Using the Java Collections Framework

Ray Toal

Loyola Marymount University and

Ticketmaster Online-CitySearch, Inc.

April 7, 1999

Outline

- Background (Java, Collections, …)
- What is in the Collections Framework
- Organization of the Framework
- Tour of the Collections Framework
- Concluding Remarks

Goals and Objectives

- To present the overall organization and examples of the use of the Java Collections Framework so that
  - Programmers will be able to start using the framework right away
  - Programmers will be able to get rid of tons of poorly commented, under-tested, *non-standard*, collection classes that defy (large-scale) reuse

What This Talk is About

- What is in the JCF and how the JCF is organized
- Why the JCF looks the way it does, and why it is better than alternatives
- How to write code (right away) using the JCF (via examples)
- Helping you to become a better Java programmer

What This Talk is NOT About

- Introductory Java Programming
- Object Oriented Programming (though the basics are assumed)
- Absolute details of the classes and interfaces (we prefer code samples)
- Language Wars
- Swing

Background

Java

- An enormously popular, buzzword-compliant language for the 1990s and beyond
- Not currently an international standard
- Consists of (1) a base language with the usual keywords, declarations, expressions and statements, and (2) a Core API
- Other languages would call the "Core API" a "standard library"

The Java Core API

- Currently consists of 59 packages (see on-line documentation for complete list)
- Informally, the Core API covers language support, utilities, input/output, graphics, graphical user interfaces, networking, concurrency and distribution, security, database connectivity

The package java.util

- Has the following stuff:
  - **Collections!!**
  - Calendars, Dates, Time zone stuff
  - i18n stuff: Locale, ResourceBundle
  - Event stuff: EventObject, Observable, Observer...
  - Some useful exception classes
  - Really miscellaneous stuff: Random, Permission classes, StringTokenizer

What is a Collection?

- An object that contains other objects
- Four main topologies: Set, Sequence, Hierarchy, Network (some people like to throw in Dictionary)
- Other metrics: allows duplicates? sorted?
- Usually described by interfaces
- Don't confuse data types with data structures

JCF Overview

Simple Example 1

```
import java.util.*;
public class SimpleSetDemo {
    public static void main(String[] args) {
      Set s = new HashSet();
      s.add("odin"); s.add("dva"); s.add("tri");
      Set t = new TreeSet();
      t.add("dva"); t.add("chityri"); t.add("shest");
      s.retainAll(t);
      for (Iterator it = s.iterator(); it.hasNext();)
            System.out.println(it.next());
    }
}
```

Simple Example 2

```
import java.util.*;
public class SimpleListDemo {
    public static void main(String[] args) {
      List s = new LinkedList();
      s.add("odin"); s.add("dva"); s.add(1, "tri");
      List t = new ArrayList();
      t.add("dva"); t.add(0, "chityri"); t.set(1, "shest");
      s.addAll(t);
      for (Iterator it = s.subList(1, 4).iterator(); it.hasNext();)
              System.out.println(it.next());
    }
}
```

Simple Example 3

```
public class ExampleDictionaryProgram {
    public static void main(String[] args) {
      java.util.HashMap spanishWords = new HashMap();
      java.util.HashMap russianWords = new HashMap();
      spanishWords.put("red", "rojo");
      spanishWords.put("brown", "caf\u00e9");
      russianWords.put("black", "\u0447\u0451\u0440\u043d\u044b\u0439");
      russianWords.put("white", "\u0431\u0435\u043b\u044b\u0439");
      javax.swing.JOptionPane.showMessageDialog(null,
              spanishWords.get("red"));
      System.exit(0);
    }
}
```

What's in the JCF?

