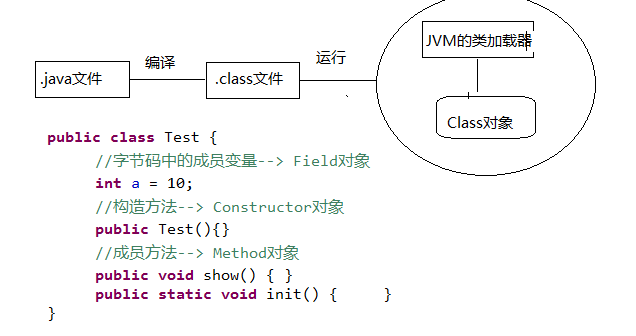
# Java中的反射

反射技术应用 :编写框架或组件。

# Class<>对象

.class文件被加载到内存后 ,JVM会生成的以下相关的对象 ，可以通过反射的API操作相关类的对象 。



1. 获取Class类的三种方法

1. 类名.class

2. 对象.getClass()

3. Class.forName("com.koterwong.Test") :传入完整的类名,最常见

1. 通过Class对象获得构造器、成员变量、调用成员方法。

**//该例子写出获取对象的方法.**

**//定义目标类， 要进行反射的类。**

**class** Person{

**public** **int** id;

**private** String name;

**public** Person() { }

**public** Person(String name) {

**this**.name = name;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

}

**public** **class** Test{

**public** **static** **void** main(String[] args) **throws** Exception {

Class personClass = Class.*forName*("Person");

// 获取所有的构造器对象返回hiconstructor数组

Constructor []constructors = personClass.getConstructors();

//获取指定参数的构造器对象。

Constructor constructor =personClass.getConstructor(String.**class**);

//用过newInstance()方法目标类的对象。

Person person1 = (Person)constructor.newInstance("koter");

System.*out*.println(person1.getName());

//2. 获取属性getDeclaredFields(),包括私有属性.

Field[] fields = personClass.getDeclaredFields();

//获取属性getDeclaredFields("属性名") ,获取指定属性

Field nameField = personClass.getDeclaredField("name");

//因为是私有属性,所以要设置可访问

nameField.setAccessible(**true**);

//属性依赖对象而存在,要声明set和get那个对象的属性

nameField.set(person1, "wong");

Object object = nameField.get(person);

System.*out*.println(object);

// 3. 获取成员方法 ,通过method的invoke方法传入指定参数可执行

// incoke调用静态方法,第一个参数不用传入对象,传入null即可.

Method[] methods = personClass.getDeclaredMethods();

Method methodSetName = personClass.getDeclaredMethod

("setName",String.**class**);

methodSetName.invoke(person1,"koterwong");

Method methodGetName = personClass.getDeclaredMethod("getName");

System.*out*.println(methodGetName.invoke(mPerson));

}

}

案例：晚会案例

1. 编写晚会程序,测试程序

2. 为了程序更好维护和扩展 ,做什么与怎么做分离

问题：在晚会类 维护所有出演演员 ---- 不想修改晚会类 ---- 工厂结构 （中介）

3. 用工厂将做什么与怎么做进行解耦合

4、将演员实例类写入配置文件

工厂+反射+配置文件 --- 编写便于扩展、便于维护程序

**public** **class** Factory {

// 提供准备 歌手方法

**public** **static** Singable getSingable() {

// 读取配置文件 src/party.properties

String className = ResourceBundle.getBundle("party").getString(

"Singable");

// 已知完整类名 获得对象

**try** {

Object obj = Class.*forName*(className).newInstance();

**return** (Singable) obj;

} **catch** (Exception e) {

e.printStackTrace();

**throw** **new** RuntimeException("中介出问题了！");

}

}

// 提供准备 舞蹈方法

**public** **static** Danceable getDanceable() {

// 读取配置文件 src/party.properties

String className = ResourceBundle.getBundle("party").getString(

"Danceable");

// 已知完整类名 获得对象

**try** {

Object obj = Class.*forName*(className).newInstance();

**return** (Danceable) obj;

} **catch** (Exception e) {

e.printStackTrace();

**throw** **new** RuntimeException("中介出问题了！");

}

}

// 提供准备 演员方法

**public** **static** Performable getPerformable() {

// 读取配置文件 src/party.properties

String className = ResourceBundle.getBundle("party").getString(

"Performable");

// 已知完整类名 获得对象

**try** {

Object obj = Class.*forName*(className).newInstance();

**return** (Performable) obj;

} **catch** (Exception e) {

e.printStackTrace();

**throw** **new** RuntimeException("中介出问题了！");

}

}

}