CS202 Java Object Oriented Programming

Introduction to Collection Classes

Chengyu Sun
California State University, Los Angeles

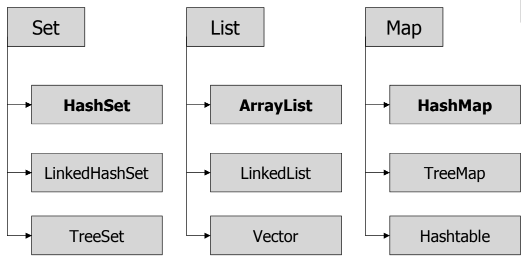
Arrays Are Not Enough

- The Good
  - Easy to use
  - Space efficient
  - Constant time to access an array element
- The Bad
  - Cannot dynamically add or remove elements

Java Collection Framework

- In the java.util package
  - http://java.sun.com/j2se/1.5.0/docs/api/java/util/package-summary.html
- Include a number of interfaces, classes, and algorithms

Collection Framework For Dummies



List, Set, and Map

- `List`
  - When the elements need to be ordered
  - Can be used as a dynamic array
  - Allow duplicates
- `Set`
  - When the elements do not need to be ordered
  - Do not allow duplicates
- `Map`
  - When the elements are <key,value> pairs
  - Associative array

Examples

- CollectionTest.java
  - add(), remove(), get(), clear()
  - contains(), size(),

Iterate Through All Elements in A Collection

- `Iterator`
  - hasNext()
  - next()
- `for` loop (Java 1.5)

Benefits and Limitations of Collection Classes

- Benefits
  - Resizable
  - Elements can be of any class type
- Limitations
  - Cannot hold values of primitive types
  - Element access via an `Object` reference
  - *Both of these limitations have been addressed in Java 1.5*

Topics Related to Collection Classes

- Wrapper classes
- Auto Boxing/Unboxing (Java 1.5)
- Generics (Java 1.5)

Wrapper Classes

- For each primitive type there's a corresponding class
  - boolean – Boolean
  - int – Integer
  - char – Character
  - double – Double
  - …
- Provide some utility functions for a certain primitive type

Integer

- Constants
  - `MAX_VALUE` and `MIN_VALUE`
- Methods for conversions among different types

$$\begin{tikzcd}
\text{int} \arrow[r, shift left, "\text{Constructor}"] \arrow[rr, bend right=40, "??"'] & \text{Integer} \arrow[l, shift left, "\text{intValue}"] \arrow[r, shift left, "\substack{\text{toString}\\ \text{toBinaryString}\\ \text{toHexString}\\ \text{toOctalString}}"] & \text{String} \arrow[l, shift left, "\substack{\text{Constructor}\\ \text{valueOf}}"] \arrow[ll, bend left=30, "\text{parseInt}"]
\end{tikzcd}$$

Integer Example

```
Integer a = new Integer(10);             // int to Integer

int d = a.intValue();                    // Integer to int
int c = Integer.parseInt( "1234");       // String to int

String bin = Integer.toBinaryString(a);  // Integer to String
String hex = Integer.toHexString(a);     // Integer to String
String oct = Integer.toOctalString(a);   // Integer to String

Integer b = new Integer("10");           // String to Integer
Integer n = Integer.valueOf( "101" );    // String to Integer
Integer m = Integer.valueOf( "101", 2 );// String to Integer
```

Lab Assignment Revisited

- Given an integer number consists of only 0’s and 1’s, get its binary, octal, and hexadecimal values

```
Scanner in = new Scanner(System.in);
int a = in.nextInt();

String s = “” + a; // int to String

int v2 = Integer.valueOf(s, 2).intValue();
int v8 = Integer.valueOf(s, 8).intValue();
int v16 = Integer.valueOf(s,16).intValue();
```

Auto Boxing/Unboxing

- A Java 1.5 feature
- Automatically convert between a primitive type and its wrapper class type

```
int a = 10;
Integer b = a;
Integer c = new Integer(20);

int d = b + c;
```

Generics

- A Java 1.5 feature
- Specify the element type of a collection

```
ArrayList<String> list1 = new ArrayList<String>();
ArrayList<Integer> list2 = new ArrayList<Integer>();

list1.add( "100" );
list2.add( 100 );
list1.add( 200 ); // error!
list2.add( "200" ); // error!
```