Java 基础语法 Java 基本数据类型

# Java 对象和类

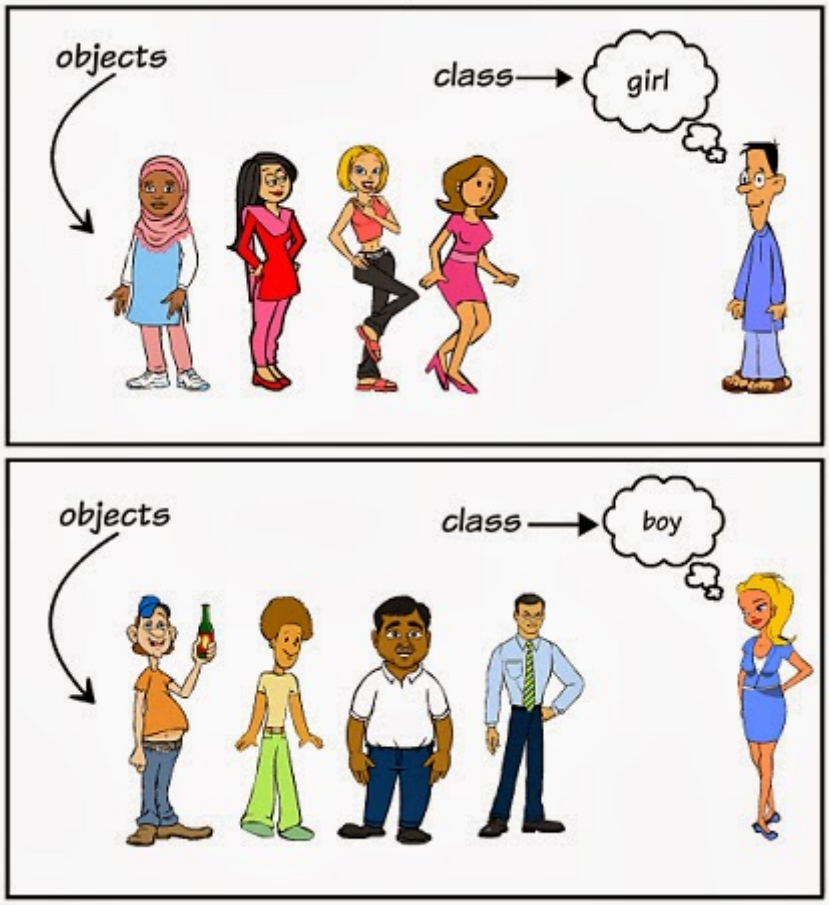
Java作为一种面向对象语言。支持以下基本概念：

- 多态
- 继承
- 封装
- 抽象
- 类
- 对象
- 实例
- 方法
- 重载

本节我们重点研究对象和类的概念。

- **对象**：对象是类的一个实例（**对象不是找个女朋友**），有状态和行为。例如，一条狗是一个对象，它的状态有：颜色、名字、品种；行为有：摇尾巴、叫、吃等。
- **类**：类是一个模板，它描述一类对象的行为和状态。

下图中男孩女孩为类，而具体的每个人为该类的对象：




## Java中的对象

现在让我们深入了解什么是对象。看看周围真实的世界，会发现身边有很多对象，车，狗，人等等。所有这些对象都有自己的状态和行为。

拿一条狗来举例，它的状态有：名字、品种、颜色，行为有：叫、摇尾巴和跑。

对比现实对象和软件对象，它们之间十分相似。

软件对象也有状态和行为。软件对象的状态就是属性，行为通过方法体现。

在软件开发中，方法操作对象内部状态的改变，对象的相互调用也是通过方法来完成。

## Java中的类

类可以看成是创建Java对象的模板。

通过下面一个简单的类来理解下Java中类的定义：

```
public class Dog{
String breed;
int age;
String color;
void barking(){
}
void hungry(){
}
void sleeping(){
}
}
```

一个类可以包含以下类型变量：

- **局部变量**：在方法、构造方法或者语句块中定义的变量被称为局部变量。变量声明和初始化都是在方法中，方法结束后，变量就会自动销毁。
- **成员变量**：成员变量是定义在类中，方法体之外的变量。这种变量在创建对象的时候实例化。成员变量可以被类中方法、构造方法和特定类的语句块访问。
- **类变量**：类变量也声明在类中，方法体之外，但必须声明为static类型。

一个类可以拥有多个方法，在上面的例子中：barking()、hungry()和sleeping()都是Dog类的方法。

## 构造方法

每个类都有构造方法。如果没有显式地为类定义构造方法，Java编译器将会为该类提供一个默认构造方法。

在创建一个对象的时候，至少要调用一个构造方法。构造方法的名称必须与类同名，一个类可以有多个构造方法。

下面是一个构造方法示例：

```
public class Puppy{
public Puppy(){
}
public Puppy(String name){
// 这个构造器仅有一个参数：name
}
}
```

## 创建对象

对象是根据类创建的。在Java中，使用关键字new来创建一个新的对象。创建对象需要以下三步：

- **声明**：声明一个对象，包括对象名称和对象类型。
- **实例化**：使用关键字new来创建一个对象。
- **初始化**：使用new创建对象时，会调用构造方法初始化对象。