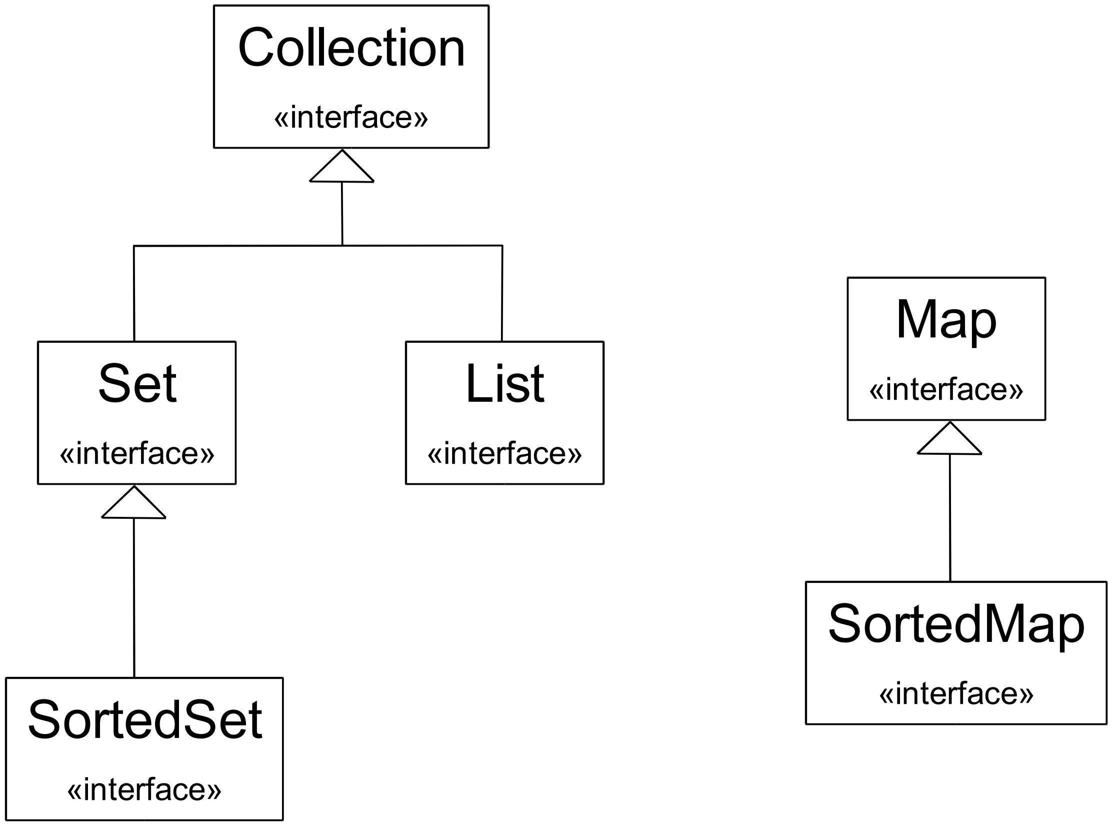
- Collection Interfaces
- Collection Implementations
  - *general purpose / wrapper / abstract / convenience / legacy*
- Algorithms
  - *sort, shuffle, search, min, max, copy, fill, ...*
- Infrastructure
  - *iterators, ordering, exceptions*

Design Goals

- Small in size and conceptual weight
  - don't let number of classes skyrocket by trying to capture subtleties with distinct classes
  - Method in interface IFF it is a basic operation or there's a compelling reason why someone would want to override
- Interoperability among reasonable representations

Tour of the JCF

Collection Interfaces



Collection

```
public interface Collection {
    int size();
    boolean isEmpty();
    boolean contains(Object element);
    boolean add(Object element);
    boolean remove(Object element);
    Iterator iterator();

    boolean containsAll(Collection c);
    boolean addAll(Collection c);
    boolean removeAll(Collection c);
    boolean retainAll(Collection c);
    void clear();

    Object[] toArray();
    Object[] toArray(Object a[]);
}
```

Set

```
public interface Set extends Collection {
    // intentionally empty.
}
```

