**今天完成对的事情：**

**第一部分：基础知识**

**内容预览：**

* **JAVA基本语法要点**
* **JAVA对象和类**
* **JAVA构造方法**
* **JAVA创建对象**
* **访问实例变量和方法**
* **源文件声明规则**
* **JAVA包的引用**

1. **JAVA基本语法**

**Java编程时需要注意以下几点：**

* **大小写敏感**

Java是大小写敏感的，这就意味着标识符Hello与hello是不同的

* **类名**

类名的首字母应该大写。如果类名由若干单词组成，那么每个单词的首字母应该大写，例如 MyFirstJavaClass

* **方法名**

所有的方法名都应该以小写字母开头。如果方法名含有若干单词，则后面的每个单词首字母大写。

* **源文件名**

源文件名必须和类名相同。当保存文件的时候，你应该使用类名作为文件名保存（切记Java是大小写敏感的），文件名的后缀为.java。

* **主方法入口**

所有的Java 程序由public static void main(String []args)方法开始执行。

1. **继承**

在Java中，一个类可以由其他类派生。如果要创建一个类，而且已经存在一个类具有你所需要的属性或方法，那么你可以将新创建的类继承该类。

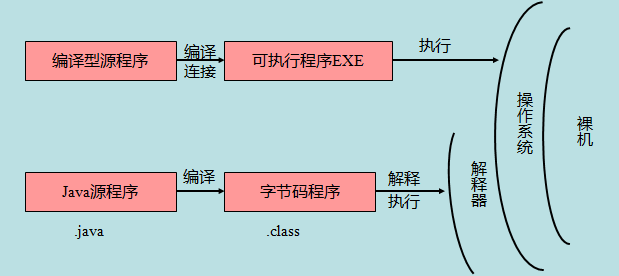
1. **接口**

在Java中，接口为对象间相互通信的协议。

接口只定义派生要用到的方法，但是方法的具体实现完全取决于派生类。

1. **JAVA源码与编译型运行区别**

如下图所示



1. **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

**这里重点研究对象和类的概念**

* **对象**

对象是类的一个实例（拥有具体属性的实际对象）

例如狗是一个种类的称呼，而一条黑色沙皮狗就具有真实属性（状态：颜色、名字、品种等）（行为：摇尾巴、叫、吃等）

* **类**

类是创建对象的“模版”

