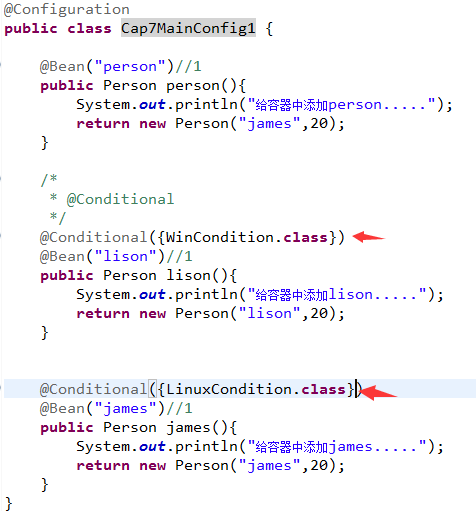
**四 CAP5章节 @Conditional条件注册bean**

1,将IOC容器注册bean时, 当操作系统为WINDOWS时,注册Lison实例; 当操作系统为LINUX时, 注册James实例,此时要用得@Conditional注解进行定制化条件选择注册bean;

2,新建Cap7MainConfig1.java,



新建WinCondition.java类做为条件类, 同时必须得实现spring提供的Confition接口



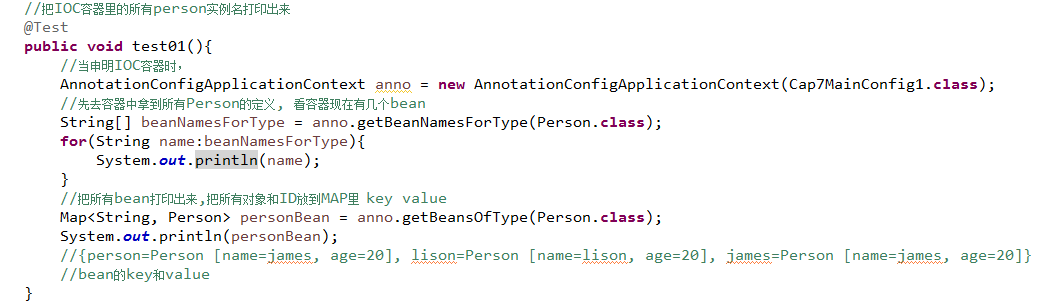
新建 LinuxCondition条件类, 用来处理操作系统为LINUX的bean注册



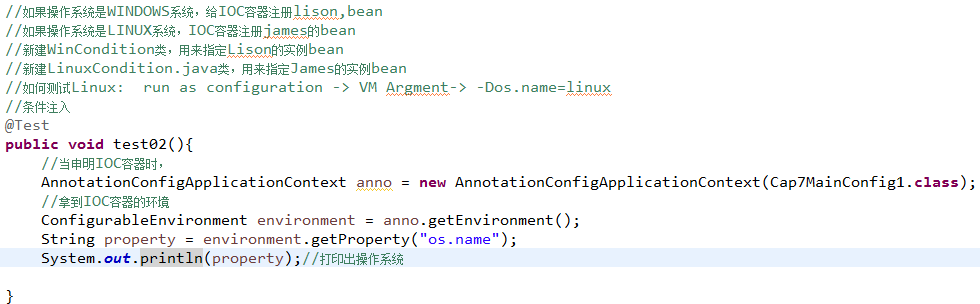
2, 把IOC容器里的所有person实例名打印出来(为了看效果,刚开始在配置类可以**不加@Conditinal**)

@Test

**public** **void** test01()



3, 新建test02(), 测试@Conditional 条件加载bean到IOC容器(**加上@Conditinal**)



当引入@Conditional时, 容器可以选择性的注册bean.