List

```
public interface List extends Collection {
    Object get(int index);
    Object set(int index, Object element);    // Optional
    void add(int index, Object element);      // Optional
    Object remove(int index);                 // Optional
    boolean addAll(int index, Collection c);  // Optional

    int indexOf(Object o);
    int lastIndexOf(Object o);

    ListIterator listIterator();
    ListIterator listIterator(int index);

    List subList(int from, int to);
}
```

Map

```
public interface Map {
    Object put(Object key, Object value);
    Object get(Object key);
    Object remove(Object key);
    boolean containsKey(Object key);
    boolean containsValue(Object value);

    void putAll(Map t);

    public Set keySet();
    public Collection values();
    public Set entrySet();

    public interface Entry {
        Object getKey();
        Object getValue();
        Object setValue(Object value);
    }
}
```

SortedSet

```
public interface SortedSet extends Set {
    SortedSet subSet(Object fromElement, Object toElement);
    SortedSet headSet(Object toElement);
    SortedSet tailSet(Object fromElement);

    Object first();
    Object last();

    Comparator comparator();
}
```

SortedMap

```
public interface SortedMap extends Map {
    SortedMap subMap(Object fromKey, Object toKey);
    SortedMap headMap(Object toKey);
    SortedMap tailMap(Object fromKey);

    Object first();
    Object last();

    Comparator comparator();
}
```

General Purpose Implementations

Interfaces	Implementations			
	Hash	Resizable	Balanced	Linked
	Table	Array	Tree	List
Set	HashSet		TreeSet	
List		ArrayList		LinkedList
Map	HashMap		TreeMap	

Wrapper Implementations

```
public class Collections {
    ...
    public static Collection synchronizedCollection(Collection c)
    public static List synchronizedList(List list)
    public static Map synchronizedMap(Map m)
    public static Set synchronizedSet(Set s)
    public static SortedMap synchronizedSortedMap(SortedMap m)
    public static SortedSet synchronizedSortedSet(SortedSet s)
    public static Collection unmodifiableCollection(Collection c)
    public static List unmodifiableList(List list)
    public static Map unmodifiableMap(Map m)
    public static Set unmodifiableSet(Set s)
    public static SortedMap unmodifiableSortedMap(SortedMap m)
    public static SortedSet unmodifiableSortedSet(SortedSet s)
    ...
}
```

Convenience Implementations

```
public class Collections {
    ...
    // All of the following are immutable
    public static List EMPTY_LIST = ...
    public static Set EMPTY_SET = ...
    public static Set singleton(Object o) {...}
    public static List nCopies(int n, Object o) {...}
    ...
}
```

Some Algorithms

```
public class Collections {
    ...
    // some of these have versions taking Comparators too
    public static int binarySearch(List list, Object key) {…}
    public static void copy(List dest, List src) {…}
    public static void fill(List list, Object o) {…}
    public static Object max(Collection coll) {…}
    public static Object min(Collection coll) {…}
    public static void reverse(List list) {…}
    public static void shuffle(List list) {…}
    public static void shuffle(List list, Random rnd) {…}
    public static void sort(List list) {…}
    ...
}
```

Iterators

```
public interface Iterator {
    boolean hasNext();
    Object next();
    void remove();
}

public interface ListIterator extends Iterator {
 void add(Object o)
 int nextIndex()
 boolean hasPrevious()
 Object previous()
 int previousIndex()
 void set(Object o)
}
```

Concluding Remarks

Advice

- Use the new collections in all your new work; port old code if feasible
- Know something about performance
- Compose your own quick-reference guide to the JCF
- Tell your friends who still use Vector and Hashtable to get a life :-)

How to get good at this

- Read the on-line *Collections Framework Overview*, the on-line *Annotated Outline of the Collections Framework*, and the *Collections Framework FAQ*
- Do the Collections trail on the on-line Java tutorial
- Read the on-line API reference for java.util
- Write tons of code