类型、参数一致，那么就成子类的这个方法覆盖了父类的方法

2、细节

2.1、子类方法的形参列表、方法名称，要和父类的形参列表、方法名称完全一致
2.2、子类方法的返回类型和父类方法的返回类型一致，或者是父类返回类型的子类
2.3、子类方法不能缩小父类方法的访问权限

3、重载和重写区别

名称	发生范围	方法名	参数列表	返回类型	修饰符
重载(overload)	本类	必须一样	类型、个数、顺序至少有一个不同	无要求	无要求
重写(override)	父子类	必须一样	相同	子类重写的方法中，返回类型和父类返回一致，或者是其返回类型的子类	子类方法不能缩小父类方法的访问权限

七、Object类详解，垃圾回收机制

1、equals

```
# ==和equals的区别
    1.1、==：既可以判断基本类型，也可以判断引用类型
    1.2、==：如果判断基本类型，判断的是值是否相等。如：int i = 10;double d = 10.0;
    1.3、==：如果判断引用类型，判断的是地址是否相等，即判断是不是同一个对象
    1.4、equals：是Object类中的方法，只能判断引用类型
    1.5、默认判断的是地址是否相等，子类中往往重写该方法，用于判断内容是否相等。如：Integer、String
```

```
# 重写equals方法
    判断两个Person对象的内容是否相等，如果两个Person对象的各个属性值都一样，则返回true，反制false
```

2、hashCode

```
    2.1、提高具有哈希结构的容器的效率
    2.2、两个引用，如果指向的是同一个对象，则哈希值肯定是一样的
    2.3、两个引用，如果指向的是不同对象，则哈希值不一样
    2.4、哈希值主要根据地址号来的，但不能完全将哈希值等价于地址
```

3、toString

```
# 简介
    默认返回：包名+类名+@+哈希值的十六进制，对象的属性信息
```

```
# 细节
    3.1、重写toString方法，打印对象或拼接对象时，都会自动调用该对象的toString形式
    3.2、当直接输出一个对象时，会调用并输出该对象的toString方法
```

4、finalize

```
    4.1、当对象被回收时，系统自动调用该对象的finalize方法。子类可以重写该方法，做一些释放资源的操作
    4.2、什么时候被回收：当某个对象没有任何引用时，则jvm就认为这个对象是一个垃圾对象，就会使用垃圾回收机制来销毁该对象，在销毁该对象前，会先调用finelize方法
    4.3、垃圾回收机制的调用，是由系统来决定（即有自己的GC算法），也可以通过System.gc() 主动触发垃圾回收机制
```

八、Debug

1、简介

1.1、断点调试是指在程序的某一行设置一个断点，调试时，程序运行到这一行就会挺住，然后可以一步一步往下调试。调试过程中可以看到各个变量当前的值，出错的话，调试到出错的代码行即显示错误，停下。进行分析进而找到这个Bug

1.2、断点调试是程序员必须掌握的技能

1.3、断点调试也能帮助我们查看java底层源代码的执行过程，提高程序员的JAVA水平

2、快捷键

F7：跳入方法内

F8：逐行执行代码

Shift + F8：跳出

F9：执行到下一个断点