

Objectives

- Javadoc feedback
- Collections
 - Generics
 - Lists
 - Sets

Oct 7, 2011

Sprenkle - CSCI209

1

Writing Javadoc Comments

- Formatting of Javadocs
 - Tags always go last
- Hints for others using your code
- See examples in Java's API

Oct 7, 2011

Sprenkle - CSCI209

2

MediaItem toString Method

```
public String toString() {
    String classname = getClass().toString();
    StringBuilder rep = new StringBuilder(classname);
    ...
}
```

- Addresses some issues some of you had...

Can you guess what is happening?
What issues does this solve?
Why does it work?

Oct 7, 2011

Sprenkle - CSCI209

3

Naming Variables

- A variable to track if that item is currently present in the collection
- Good names? Good methods?

Oct 7, 2011

Sprenkle - CSCI209

4

Using booleans in if statements

```
if( inCollection == true ) {
    System.out.println("In collection");
}
```

```
if( inCollection ) {
    System.out.println("In collection");
}
```

Oct 7, 2011

Sprenkle - CSCI209

5

COLLECTIONS

Oct 7, 2011

Sprenkle - CSCI209

6

Collections

- Sometimes called *containers*
- Group multiple elements into a single unit
- Store, retrieve, manipulate, and communicate aggregate data
- Represent data items that form a natural group
 - Poker hand (a collection of cards)
 - Mail folder (a collection of messages)
 - Telephone directory (a mapping of names to phone numbers).

Oct 7, 2011

Sprenkle - CSCI209

7

Collections Framework

- *Unified architecture* for representing and manipulating collections
- More than arrays
 - More flexible, functionality, dynamic sizing
- `java.util`

Oct 7, 2011

Sprenkle - CSCI209

8

Collections Framework

- **Interfaces**
 - Abstract data types that represent collections
 - Collections can be manipulated *independently* of implementation
- **Implementations**
 - Concrete implementations of collection interfaces
 - Reusable data structures
- **Algorithms**
 - Methods that perform useful computations on collections, e.g., searching and sorting
 - Reusable functionality
 - **Polymorphic**: same method can be used on many different implementations of collection interface

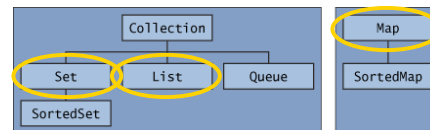
Oct 7, 2011

Sprenkle - CSCI209

9

Core Collection Interfaces

- Encapsulate different types of collections



Oct 7, 2011

Sprenkle - CSCI209

10

GENERICS

Oct 7, 2011

Sprenkle - CSCI209

11

Example of the Way It Was

- Before Java 1.5
- Doesn't know what *type* of data is in the List

```

List myIntList = new LinkedList();
myIntList.add(new Integer(0));
...
Integer x = (Integer) myIntList.get(0);
  
```

Returns an Object

- Have to cast object we get out of list to desired type
- What if someone put in an object of wrong type previously?
- Have similar issue in Python

Oct 7, 2011

Sprenkle - CSCI209

12

Generic Collection Interfaces

- Added to 1.5
- Declaration of the Collection interface:


```
public interface Collection<E>{ ... }
```

Type parameter

 - > <E> means interface is generic for element class
- When declare a Collection, **specify type** of object it contains
 - > Make sure put in, get out appropriate type
 - > Allows compiler to verify that object's type is correct
 - Reduces errors at runtime
- Example, a hand of cards: Always declare type

```
List<Card> hand = new ArrayList<Card>();
```

Oct 7, 2011

Sprenkle - CSCI209

13

Comparable Interface

- Also uses Generics

```
public interface Comparable<T>

int compareTo(T o)
```

The type it compares

You may have seen the warning in assignment 6

Oct 7, 2011

Sprenkle - CSCI209

14

Types Allowed with Generics

- Can only contain Objects, not primitive types
 - > Autoboxing and Autounboxing to the rescue!
 - Example: If collecting `ints`, use `Integer`

Oct 7, 2011

Sprenkle - CSCI209

15

Comparing: Before & After Generics

- Before Generics

```
List myIntList = new LinkedList();
myIntList.add(new Integer(0));
...
Integer x = (Integer) myIntList.get(0);
```

