第8章 泛型程序设计

注解：
注解，一种元数据形式，提供有关程序的数据，该数据不属于程序本身。注释对其注释的代码的操作没有直接影响。
注解用途：

- 编译前：为编译器提供编译检查的依据，辅助检查代码错误或抑制检查异常。
- 编译中或发布时：给编译器提供信息生成代码或给其他工具提供信息生成文档等。
- 运行时：在运行过程中提供信息给解释器，辅助程序执行。

注解经常和反射结合起来用，多用于框架中。

反射的功能：

- 在运行时获取任意一个对象所属的类型信息，包括修饰符、泛型、父类、实现的接口、注解等；
- 在运行时构造任意一个类的对象；
- 在运行时获取任意一个类所具有的构造方法、成员变量和方法；
- 在运行时访问任意一个对象的成员变量和方法；

反射的应用：

- 通过使用类全名创建类实例来使用外部用户定义的类。
  例如Java数据库开发中，需要在运行时使用JDBC驱动包中的驱动类，可以通过反射机制在运行中获取。 Class.forName("com.mysql.cj.jdbc.Driver");
- 开发类浏览器和智能IDE。
  例如Eclipse工具，左侧的包浏览器可以查看类的结构，右侧代码编辑区，如果启用了提示功能，在对象后输入“.”运算符，会自动提示该对象所属类的所有可用属性和方法。这些IDE工具的功能需要反射机制实现。
- 在测试工具中用于检测类的内部结构。
  例如Java的单元测试框架Junit就是基于反射和注解实现的
- 在框架开发中用于实现配置信息的处理。
  例如Java Web开发中要学习的Struts2，Spring的框架功能的实现都需要用到反射
- 实现Java的动态代理。

8.9 反射和泛型

反射允许你在运行时分析任意的对象。如果对象是泛型类的实例，关于泛型类型参数则得不到太多信息， 因为它们会被擦除。

8.9.1 泛型 `class` 类

`Class` 类是泛型的。类型参数十分有用， 这是因为它允许 `ClaSS<T>` 方法的返回类型更加具有针对性。

8.9.2 使用 `class` 参数进行类型匹配

匹配泛型方法中的 `Class<I>` 参数的类型变量很有实用价值。

8.9.3 虚拟机中的泛型类型信息

`Java` 泛型的卓越特性之一是在虚拟机中泛型类型的擦除。令人感到奇怪的是， 擦除的类仍然保留一些泛型祖先的微弱记忆。例如， 原始的 `Pair` 类知道源于泛型类 `Pair<T>`， 即使一个 `Pair` 类型的对象无法区分是由 `PaiKString>` 构造的还是由 `PaiKEmployee>` 构造的。