类定义对象可以包含的值，以及可以对该对象执行的操作。

面向对象就是对对象执行操作，而不是C++类对自己执行操作。

1. JAVA中的对象

对比现实对象和软件对象，它们之间十分相似。

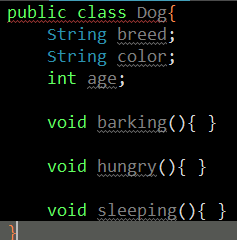
软件对象也有状态和行为。

* + - 软件状态就是属性
    - 软件行为通过方法体现

1. JAVA中的类

类可以看成是创建Java对象的模板。

以下代码示例来展示JAVA中类的定义：

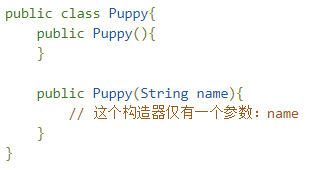


1. **构造方法**

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。**

以下为一个构造方法的示例：

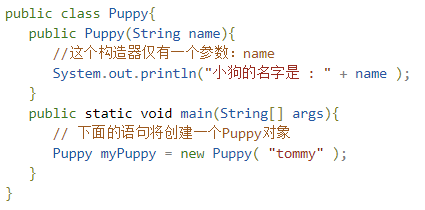


1. **创建对象**

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**。**

以下是一个简单创建对象的示例



以下是一个向上转型（upcasting）创建对象的示例



**创建对象的过程：**

* **声明**

**声明一个对象的名称为b，并使对象的类型转换为Animal的类型**

* **实例化**

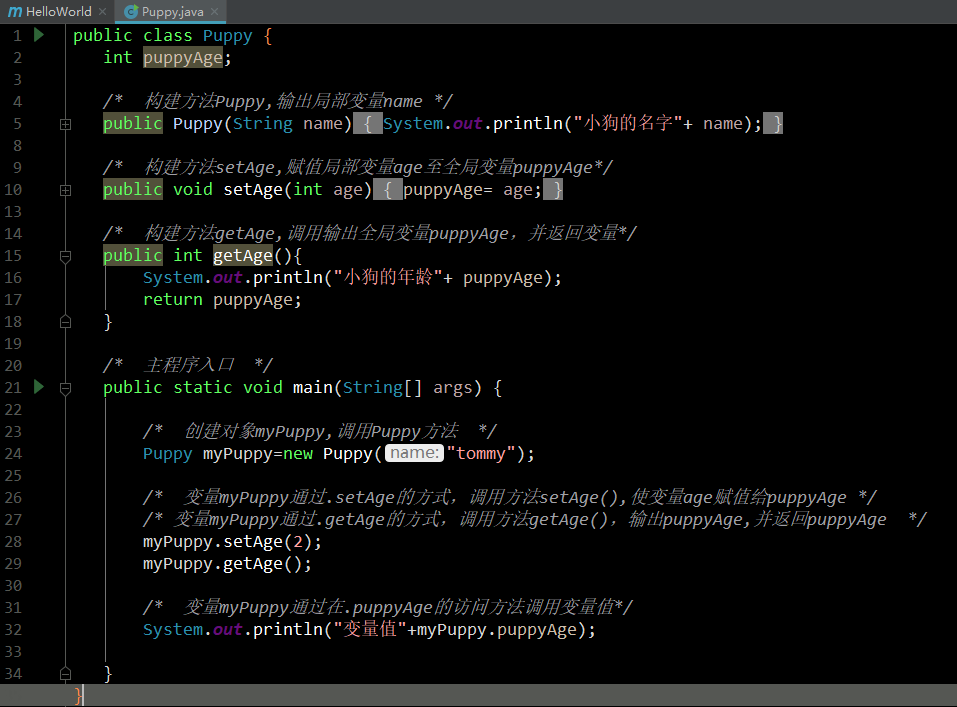
**根据new关键词，创建一个以Dog()类为模版，并被b引用的对象**

* **初始化**

**使用new创建对象时，会调用Dog()的构造方法初始化对象b**

1. **访问实例变量和方法**

**自己尝试编写了一遍代码，并且附注了自己的理解**



1. **源文件声明规则**

**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，要特别注意源文件声明规则。**

* **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**

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。

1. **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。

1. **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 import java.io.\*;**

这条指令会命令编译器载入java\_installation/java/io路径下的所有类

1. **简单的示例**

在该示例中，创建两个类：Employee 和 EmployeeTest

1. **在同一文件夹内，如果声明是public或protected或pravite的类，不需要用import引用**
2. **在不同文件夹内，需要用import引用**

// 文件要与类名一致 //

**1# Employee类**

**该类有四个成员变量：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。

**2# 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



**第二部分：修真任务**

**内容预览：**

* Spring基础认识
* Spring的核心机制
  + Bean机制
  + 依赖注入

1. **Sping基础认识**

Spring是一个从实际开发中抽取出来的框架，因此它完成了大量开发中的通用步骤，留给开发者的只是特定应用的相关部分，从而大大提高了企业应用的开发效率。

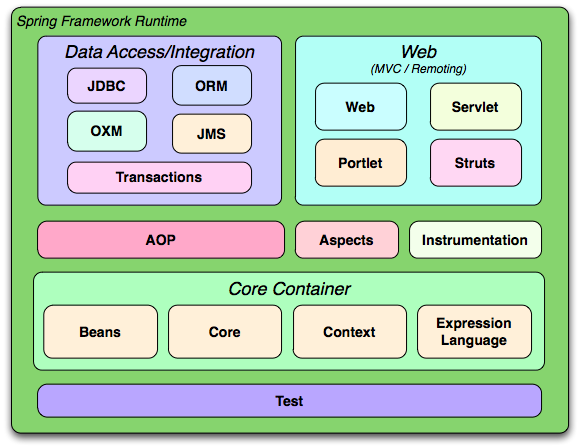
**Spring优点有如下几条：**

* 低侵入式设计，代码的污染极低
* 独立于各种应用服务器

基于Spring框架的应用，可以真正实现Write Once，Run Anywhere的承诺

* Spring的IoC容器降低了业务对象替换的复杂性，提高了组件之间的解耦
* Spring的AOP支持允许将一些通用任务如安全、事务、日志等进行集中式管理，从而提供了更好的复用
* Spring的ORM和DAO提供了与第三方持久层框架的良好整合，并简化了底层的数据库访问
* Spring的高度开放性，并不强制应用完全依赖于Spring，开发者可自由选用Spring框架的部分或全部

