千锋教育
www.mobiletrain.org

千锋Java学院出品

反 射

Java Platform Standard Edition

良心教育，匠心品质

课程目标

CONTENTS

千锋教育
www.mobiletrain.org

ITEMS 1 什么是类对象

ITEMS 2 获取类对象的方法

ITEMS 3 常见操作

ITEMS 4 设计模式介绍

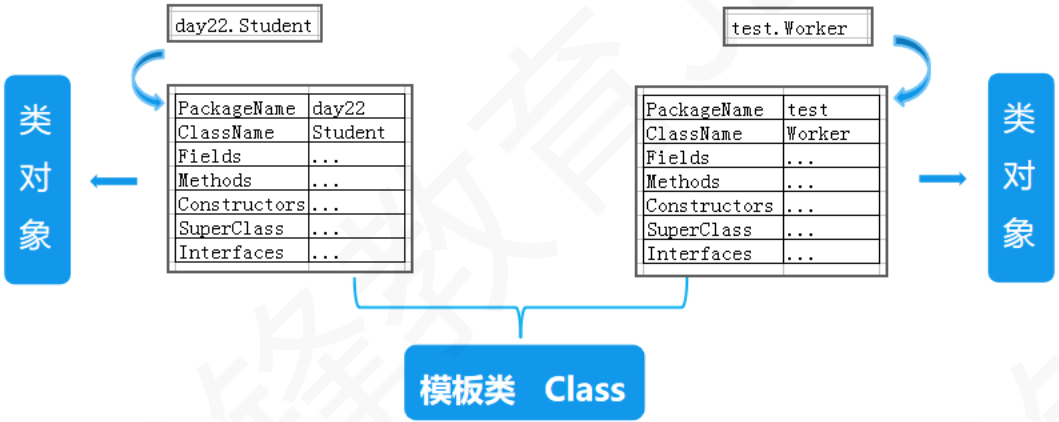
ITEMS 5 单例设计模式

ITEMS 6 工厂设计模式

什么是类对象

千锋教育 www.mobiletrain.org

- 类的对象：基于某个类 new 出来的对象，也称为实例对象。
- 类对象：类加载的产物，封装了一个类的所有信息（类名、父类、接口、属性、方法、构造方法）



获取类对象

- 通过类的对象，获取类对象
  - Student s = new Student();
  - Class c = s.getClass();
- 通过类名获取类对象
  - Class c = 类名.class;
- 通过静态方法获取类对象
  - Class c=Class.forName(“包名.类名”);

常用方法

- public String getName()
- public Package getPackage()
- public Class<? super T> getSuperclass()
- public Class<?>[] getInterfaces()
- public Field[] getFields()
- public Method[] getMethods()
- public Constructor<?>[] getConstructors()
- public T newInstance()

设计模式

- 什么是设计模式：
  - 一套被反复使用、多数人知晓的、经过分类编目的、代码设计经验的总结。
  - 可以简单理解为特定问题的固定解决方法。
- 好处：
  - 使用设计模式是为了可重用代码、让代码更容易被他人理解、保证代码可靠性、重用性。
- 在Gof的《设计模式》书中描述了23 种设计模式。

工厂设计模式

- 开发中有一个非常重要的原则“开闭原则”，对拓展开放、对修改关闭。
- 工厂模式主要负责对象创建的问题。
- 可通过反射进行工厂模式的设计，完成动态的对象创建。

单例模式

- 单例（Singleton）：只允许创建一个该类的对象。

- 方式1：饿汉式（类加载时创建，天生线程安全）

```
class Singleton {

    private static final Singleton instance = new Singleton();

    private Singleton(){}

    public static Singleton getInstance(){
        return instance;
    }
}
```

单例模式

- 方式2：懒汉式（使用时创建，线程不安全，加同步）

```
class Singleton{
    private static Singleton instance = null;

    private Singleton(){}

    public static synchronized Singleton getInstance(){
        if(instance == null){
            instance = new Singleton();
        }
        return instance;
    }
}
```

单例模式

- 方式3：懒汉式（使用时创建，线程安全）

```
class Singleton {

    private Singleton() {}

    private static class Holder {
        static Singleton s = new Singleton();
    }

    public static Singleton instance() {
        return Holder.s;
    }
}
```

枚举

- 什么是枚举：
  - 枚举是一个引用类型，枚举是一个规定了取值范围的数据类型。
- 枚举变量不能使用其他的数据，只能使用枚举中常量赋值，提高程序安全性。
- 定义枚举使用enum关键字。
- 枚举的本质：
  - 枚举是一个终止类，并继承Enum抽象类。
  - 枚举中常量是当前类型的静态常量。

注解

- 什么是注解：
  - 注解(Annotation)是代码里的特殊标记，程序可以读取注解，一般用于替代配置文件。
- 开发人员可以通过注解告诉类如何运行。
  - 在Java技术里注解的典型应用是：可以通过反射技术去得到类里面的注解，以决定怎么去运行类。
- 常见注解：@Override、@Deprecated
- 定义注解使用@interface关键字，注解中只能包含属性。

注解

- 注解属性类型：
  - String类型
  - 基本数据类型
  - Class类型
  - 枚举类型
  - 注解类型
  - 以上类型的一维数组

注解

- 元注解:用来描述注解的注解。
- @Retention:用于指定注解可以保留的域。
  - RetentionPolicy.CLASS:
    - 注解记录在class文件中，运行Java程序时，JVM不会保留，此为默认值。
  - RetentionPolicy.RUNTIME:
    - 注解记录在 class文件中，运行Java程序时，JVM会保留，程序可以通过反射获取该注释。
  - RetentionPolicy.SOURCE:
    - 编译时直接丢弃这种策略的注释。
- @Target:
  - 指定注解用于修饰类的哪个成员。

总结

- 类对象：
  - Class对象，封装了一个类的所有信息；程序运行中，可通过Class对象获取类的信息。
- 获取类对象的三种方式：
  - Class c = 对象.getClass();
  - Class c = 类名.class;
  - Class c=Class.forName(“包名.类名”);
- 工厂模式：
  - 主要用于创建对象，通过反射进行工厂模式的设计，完成动态的对象创建。
- 单例模式：
  - Singleton，只允许创建一个该类的对象。
- 枚举类型、注解类型

THANK YOU

做真实的自己，用良心做教育

Author：Aaron
Version：9.0.2