**今天完成的事情：**

**第一部分：JAVA基础**

**学习预览：**

* 接口
* 包

**第二部分：修真任务**

**学习预览：**

* Spring框架的重新认识

**第一部分**

1. **接口**

接口（interface）是一个抽象类型，是抽象方法的集合，接口通常以interface来声明

一个类通过继承接口的方式，来继承接口的抽象方法

**接口的保存方式：**

* 一个接口可以有多个方法
* 接口文件保存在 .java 结尾的文件中，文件名与接口名一致
* 接口的字节码文件保存在 .class 结尾的文件中
* 接口相应的字节码文件必须在与包名称相匹配的目录结构中

**接口与类的区别：**

* 接口没有构造方法
* 接口不能用于实例化对象
* 接口中的方法不能有具体功能
* 接口中不能含有静态代码块以及静态方法 ( 用static修饰的方法 )
* 接口中的所有的方法必须是抽象方法（ 默认声明为public abstract ）
* 接口中的成员变量必须是静态的，( 默认声明为public static final变量 )

（并且只能是 public，用 private 修饰会报编译错误）

* 接口支持多继承，一个类却可以实现多个接口

1. **接口的声明**

接口的声明特性：

* **接口是隐性抽象的**

当声明一个接口的时候，不必使用abstract关键字（默认）

* **接口中的方法也是隐性抽象的**

当声明一个接口方法时，也不必使用abstract关键字（默认）

* **接口中的方法都是公共的**

当声明一个接口方法时，使用public修饰符

语法格式为：

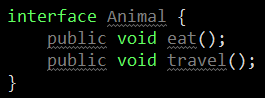
**[ 可见度 ] interface 接口名称 [ extends 其他的接口名名 ] {**

**// 声明变量**

**// 抽象方法**

**}**

实例如下：



**接口有以下特性：**

* 接口是隐式抽象的，当声明时不必使用abstract关键字
* 接口中的方法是隐式抽象的，当声明时不需要abstract关键字
* 接口中的方法都是公有的

1. **接口的实现：**

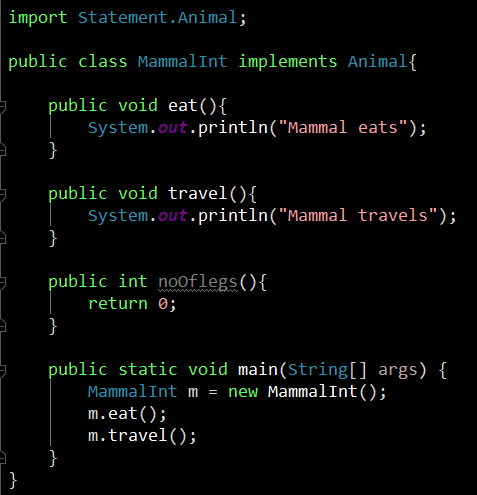
* 当类实现接口的时候，类要实现接口中所有的方法

否则，类必须声明为抽象的类

* 类使用implements关键字实现接口（继承）

在类声明中，Implements关键字放在class声明后面

实例如下：



**类MammalInt 实现（继承）了Animal接口**

**并重写了接口Animal中的方法**

**重写接口方法时，注意事项：**

* 类在实现接口的方法时，不能抛出强制性异常

只能在接口中，或者继承接口的抽象类中抛出该强制性异常

* 类在重写方法时要保持一致的方法名

并且应该保持相同或者相兼容的返回值类型

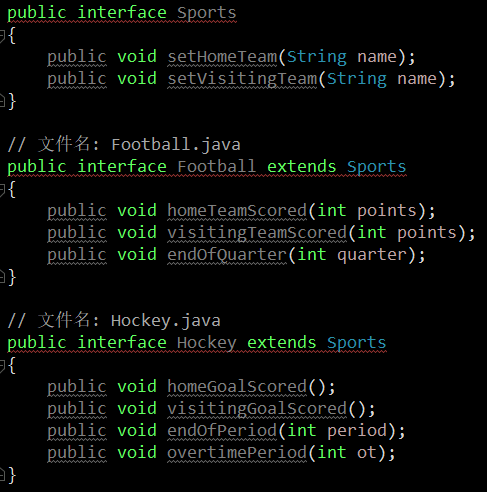
* 如果实现接口的类是抽象类，那么就没必要实现该接口的方法

1. **接口的继承**

一个接口能继承另一个接口，与类的继承相似

接口的继承使用extends关键字，子接口继承父接口的方法

实例如下：



**上面的Sports接口被Hockey和Football接口继承：**

* 其中，Hockey接口自己声明了四个方法，从Sports接口继承了两个方法

这样，实现Hockey接口的类需要实现六个方法

* 相似的，Football接口自己声明了三个方法，从Sports接口继承了两个方法

这样，实现Football接口的类需要实现五个方法

1. **接口的多继承**

在JAVA中，类的多继承是不合法的，但接口可以多继承

在接口的多继承中extends关键字只需要使用一次，在其后跟着继承接口，例如：

**public interface Hockey extends Sports, Event**

1. **标记接口**

标记接口就是没有任何方法的接口

最常用的继承接口是没有包含任何方法的接口

**标记接口是没有任何方法和属性的接口：**

它仅仅表明它的类属于一个特定的类型,供其他代码来测试允许做一些事情

**标记接口作用：**

它就是给某个对象打个标（盖个戳），使对象拥有某个或某些特权

例如：java.awt.event包中的MouseListener接口继承的java.util.EventListener接口

定义如下：

**package java.util;**

**public interface EventListener { }**

**标记接口主要用于以下两种目的：**

* **建立一个公共的父接口：**

正如EventListener接口，这是由几十个其他接口扩展的Java API，你可以使用一个标记接口来建立一组接口的父接口

* **向一个类添加数据类型：**

这种情况是标记接口最初的目的，实现标记接口的类不需要定义任何接口方法(因为标记接口根本就没有方法)，但是该类通过多态性变成一个接口类型

**第二部分**

1. **Spring框架的重新认识**
2. **Spring是什么**

Spring是一款为了解决企业应用开发的复杂性而创建的轻量级Java框架。框架的主要优势之一就是其分层架构，分层架构允许使用者选择哪一个组件，同时为J2EE应用程序开发提供集成的框架。