**Spring框架的组成结构图如下所示：**



\*\*\*\*\* 每个部分都了解一遍 \*\*\*\*\*

1. **Spring的核心机制**
2. **管理Bean（这个Bean是真的不能理解啊啊啊啊啊啊啊啊啊）**

程序主要是通过Spring容器来访问容器中的Bean

1. **Bean的概念（终于理解了，壮壮师兄一番点播，恍然大悟）**

Bean是Spring容器（框架）里用来管理类的机制

**在Spring框架中，通常把用一种功能的类放在一块，简称一个盒子内。**

**然后就把这个盒子内的类定义一个名称“bean”（豆子），并赋予唯一ID。**

**下次使用这种类的时候，直接引用bean（豆子）+ID就行。**

言简意赅的称呼引用管理机制------类的管理

1. **ApplicationContext是Spring容器最常用的接口**

**该接口有如下两个实现类：**

* ClassPathXmlApplicationContext

从类加载路径下搜索配置文件，并根据配置文件来创建Spring容器

* FileSystemXmlApplicationContext

从文件系统的相对路径或绝对路径下去搜索配置文件，并根据配置文件来创建Spring容器

1. **依赖注入**

**Spring框架的核心功能有两个：**

* Spring容器实现功能的超级大工厂，负责创建、管理所有的Java对象

这些Java对象被称为Bean。

* Spring容器管理容器中Bean之间的依赖关系

Spring使用一种被称为"依赖注入"的方式来管理Bean之间的依赖关系。

使用依赖注入，不仅可以为Bean注入普通的属性值，还可以注入其他Bean的引用。**依赖注入是一种优秀的解耦方式**，其可以让Bean以配置文件组织在一起，而不是以硬编码的方式耦合在一起。

1. **理解依赖注入**

当某个Java对象（调用者）需要调用另一个Java对象（被依赖对象）的方法时，在传统模式下通常有两种做法：

**1# 原始做法**

调用者主动创建被依赖对象，然后再调用被依赖对象的方法

**2# 简单工厂模式**

调用者先找到被依赖对象的工厂，然后主动通过工厂去获取被依赖对象，最后再调用被依赖对象的方法

**注意上面的主动二字，这必然会导致调用者与被依赖对象实现类的硬编码耦合，非常不利于项目升级的维护。使用Spring框架之后，调用者无需主动获取被依赖对象，调用者只要被动接受Spring容器为调用者的成员变量赋值即可，由此可见，使用Spring后，调用者获取被依赖对象的方式由原来的主动获取，变成了被动接受——所以Rod Johnson称之为控制反转。**

**另外从Spring容器的角度来看，Spring容器负责将被依赖对象赋值给调用者的成员变量——相当于为调用者注入它依赖的实例，因此称之为依赖注入。**

**明天计划完成的事情：**

1. JAVA基础知识的学习

1# 数据类型

2# 变量类型

3# 修饰符

4# 运算符

5# 循环/条件语句

1. Spring框架的继续学习

**遇到的问题：**

1. 什么是Bean的概念，看到Spring架构的基础特点时各种不能理解

壮壮师兄一番点播，恍然大悟，师兄强无敌。

**在Spring框架中，通常把用一种功能的类放在一块，简称一个盒子内**

**然后就把这个盒子内的类定义一个名称“bean”（豆子），并赋予唯一ID**

**下次使用这种类的时候，直接引用bean（豆子）+ID就行**

**言简意赅的称呼引用管理机制------类的管理**

1. 依赖注入

学习依赖注入概念过程中发现牵扯到其他的概念，例如工厂模式

在搜索相关资料后，大概理解了概念

**收获：**

1. 学习了JAVA的基础知识
2. 初步了解Spring框架
3. 理解了Bean的概念
4. 理解了工厂模式的好处