#### ****JAVA集合框架之预览****

#### ****介绍****

#### Java 2（jdk1.2）平台包含一个集合框架。 集合是表示一组对象的对象（如经典的Vector类）。 集合框架是用于表示和操作集合的统一体系结构，允许独立于其表示的细节操纵集合。 **一个集合框架的主要优点(官方直译)是：** https://docs.spring.io/spring-framework/docs/5.0.0.BUILD-SNAPSHOT/spring-framework-reference/html/images/callouts/1.png通过提供有用的数据结构和算法来减少编程工作量，因此您不必亲自编写它们。 https://docs.spring.io/spring-framework/docs/5.0.0.BUILD-SNAPSHOT/spring-framework-reference/html/images/callouts/2.png通过提供有用的数据结构和算法的高性能实现来提高性能。 由于每个接口的各种实现都是可以互换的，程序可以通过切换实现来轻松调整。 https://docs.spring.io/spring-framework/docs/5.0.0.BUILD-SNAPSHOT/spring-framework-reference/html/images/callouts/3.png通过建立通用语言来回传递集合，提供不相关的API之间的互操作性。 https://docs.spring.io/spring-framework/docs/5.0.0.BUILD-SNAPSHOT/spring-framework-reference/html/images/callouts/4.png通过消除学习多个临时收集API的需要，减少了学习API所需的工作量。 https://docs.spring.io/spring-framework/docs/5.0.0.BUILD-SNAPSHOT/spring-framework-reference/html/images/callouts/5.png通过消除生成临时集合API的需求，减少设计和实现API所需的工作量。 https://docs.spring.io/spring-framework/docs/5.0.0.BUILD-SNAPSHOT/spring-framework-reference/html/images/callouts/6.png通过为集合和算法提供标准接口来操纵它们，促进软件重用。 **程序员的大白话如下：** https://docs.spring.io/spring-framework/docs/5.0.0.BUILD-SNAPSHOT/spring-framework-reference/html/images/callouts/1.png降低编程难度 https://docs.spring.io/spring-framework/docs/5.0.0.BUILD-SNAPSHOT/spring-framework-reference/html/images/callouts/2.png提高程序性能 https://docs.spring.io/spring-framework/docs/5.0.0.BUILD-SNAPSHOT/spring-framework-reference/html/images/callouts/3.png提高API间的互操作性 https://docs.spring.io/spring-framework/docs/5.0.0.BUILD-SNAPSHOT/spring-framework-reference/html/images/callouts/4.png降低学习难度 https://docs.spring.io/spring-framework/docs/5.0.0.BUILD-SNAPSHOT/spring-framework-reference/html/images/callouts/5.png降低设计和实现相关API的难度 https://docs.spring.io/spring-framework/docs/5.0.0.BUILD-SNAPSHOT/spring-framework-reference/html/images/callouts/6.png增加程序的重用性 **注意:** Java容器里只能放对象，对于基本类型（int, long, float, double等），需要将其包装成对象类型后（Integer, Long, Float, Double等）才能放到容器里。很多时候拆包装和解包装能够自动完成。这虽然会导致额外的性能和空间开销，但简化了设计和编程。

#### ****集合框架包括：****

集合接口 - 表示集合的不同类型，例如set，list 和map。这些接口构成了框架的基础。  
通用实现 - 集合接口的主要实现。  
传统实现 - 早期版本Vector和Hashtable中的集合类已经过改进以实现集合接口。  
特殊目的实现 - 为在特殊情况下使用而设计的实现。这些实现显示非标准的性能特征，使用限制或行为。  
并发实现 - 为高度并发使用而设计的实现。  
包装器实现 - 为其他实现添加功能，例如同步。  
便利实现 - 集合接口的高性能“微型实现”。  
抽象实现 - 部分实现集合接口以促进自定义实现。  
算法 - 对集合执行有用功能的静态方法，例如排序列表。  
基础架构 - 为集合接口提供重要支持的接口。  
阵列实用程序 - 基元和参考对象数组的实用程序函数。严格来说，并不是Collections Framework的一部分，这个功能同时被添加到Java平台，并依赖于相同的基础架构。

#### ****泛型（Generics）****

java集合框架经常和泛型联系在一起使用，可谓是谁也离不开谁，Java容器能够容纳任何类型的对象，这一点表面上是通过泛型机制完成，Java泛型不是什么神奇的东西，只是编译器为我们提供的一个“语法糖”，泛型本身并不需要Java虚拟机的支持，只需要在编译阶段做一下简单的字符串替换即可。实质上Java的单继承机制才是保证这一特性的根本，因为所有的对象都是Object的子类，容器里只要能够存放Object对象就行了。  
事实上，所有容器的内部存放的都是Object对象，泛型机制只是简化了编程，由编译器自动帮我们完成了强制类型转换而已。JDK 1.4以及之前版本不支持泛型，类型转换需要程序员显式完成。

public static void main(String[] args) {

*//JDK 1.4 or before*

ArrayList list = new ArrayList();

list.add(new String("qzj"));

