Java Interfaces, Polymorphism

Lecture 29

Borrowing from slides by Alan Hu, Kurt Eiselt, Paul Carter, and Tamara Munzner

News

- Midterm 2 is marked
  - Grades will be online by Friday
- Homework 3 is now online
  - Due December 4

Reading Assignments

- Reading for this week: class design
  - Edition 3: Ch 8.1-8.9
  - Edition 2: Ch 9.1-9.9
- Reading for this/next week: interfaces
  - Edition 2: Ch 11.1-11.3

Recap: Word Count

- Final implementation of WordCount
  - Use Bubble Sort to sort array of words
  - Use Binary Search for ONE copy of the word
  - Then go backwards and forwards from that point to count all instances

Aside – Order of Termination Conditions

- We had this code in the

```java
// search upward (higher index)
for( int i= index+1 ;
i< numWords &
  searchWord.equals( words[i] ) ;
i++ )
counter++;
```
- Does the order of the two conditions in the logical expression matter?

Reminder: Short-Circuiting Logical Operators

- Consider expression

```java
if ((b > a) && (c == 10))
System.out.println("this should print");
```
- Java evaluates left to right
  - if (b>a) is false, does value of (c == 10) matter?
  - No – result of && will be false, if only one operand is
  - short-circuiting: Java does not evaluate
    - aka lazy evaluation
  - So Java does not evaluate (c == 10)
Recap: Complexity of Algorithms
- What is the performance of algorithms?
- How many operations are required for a problem of a given size?
  - How many comparisons are needed for finding an object in an array?
  - How many comparisons are required for sorting an array?
  - How many floating point operations are required for multiplying two matrices?
- ...

Recap: Complexity of Sorting & Searching Algorithms
- How many comparisons are required to find an object in an array with n elements?
  - Linear search: $O(n)$ – average, worst case
  - Binary search: $O(\log(n))$ – avg, worst case
- How many comparisons are needed to sort an array with n elements?
  - Bubble Sort: $O(n^2)$ – avg, worst case
  - Quick Sort: $O(n \log(n))$ - avg; $O(n^2)$ – worst

Recap: Complexity – Does it Matter?
- Computers are fast, right?
- But: for large datasets, the complexity really makes a difference!
  - E.g: array with 1 million entries
    - Linear Search: on avg 500,000 comparisons
    - Binary Search: on avg 30 comparisons!
    - Bubble Sort: on avg 500 Billion comparisons
    - Quick Sort: on avg 30 Million comparisons!
  - In both cases, the improvement is a factor of 16,667!

Objectives
- Understand the concept of interfaces.
- Learn how to use pre-defined interfaces to write reusable code.

Motivation
- How can we make our sorting and binary search algorithms work for other data types?
  - What is required of a data type in order to be sortable?
  - What is required for binary search?

Motivation
- How can we make our sorting algorithm work for other data types?
  - What is required of a data type in order to be sortable?
- Sorting & searching requires that data is comparable
  - E.g. Strings: compareTo method
  - All data types that have this method are said to share and interface
Interfaces in Real Life

- What does it mean
- When a product says it’s “USB compatible”?
- When a gas station sells “regular unleaded (87 octane) gas”?
- When you buy a CD that says “Compact Disc Digital Audio”?

The producer promises that the product has certain features and behaviors.
If the user uses only those features and behaviors, then everything should work right.

Interfaces as Contracts

- If you buy a “USB” product, but the plug is shaped wrong, who is responsible?
- If you put 87 octane gas in a car that runs on diesel, who is responsible?
- If you buy a CD, but it actually installs secret spyware on your computer, who is responsible?
- Producer promises to supply certain features
- Consumer promises to use only those features.

Interfaces Save Effort

- You can buy any USB devices and plug them in, as long as you obey the USB standard.
- You can buy regular gas from any gas station, as long as your car doesn’t demand some other fuel.
- You can buy a CD from any manufacturer, and it will play properly.
- Imagine if you had to negotiate the details of every purchase!

Interfaces in Java Save Effort, Too

- Java has a similar concept of interfaces.
- A class can be declared to implement an interface.
  - This is a promise by the writer of the class that it has certain public methods available that behave a certain way.
  - Each class usually has additional methods
  - Code can declare variables with the interface name (not a class name).
  - Java will let you use only the interface methods.
  - But now, your code will run with any class that implements that interface!

Example: the Comparable interface

- Java provides an interface called Comparable.
  - Think of this like the name of a class.
  - Objects that implement Comparable must provide a compareTo() method:
    - Returns an int < 0 if this < argument
    - Returns 0 if this == argument
    - Returns an int > 0 if this > argument
- Many classes implement Comparable: String, Integer, Double, etc.
Example: Sorting an Array
- Using Comparable, we can write the same code, that sorts different kinds of arrays!

Implementing an Interface
- Define a class as usual, but claim that it implements the interface, e.g.:
  ```java
  public class WordCount implements Comparable {
    ...
  }
  ```
- Define all methods needed for interface, e.g.:
  ```java
  public int compareTo(WordCount x) {
    ...
  }
  ```

Example: Sorting an Array
- We'll see next time how to design our own interfaces.