Visitors, Visitors, Visitors …

Corky Cartwright
Department of Computer Science
Rice University

Review: the Visitor Pattern

- Externalizes the *interpreter* pattern for defining operations on *composite* pattern hierarchies.
- In principle, all recursively defined operations on a composite hierarchy can be defined using visitors. Is this a good idea?
- Probably not. The visitor pattern adds modest additional computational overhead (more method calls, notation) which may not be entirely eliminated by the JIT compiler. It also adds notational overhead (`.accept(new Method Visitor(...))` instead of `.method(...)`). Simple, intuitively intrinsic operations should probably be defined using the interpreter pattern. When designing a composite hierarchy, define a set of simple basic operations using the interpreter pattern and define everything else using visitors. It is a judgment call.

Example: Functional Lists

- Primitive operations:

```
int length();
boolean contains();
List concat(List other);
List reverse();
List eltAt(int i);
List subList(int i, int len);
ListIterator iterator();

List sort();
List merge(List other);
List sort(Comparator c);
List merge(Comparator c, List other);
```

Road Map for Remainder of Course

- Friday: full Java for our Intermediate Level subset. What extra code do we have to write in full Java? Constructors, accessors, `toString()`, `equals(…)`, `hashCode()` , visibility modifiers, `final` modifier.
- Monday: simple generic types. Complex generics are beyond the capacity even of the Java language designers. See …
- Wednesday: more Java mechanics for functional code:
  - Static members of classes
  - Checked vs. unchecked exceptions
- Friday: imperative Java including mutation, arrays/vectors, loops.

Road Map for Remainder of Course

- Friday: full Java for our Intermediate Level subset. What extra code do we have to write in full Java? Constructors, accessors, `toString()`, `equals(…)`, `hashCode()` , visibility modifiers, `final` modifier.

Next Week:

- Monday: simple generic types. Complex generics are beyond the capacity even of the Java language designers. See TechRepublic blurb
- Wednesday: more Java mechanics for functional code:
  - Static members of classes
  - Checked vs. unchecked exceptions
  - Nested and inner classes.
- Friday: imperative Java including mutation, arrays/vectors, loops.

Following Weeks:

- Mutable Linear Data Structures and Trees
- Sorting and Searching including Hashing and Search Trees
- Memoization and Dynamic Programming
- Concurrency and Event Driven Programming

For Next Class

- Homework Due.
- Please report problems with DrJava Language Levels.