String qzj = (String) list.get(0);*//显式类型转换*

System.out.println(qzj.toUpperCase());

}

public static void main(String[] args) {

*//JDK 1.5 or latter*

ArrayList<String> list = new ArrayList<String>();*//参数化类型*

*//ArrayList<String> list = new ArrayList();//参数化类型 JDK 1.7 or latter*

list.add(new String("qzj"));

String qzj = list.get(0);*//隐式类型转换，编译器自动完成*

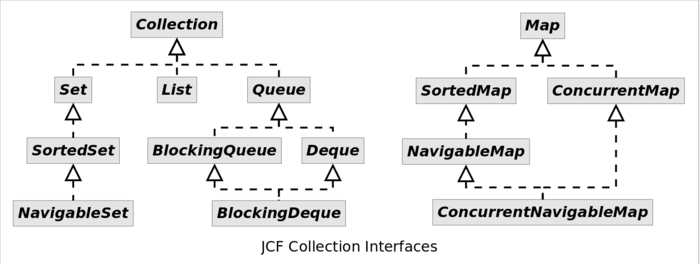
System.out.println(qzj.toUpperCase());

}

#### ****语法糖****

Java语法糖系列，所以首先讲讲什么是语法糖。语法糖是一种几乎每种语言或多或少都提供过的一些方便程序员开发代码的语法，它只是编译器实现的一些小把戏罢了，编译期间以特定的字节码或者特定的方式对这些语法做一些处理，开发者就可以直接方便地使用了。这些语法糖虽然不会提供实质性的功能改进，但是它们或能提高性能、或能提升语法的严谨性、或能减少编码出错的机会。Java提供给了用户大量的语法糖，比如泛型、自动装箱、自动拆箱、foreach循环、变长参数、内部类、枚举类、断言（assert）等

#### ****java集合框架十四大接口****

为了规范容器的行为，统一设计，JCF定义了14种容器接口（collection interfaces），它们的关系如下图所示：  


Map接口没有继承自Collection接口，因为Map表示的是关联式容器而不是集合。但Java为我们提供了从Map转换到Collection的方法，可以方便的将Map切换到集合视图。  
上图中提供了Queue接口，却没有Stack，这是因为Stack的功能已被JDK 1.6引入的Deque取代。

#### ****接口实现****

上述接口的通用实现见下表：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | **Implementations** | | | | |
| **Hash Table** | **Resizable Array** | **Balanced Tree** | **Linked List** | **Hash Table + Linked List** |
| **Interfaces** | **Set** | [HashSet](https://docs.oracle.com/javase/6/docs/api/java/util/HashSet.html) |  | [TreeSet](https://docs.oracle.com/javase/6/docs/api/java/util/TreeSet.html) |  | [LinkedHashSet](https://docs.oracle.com/javase/6/docs/api/java/util/LinkedHashSet.html) |
| **List** |  | [ArrayList](https://docs.oracle.com/javase/6/docs/api/java/util/ArrayList.html) |  | [LinkedList](https://docs.oracle.com/javase/6/docs/api/java/util/LinkedList.html) |  |
| **Deque** |  | [ArrayDeque](https://docs.oracle.com/javase/6/docs/api/java/util/ArrayDeque.html) |  | [LinkedList](https://docs.oracle.com/javase/6/docs/api/java/util/LinkedList.html) |  |
| **Map** | [HashMap](https://docs.oracle.com/javase/6/docs/api/java/util/HashMap.html) |  | [TreeMap](https://docs.oracle.com/javase/6/docs/api/java/util/TreeMap.html) |  | [LinkedHashMap](https://docs.oracle.com/javase/6/docs/api/java/util/LinkedHashMap.html) |

##### ****参考文献****

[Collections Framework Overview](https://docs.oracle.com/javase/6/docs/technotes/guides/collections/overview.html)  
[The For-Each Loop](https://docs.oracle.com/javase/1.5.0/docs/guide/language/foreach.html)

##### ****参考本系列文章****

[Java 集合框架之概览](http://www.spring4all.com/article/1042)  
[java集合框架之ArrayList深度解析（一）](http://www.spring4all.com/article/1045)  
[java集合框架之ArrayList深度解析（二）](http://www.spring4all.com/article/1050)  
[java集合框架之ArrayList深度解析（三）](http://www.spring4all.com/article/1055)  
[java集合框架之ArrayList深度解析（四）](http://www.spring4all.com/article/1056)