Java学校课程笔记

目标

1.面向对象的基本思想；

2.面向对象的基本特征；

3.开发工具。

Java

1. 面向对象；简单性；跨平台。
2. Java是编译型和解释性两种类型的集合。
3. `.class` 自解码文件。
4. Java虚拟机跨平台原理 ： **Java源代码---->字节码 -----> JVM ， Java虚拟机**

API —— Java应用程序接口

后缀及目录

1. bin：提供JDK工具程序，
2. include :本地方法文件；
4. lib：可执行文件所使用的库文件；
5. src.zip.java提供API类的源代码;
6. .Jar 表示Java的压缩文件。

环境变量注意

“.”表示当前目录。

DOS命令

1. `cd ..`返回上级目录；
2. `cd xxx`进入此文件目录；
3. `dir`查看文件夹目录内容和状态。

Eclipse

开源开发环境

My eclipse ： 安装过固定插件，收费项目。

数据类型

一、标识符和关键字

1.标识符的组成：**“字”“数”“美”“人”“线”。**

2.标识符的命名规则：一个标识符可以由几个英文单词组成；

类名，每个单词的首字母必须大写；

方法名，第一个单词首字母小写，其他单词首字母大写；（驼峰形式）【getRecorName()】

常量名，常量命名全部大写，单词间用下划线隔开；（final）

变量名，驼峰形式；

标识符，驼峰形式；

包名，每个单词每个字母都要小写，多个单词用“ . ”分开；

注意：Java语言区分大小写

3. 数据类型：非数值型，char, boolean, String ， Scanner...
4. 常量与变量：声明及初始化变量的格式：
5. `final` 关键字： 常量命名关键字。

二、String 类

三、Scanner 类

1.**导入Scanner**：`import java.util'`

2.**创建Scanner对象**：`Scanner input =new Scanner(System.in)`;

3.**获得键盘数据**：`int now= input.nextlnt();`

***默认包：**`Java.lang`

常用方法：

1. `String next()` :获得一个字符串；
2. `int nextlnt()` :获得一个整数类型；
3. `double nextDouble()` :获得一个双精度浮点型；
4. `has`

运算符

与C语言基本一致。

1. `instanceof` 是 Java 的一个二元操作符，类似于 ==，>，< 等操作符。

   `instanceof` 是 Java 的保留关键字。它的作用是测试它左边的对象是否是它右边的类的实例，返回 boolean 的数据类型。

***注意：“ == ”表示地址之间的的比较；**

```
/*
 author by runoob.com
 Main.java
 */
import java.util.ArrayList;
import java.util.Vector;

public class Main {

public static void main(String[] args) {
   Object testObject = new ArrayList();
      displayObjectClass(testObject);
   }
   public static void displayObjectClass(Object o) {
      if (o instanceof Vector)
      System.out.println("对象是 java.util.Vector 类的实例");
      else if (o instanceof ArrayList)
      System.out.println("对象是 java.util.ArrayList 类的实例");
      else
      System.out.println("对象是 " + o.getClass() + " 类的实例");
   }
}
```

快捷键

1. `ctrl +shift +o`:快速导包。

类型转换

1.`byte ,short ,int,long,float,double` 从左向右自动转换， 低+高 = 高；先把低转成高 高+高=高； JVM自动转换，自动类型转换；

2.把高的赋给低的时候要进行强制类型转换，或者把靠右的类型赋给靠左边的类型，要进行强制类型转换；

3.强制类型转换的格式：`int a +(int)double`

4.***强制类型转换存在精度缺失。**

字符串的比较

equal方法

Java数组

1. 数组的声明。
2. 数组的初始化：
3.

注意

1. `\t` 表示制表符；
2. `\n` 表示换行；
3. 导包的原因：告诉JVM在哪里寻找类文件夹。

面向对象

一、类和对象

1. 分类：万物皆是对象，通过不同事物的相同特征将实物分成不同的类。
2. **类和对象**： 类包含对象；
3. 类：对象具有的特征——**属性。**
4. 每个对象的每个属性的拥有特定值。
5. 对象具有的各种操作——**方法。**
6. 类：具有相同属性和方法的一组对象的集合，类是对象的抽象，对象是类的实例。
7. 成员变量；
8. 成员方法；
9. 定义类：1）定义类名；2）编写类的属性；3）编写类的方法。

   若类没有属性也没有元素，称作空类
10. 在一个Java文件当中可以包含多个类，Java文件的文件名必须与首个类名称一致。

注意：类名首字母必须大写；对象名要见名知意；

12. 定一个方法：1）访问权限 2）返回值类型 3） 方法名 4）括号里面的参数（可以没有亦可以一个或多个）

```
public int setAge(int age){
    this.age = age;
}
```

