CMSC433, Spring 2001 Programming Language Technology and Paradigms Basic Java, continued

Alan Sussman

October 9, 2001

Administrivia

- Midterm Thursday
  - on everything covered up through today
- Project 2 questions?
  - problems with parsing arguments?
- Project 1 commentary due tomorrow (10/10) at 6PM
  - submit with **submit 11 1.txt 2.txt**
  - final drivers used for grading posted
- Project 2 out for commentary soon
  - due 10/24
- Project 3 out by tomorrow - threads
  - due dates will change!

Last time - Java

- I/O
  - usually best to turn everything into *PrintWriter* for output and *BufferedReader* for input
- Packages
  - like C++ namespaces
  - use **import** to allow short names
- Wrapper classes allow using primitive types as classes, and provide utility functions
- Exceptions
  - just classes with special semantics
  - methods must declare all exceptions they might throw
  - all exceptions eventually get caught

Inner Classes

- Allow a class to be defined within a class or method
- New class has access to all variables in scope
- Classes can be anonymous
- 4 kinds of inner classes
  - nested classes/interfaces
  - standard inner classes
  - method classes and anonymous classes
- Lots of important details

Nested classes/interfaces

- Not really an inner class
  - not associated with an instance of the outer class
- Defined like a static class method/variable
- Can refer to all static methods/variables of outer class, transparently
- Used to localize/encapsulate classes only used by the outer class
  - information hiding/packaging
- Used to package helper classes/interfaces
  - like a mini-package for each class

Example

```
public class LinkedList {
  // Keep this private; no one else see the implementation
  private static class Node {
    Object value;  Node next;
    Node(Object v)  { value = v;  next = null;  }  };
  // Put here to show that this is the Transformer for LinkedList
  public static interface Transformer {public Object transform(Object v); }
  Node head, tail;
  public void applyTransformer(Transformer t) {
    for (Node n = head; n != null; n = n.next)
      n.value = t.transform(n.value);
  }
  public void append(Object v) {
    Node n = new Node(v);
    if (tail == null) head = n;
    else tail.next = n;
    tail = n;  }  }
```

```
public class getStringRep
    implements LinkedList.Transformer {
  public Object transform(Object o) {
    return o.toString(); }
```

Standard Inner Classes

- Defined like a class method/variable
- Each instance associated with an instance of the outer class
- If class **A** is outer class
  - use **A.this** to get **this** for instance of outer class
- Can refer to all methods/variables of outer class
  - transparently
- Can't have any static methods/variables

Example

```
public class FixedStack {
  Object [] array;
  int top = 0;
  class MyEnum implements java.util.Enumerator {
     int count = top;
     public boolean hasMoreElements() { return count > 0; }
     public Object nextElement() {
          if (count == 0)
            throw new NoSuchElementException("FixedStack");
          return array[--count];  }
  }
  public java.util.Enumerator enumerateAll() {
          return new MyEnum();  }
}
```


Method and Anonymous Classes

- Can refer to all methods/variables of outer class
- Can refer to **final** local variables
- Can't have any static methods/variables
- Method classes defined like a method variable
- Anonymous classes defined in new expression
  - **new BaseClassOrInterface() {** *extensions* **}**

Method class Example

```
public class FixedStack {
  Object [] array;
  int top = 0;
  public java.util.Enumerator enumerateOldestK(final int k) {
     class MyEnum implements java.util.Enumerator {
       int pos = 0;
       public boolean hasMoreElements() {
         return pos < k && pos < top; }
       public Object nextElement() {
         if (!hasMoreElements()) {
           throw new NoSuchElementException("FixedStack");
         return array[pos++];  }
     }
     return new MyEnum(); }
}
```


Anonymous class Example

```
public class FixedStack {
  Object [] array;
  int top = 0;
  public java.util.Enumerator enumerateOldestK(final int k) {
    return   new java.util.Enumerator() {
      int pos = 0;
      public boolean hasMoreElements()
           { return pos < k && pos < top; }
      public Object nextElement() {
           if (!hasMoreElements())
             throw new NoSuchElementException("FixedStack");
           return array[pos++]; }
    }
} }
```


Important details

- If class **B** is defined inside of class **A**
  - a synchronized method of **B** locks **B.this**, not **A.this**
  - may want to lock **A.this** for synchronization
  - can have many **B**'s for each **A**
- Can't define constructor for anonymous inner class
- Inner classes are a compile-time transformation
  - separate class file generated for each inner class
  - have $'s in names