- After Generics

```
List<Integer> myIntList = new LinkedList<Integer>();
myIntList.add(new Integer(0));
...
Integer x = myIntList.get(0);
```

✓ Improved readability and robustness

Oct 7, 2011

Sprenkle - CSCI209

16

LISTS

Oct 7, 2011

Sprenkle - CSCI209

17

List

- An *ordered* collection of elements
- Can contain duplicate elements
- Has control over where objects are stored in the list

Oct 7, 2011

Sprenkle - CSCI209

18

List Interface

- **boolean** `add(<E> o)`
 - Boolean so that List can refuse some elements
 - e.g., refuse adding **null** elements
- **<E>** `get(int index)`
 - Returns element at the position index
 - Different from Python: no shorthand
 - Can't write `list[pos]`
- **int** `size()`
 - Returns the number of elements in the list
- And more!
 - `contains`, `remove`, `toArray`, ...

Oct 7, 2011

Sprenkle - CSCI209

19

Common List Implementations

- **ArrayList**
 - Resizable array
 - Used most frequently
 - Fast
- **LinkedList**
 - Use if adding elements to ends of list
 - Use if often delete from middle of list
 - Implements Deque and other methods so that it can be used as a stack or queue

How would you find the other implementations of List?

Oct 7, 2011

Sprenkle - CSCI209

20

Implementation vs. Interface

Implementation choice only affects performance

- Preferred Style:
 1. Choose an implementation
 2. Assign collection to variable of corresponding **interface** type
- Methods should accept interfaces—not implementations

```
Interface variable = new Implementation();
```

Why is this the preferred style?

Oct 7, 2011

Sprenkle - CSCI209

21

Implementation vs. Interface

Implementation choice only affects performance

- Preferred Style:
 1. Choose an implementation
 2. Assign collection to variable of corresponding **interface** type
- Why?
 - Program does not depend on a given implementation's methods
 - Access only using interface's methods
 - Programmer can change implementations
 - Performance concerns or behavioral details

Oct 7, 2011

Sprenkle - CSCI209

22

Discussion of Deck Class

`cards.Deck.java`

Oct 7, 2011

Sprenkle - CSCI209

23

SETS

Oct 7, 2011

Sprenkle - CSCI209

24

Set Interface

- No duplicate elements
 - Needs to determine if two elements are "logically" the same (equals method)
- Models mathematical set abstraction

Oct 7, 2011

Sprenkle - CSCI209

25

Set Interface


- **boolean** add(**<E>** o)
 - Add to set, only if not already present
- **int** size()
 - Returns the number of elements in the list
- And more! (contains, remove, toArray, ...)
 - Note: no get method -- get #3 from the set?

Oct 7, 2011

Sprenkle - CSCI209

26

Some Set Implementations

- | | |
|---|--|
| <ul style="list-style-type: none"> • HashSet  ➢ Implements set using <i>hash table</i> <ul style="list-style-type: none"> • add, remove, and contains each execute in O(1) time ➢ Used more frequently ➢ Faster than TreeSet ➢ No ordering | <ul style="list-style-type: none"> • TreeSet ➢ Implements set using a <i>tree</i> <ul style="list-style-type: none"> • add, remove, and contains each execute in O(log n) time ➢ Sorts |
|---|--|

Oct 7, 2011

Sprenkle - CSCI209

27

FindDuplicates Problem

- From the array of command-line arguments, identify the duplicates

```
public static void main(String args[]) {
}
```

Oct 7, 2011

Sprenkle - CSCI209

28

FindDuplicates

```
public static void main(String args[]) {
    Set<String> s = new HashSet<String>();
    for (String a : args) {
        if (!s.add(a)) {
            System.out.println(
                "Duplicate detected: " + a);
        }
    }
    System.out.println(s.size() +
        " distinct words detected: " + s);
}
```

How much does code change if s is a TreeSet?

Oct 7, 2011

Sprenkle - CSCI209

29

TODO

- Assignment 8: Due Wednesday
 - **Modifying MediaItem classes**
 - Adding type for Comparable (Generics)
 - Using some Collection (of your choice) to maintain library
 - Justify/explain your choice
 - Adding a user interface

Oct 7, 2011

Sprenkle - CSCI209

30