13. `this`：指当前调用的对象。
14. **值传参和引用传参**：

1.所有的参数传递都是 传值，从来没有 传引用 这个事实；

2. 所有的参数传递都会在 程序运行栈上 新分配一个 值 的复制品；
3. `java` 只有按值传递，所谓的按地址(引用)传递，也属于按值传递，只不过这个“值”是个地址；
4. 对于引用类型的传参也是传值的，传的是引用类型的值，其实就是对象的地址；
5. `java` 参数是传递值的。
6. `java` 所有对像变量都是对像的引用；
7. 或者说：传递过去的都是拷贝，区别在于拷贝的是基本数据类型还是引用；

8. 函数的形式参数，是传入参数的拷贝；引用变量之间拷贝的是【地址】，基本变量之间拷贝的是 内存中的值 （被称为直接量）；
9. 对象本身，与对象的地址 是2个东西，函数之间如果想【传递对象】，只能通过传递对象的地址来实现；

二、 构造方法重载 (封装)

构造方法用于声明对象时直接对其一些属性初始化。

- 构造方法的作用： 取代用户必须在软件程序当中对每一个成员变量的一步一步地循环“声明+定义”的过程，构造方法将简化用户的操作复杂度，使用户直接输入注册信息，即从用户的角度来直接实现对已声明成员变量的定义。
- 注意1：此类构造方法对任何原有方法和自主创建方法均符合。只要创建类，就会存在无参数构造方法:（无参构造器：Constructor）原有方法会隐藏无参构造器，自行创建的方法需要自行添加。
- 注意2：若自行创建的方法不添加无参构造器，则会在调用时出现如果不赋值给成员变量而出现报错的情况。
- 构造方法的功能是用来构造对象的，构造方法的方法名与类的名称是一样的；
- 构造方法没有返回值。
- 构造方法分 无参构造器 ，有参构造方法。
    - 无参数构造方法默认补充。（隐藏），但写出有参数的构造方法时，就必须把无参数构造方法写出来，因为系统不在补充。

方法名相同，参数个数或参数类型不同。

- 主方法在调用其他方法名相同的不同方法时，会自动选择参数个数和参数类型与之匹配的方法。

三、 Getter && Setter

getter & Setter

四、static 修饰符

1. 静态变量： static 变量称为静态变量，一般情况称为成员变量。
2. 静态方法： static 变量称为静态方法，一般情况称为成员方法。

   *生命周期：什么时候产生什么时候消亡。

   static modifier：

```
3. 静态方法不能访问成员方法，静态变量不能访问成员变量；

4. Static method 可以用类直接访问，
```

3. 静态是由类的产生而产生的。

五、包

1. package语句必须在第一行且只能有一个。
2. import 可以有多个。
3. 包命名要小写。
4. 基本包：`java.util`: 实用类方法。

六、访问控制符

1. private: 同一类；
2. default:同一类，同一包；
3. protected:同一类，同一包，子类；
4. public:同一类，同一包，子类，全局范围。

七、 继承

1. extends 关键字。
2. 子类对象可以调用父类所有的方法；
3. 子类对象可以操作父类中所有的public属性；
4. super()调用的是父类有参数的构造方法；
5. 子类的无参构造器中调用父类的有参构造中的属性；
6. 子类构造方法必须先构造父类方法；
7. 所有类的父类都是Object类的子类；
8. super()：子类构造方法中，无参构造方法中super调用父类的；
9. 子类可以继承父类中 `public` 与`protected` 的方法；
10. 继承默认权限修饰符修饰的属性和方法，但子类与父类必须再同一个包中；
11. 父类`private` 方法子类无法继承；
12. 一个子类只能继承一个类。

- 构造方方法不能继承；
- 继承扶符合 “is-a”的关系。

八、继承

1. 方法名相同；
2. 参数列表相同；
3. 返回值类型相同或是其子类；
4. 访问权限不能严于父类。
5. 父类的静态方法不能被子类覆盖为非静态方法，。。。
6. 父类私有不能被继承；
7. final 变量不能被继承。

九、多态

1. **方法多态：**

   重载：

   重写：

2. **类多态：**

   向上转型： `FatherClass fatherObject = new sonClass();` **自动转换**

   向下转型： `SonClass sonObject = ( SonClass )fatherInstance;` **强制转换**

   - 向下转型必须使用强制转换。
   - 用向上转型解决多次重载一个方法。
   - **向上转型的目的是“统一参数”。**

- 子类向上转型之后变成了一个父类的应用。

十、接口 interface

1. 关键字： `interface`;
2. 接口无法实例化，必须进行实现 “ implement “ .

十一、异常类 Exception

1. 在Catch 代码块中，当前面的的异常的子类都没有捕获某一异常，则最后被Catch 捕获。