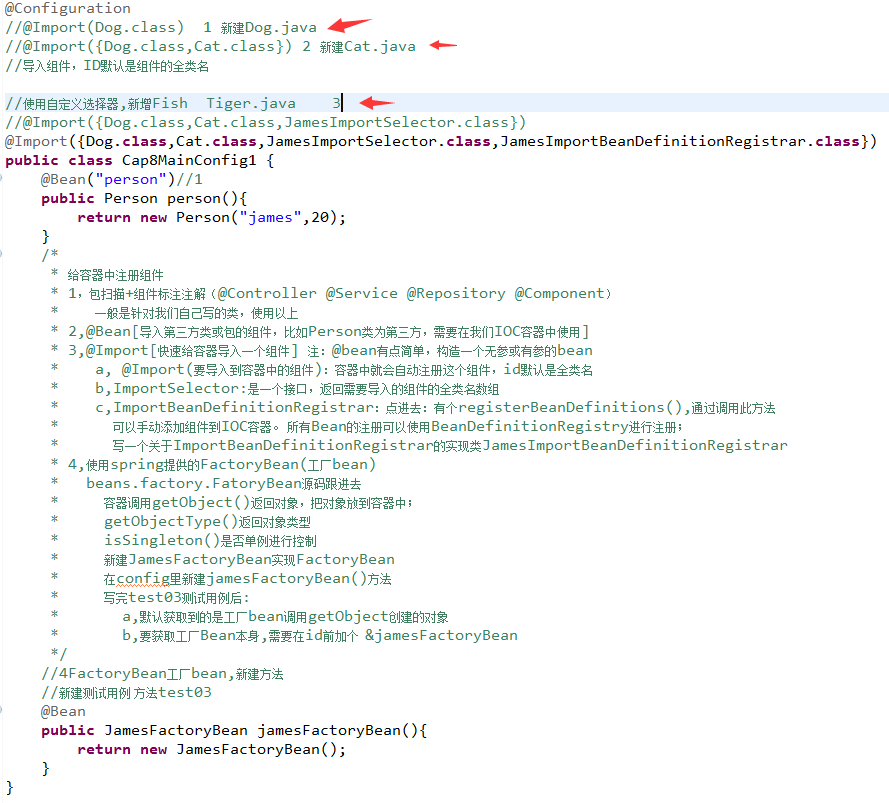
**五 CAP6章节 @Import注册bean**

同样按流程先新建Cap8MainConfig1.java配置类

新建Dog.ava类----->**public** **class** Dog {}

新建Cat.java类---->**public** **class** Cat{}

按以下1,2,3箭头步骤分别导入多个类,使用import将dog, cat的bean注册到容器中,并测试打印,看容器中是否已加载此类



**分别使用**

@Import(Dog.class)

@Import({Dog.class,Cat.class})

ImportSelector可以批量导入组件的全类名数组,自定义逻辑返回需要导入的组件JamesImportSelector.java

@Import({Dog.class,Cat.class,JamesImportSelector.class})

**新建JamesImportSelector, 新建Fish Tiger类(与建Cat和Dog一样)**



怎么测呢?



当然,除了以上,还可以通过ImportBeanDefinitionRegistrar自定义注册,向容器中注册bean;

@Import({Dog.**class**,Cat.**class**,JamesImportSelector.**class**,JamesImportBeanDefinitionRegistrar.**class**})

新建**JamesImportBeanDefinitionRegistrar**自定义注册类,实现bean注册



当然除了以上加载方式,还可以通过实现FactoryBean接口方式来加载bean

\* 使用spring提供的FactoryBean(工厂bean)

\* beans.factory.FatoryBean源码跟进去

\* 容器调用getObject()返回对象，把对象放到容器中；

\* getObjectType()返回对象类型

\* isSingleton()是否单例进行控制

\* 新建JamesFactoryBean实现FactoryBean

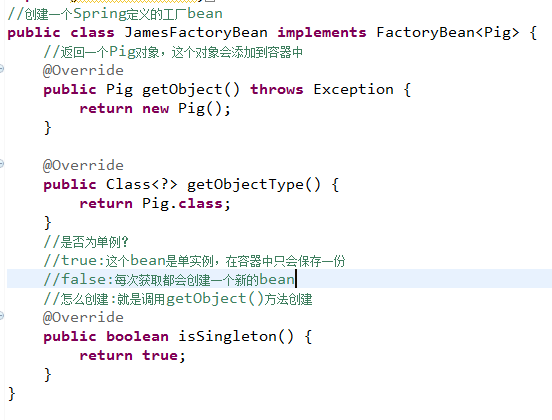
\* 在config里新建jamesFactoryBean()方法

\* 写完test03测试用例后:

\* a,默认获取到的是工厂bean调用getObject创建的对象

\* b,要获取工厂Bean本身,需要在id前加个 &jamesFactoryBean

\*/



把自定义的**JamesImportBeanDefinitionRegistrar**加入配置类进行测试