从简单性、可测试性和松耦合的角度而言，任何Java应用都可以从Spring中受益，Spring的核心是控制反转IOC和面向切面AOP

1. **Spring的特征**
2. **轻量**

从大小与开销两方面而言Spring都是轻量的。完整的Spring框架可以在一个大小只有1M多的JAR文件里发布，并且Spring所需的处理开销也是微不足道的。

Spring是非侵入式的： Spring应用中的对象不依赖于Spring的特定类。

1. **控制反转IoC**

Spring通过一种控制反转IoC的编程思想降低了程序的耦合性。

当应用了IoC，一个对象依赖的其他对象会通过被动的方式传递进来，而不是这个对象自己去创建或者自己去查找依赖对象。都由IoC组件完成

1. **面向切面AOP**

Spring提供了面向切面编程的丰富支持，允许通过分离应用的业务逻辑与系统级服务进行内聚性的开发。

应用程序只实现它们应该做的----完成业务逻辑，仅此而已，它们并不负责（甚至意识不到）其它的系统级关注点，例如日志或事物支持。

1. **容器**

Spring包含并管理应用对象的配置和生命周期，在这个意义上它是一种容器，你可以配置你的每个bean如何被创建（基于一个可配置原型，你的bean可以创建一个单独的实例或者每次需要时都生成一个新的实例），以及如何相互关联。

1. **框架**

Spring可以将简单的组件配置、组合成为复杂的应用。在Spring中，应用对象被声明式地组合，例如配置在一个XML文件里。Spring也提供了很多基础功能（事务管理、持久化框架集成等），将应用逻辑的开发留给开发者。

所有Spring的这些特征使得开发者能够编写更干净、更可管理、更易于测试的代码，它们也为Spring中的各种模块提供了基础支持。

1. **Spring的特点**
2. **方便解耦**

通过Spring提供的IOC容器，我们可以将对象之间的依赖关系交由Spring进行控制，避免硬编码所造成的过度耦合。

有了Spring，用户不必再为单实例模式类、属性文件解析等这些很底层的需求编写代码，可以更专注于上层的应用。

1. **AOP编程的支持**

通过Spring提供的AOP功能，方便进行面向切面的编程，许多不容易用传统OOP实现的功能可以通过AOP轻松应付。

1. **声明的方式管理对象**

在Spring中，我们可以从单调烦闷的事务管理代码中解脱出来，通过声明式灵活地进行事物的管理，提高开发效率和质量。

1. **方便集成各种优秀框架**

Spring不排斥各种优秀的开源框架，相反，Spring可以降低各种框架的使用难度，Spring提供了对各种优秀框架如Structs、Hibernate、Hessian、Quartz等的直接支持。

1. **降低Java EE API的使用难度**

Spring对很多难用的Java EE API（如JDBC、JavaMail、RMI等）提供了一个薄薄的封装层，通过Spring的简易封装，这些Java EE API的使用难度大为降低

1. **Java源码是经典学习范例**

Spring的源码设计精妙、结构清晰、独具匠心，处处体现着对Java设计模式灵活运用以及对Java技术的高深造诣。

Spring框架源码无疑是Java技术的最佳实践范例，如果想在短时间内迅速提高自己的Java技术水平和应用开发水平，学习和研究Spring的源码将使你获益匪浅。

1. **Spring环境搭建以及.xml文件自动提示设置**
2. **导入依赖包**
3. 通过Spring Assistant构建Spring项目可以免除导入
4. 通过POM文件添加依赖，必备依赖包有：

org.springframework.context

org.springframework.expression

org.springframework.core

org.springframework.beans

org.springframework.aop

commons-logging

1. **创建XML文件**
2. **Spring的使用是通过写配置文件的**

所以右键src->new->File，随便命名，以.xml结尾即可

1. **XML文件里面写入最基础的模板，往后所有功能都在这个模板上扩展：**

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns=<http://www.springframework.org/schema/beans>

xmlns:xsi=<http://www.w3.org/2001/XMLSchema-instance>

xsi:schemaLocation="http://www.springframework.org/schema/beans

<http://www.springframework.org/schema/beans/spring-beans-4.2.xsd>">

<!-- 配置service

<bean> 配置需要创建的对象

id ：用于之后从spring容器获得实例时使用的

class ：需要创建实例的全限定类名

-->

<bean id="User" class="java.User"></bean>

</beans>

**明天计划完成的事情：**

今天将JAVA基础知识正式学习完成，开始Spring框架的构建

现在遇到的问题是，还对Spring框架的操作不熟悉，不太明确应该从哪里开始

目前在做的事是配置好bean的配置文件，尝试构建程序，并测试成功

**遇到的问题：**

今天开始构建Spring框架，还是没有找到上手的入手点

目前在配置bean，并用实例来做测试，但是测试起来总是错误很多

希望明天能解决这个问题，开始其他构建的搭建