下面是一个创建对象的例子：

```
public class Puppy{
public Puppy(String name){
//这个构造器仅有一个参数：name
System.out.println("小狗的名字是 : " + name );
}
public static void main(String[] args){
// 下面的语句将创建一个Puppy对象
Puppy myPuppy = new Puppy( "tommy" );
}
}
```

编译并运行上面的程序，会打印出下面的结果：

```
小狗的名字是 : tommy
```

## 访问实例变量和方法

通过已创建的对象来访问成员变量和成员方法，如下所示：

```
/* 实例化对象 */
ObjectReference = new Constructor();
/* 访问类中的变量 */
ObjectReference.variableName;
/* 访问类中的方法 */
ObjectReference.methodName();
```

## 实例

下面的例子展示如何访问实例变量和调用成员方法：

```
public class Puppy{
int puppyAge;
public Puppy(String name){
// 这个构造器仅有一个参数：name
System.out.println("小狗的名字是 : " + name );
}
public void setAge( int age ){
puppyAge = age;
}
public int getAge( ){
System.out.println("小狗的年龄为 : " + puppyAge );
return puppyAge;
}
public static void main(String[] args){
/* 创建对象 */
Puppy myPuppy = new Puppy( "tommy" );
/* 通过方法来设定age */
myPuppy.setAge( 2 );
/* 调用另一个方法获取age */
myPuppy.getAge( );
/*你也可以像下面这样访问成员变量 */
System.out.println("变量值 : " + myPuppy.puppyAge );
}
}
```

编译并运行上面的程序，产生如下结果：

```
小狗的名字是 : tommy
小狗的年龄为 : 2
变量值 : 2
```

# 源文件声明规则

在本节的最后部分，我们将学习源文件的声明规则。当在一个源文件中定义多个类，并且还有import语句和package语句时，要特别注意这些规则。

- 一个源文件中只能有一个public类
- 一个源文件可以有多个非public类
- 源文件的名称应该和public类的类名保持一致。例如：源文件中public类的类名是Employee，那么源文件应该命名为Employee.java。
- 如果一个类定义在某个包中，那么package语句应该在源文件的首行。
- 如果源文件包含import语句，那么应该放在package语句和类定义之间。如果没有package语句，那么import语句应该在源文件中最前面。
- import语句和package语句对源文件中定义的所有类都有效。在同一源文件中，不能给不同的类不同的包声明。

类有若干种访问级别，并且类也分不同的类型：抽象类和final类等。这些将在访问控制章节介绍。

除了上面提到的几种类型，Java还有一些特殊的类，如：内部类、匿名类。

---

# Java包

包主要用来对类和接口进行分类。当开发Java程序时，可能编写成百上千的类，因此很有必要对类和接口进行分类。

# Import语句

在Java中，如果给出一个完整的限定名，包括包名、类名，那么Java编译器就可以很容易地定位到源代码或者类。Import语句就是用来提供一个合理的路径，使得编译器可以找到某个类。

例如，下面的命令行将会命令编译器载入java_installation/java/io路径下的所有类

```
import java.io.*;
```

---

# 一个简单的例子

在该例子中，我们创建两个类：**Employee** 和 **EmployeeTest**。

首先打开文本编辑器，把下面的代码粘贴进去。注意将文件保存为 Employee.java。

Employee类有四个成员变量：name、age、designation和salary。该类显式声明了一个构造方法，该方法只有一个参数。

**Employee.java 文件代码：**

```
import java.io.*;
public class Employee{
String name;
int age;
String designation;
double salary;
// Employee 类的构造器
public Employee(String name){
```

```
this.name = name;
}
// 设置age的值
public void empAge(int empAge){
age = empAge;
}
/* 设置designation的值*/
public void empDesignation(String empDesig){
designation = empDesig;
}
/* 设置salary的值*/
public void empSalary(double empSalary){
salary = empSalary;
}
/* 打印信息 */
public void printEmployee(){
System.out.println("名字:"+ name );
System.out.println("年龄:" + age );
System.out.println("职位:" + designation );
System.out.println("薪水:" + salary);
}
}
```

程序都是从main方法开始执行。为了能运行这个程序，必须包含main方法并且创建一个实例对象。

下面给出EmployeeTest类，该类实例化2个 Employee 类的实例，并调用方法设置变量的值。

将下面的代码保存在 EmployeeTest.java文件中。

**EmployeeTest.java 文件代码：**

```
import java.io.*;
public class EmployeeTest{
public static void main(String[] args){
/* 使用构造器创建两个对象 */
Employee empOne = new Employee("RUNOOB1");
Employee empTwo = new Employee("RUNOOB2");
// 调用这两个对象的成员方法
empOne.empAge(26);
empOne.empDesignation("高级程序员");
empOne.empSalary(1000);
empOne.printEmployee();
empTwo.empAge(21);
empTwo.empDesignation("菜鸟程序员");
empTwo.empSalary(500);
empTwo.printEmployee();
}
}
```

编译这两个文件并且运行 EmployeeTest 类，可以看到如下结果：

```
$ javac EmployeeTest.java
$ java EmployeeTest
名字:RUNOOB1
```

```
年龄:26
职位:高级程序员
薪水:1000.0
名字:RUNOOB2
年龄:21
职位:菜鸟程序员
薪水:500.0
```

Java 基础语法 Java 基本数据类型