CMSC433, Spring 2002 Programming Language Technology and Paradigms Java Threads

Alan Sussman
February 19, 2002

Administrivia

- Project 2 due next Wednesday, Feb. 27
  - questions?
- Project 1 commentary
  - should have 2 programs to comment on by now
- Next time
  - postmortem of project 1

Last time - Java

- Wrapper classes, String and StringBuffer
  - String is immutable, StringBuffer isn't
- Exceptions
  - shouldn't ignore Errors, sometimes OK to ignore Exceptions
  - Exceptions are objects, subclass of Exception
  - must declare user-defined Exceptions that can be thrown by a method (part of return type)
  - finally block always executed
- Inner classes
  - nested classes/interfaces – behave like static method/instance variable
    - can only access statics of enclosing class

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

I/O and Utility Libraries

I/O Classes

- **File**
  - directories
    - if(f.isDirectory()) System.out.println(f.list());
  - interface **FilenameFilter** – allows selection of sublist
- **OutputStream** – byte stream going out
- **Writer** – character stream going out
- **InputStream** – byte stream coming in
- **Reader** – character stream coming in

OutputStream - bytes

- base types
  - **ByteArrayOutputStream**
  - **FileOutputStream** – goes to file
  - **PipedOutputStream** – goes to **PipedInputStream**
  - **SocketOutputStream** (not public) – goes to TCP socket
- Filters – wrapped around an **OutputStream**
  - **BufferedOutputStream**
  - **ObjectOutputStream** (should implement **FilterOutputStream**) – serialization of object graph

Writer - characters

- **OutputStreamWriter**
  - wraps around **OutputStream** to get a **Writer**
  - takes characters, converts to bytes
  - can specify encoding used to convert
- **CharArrayWriter**
- **StringWriter**
- **Filters**
  - **PrintWriter** – supports **print**, **println**
  - **BufferedWriter**
- Convenience writers
  - wrap **OutputStreamWriter** around an **OutputStream**
  - **FileWriter** and **PipedWriter**

InputStream - bytes

- base types
  - **ByteArrayInputStream**
  - **FileInputStream**
  - **PipedInputStream**
  - **SocketInputStream**(not public) – comes from TCP socket
- Filters – wrapped around **InputStream**
  - **BufferedInputStream**
  - **PushedBackInputStream**
  - **ObjectInputStream**
- **SequenceInputStream** - concatenate

Reader - characters

- **InputStreamReader**
  - wrap around **InputStream** to get a **Reader**
  - takes bytes, converts to characters
  - can specify encoding used to convert
- **CharArrayReader**
- **StringReader**
- **Filters**
  - **BufferedReader** – efficient, supports **readLine()**
    - **LineNumberReader** – reports line numbers
  - **PushBackReader**
- Convenience Readers
  - wrap **InputStreamReader** around **InputStream**
  - **FileReader** and **PipedReader**

java.util

- Vector
- Dictionaries
- Enumerations and Bitsets
- Collection classes

Vector

- A list/vector abstraction
- Can insert/delete/modify elements anywhere in the list
- Can access by position
- Stack
  - extension of **Vector**
  - adds push, pop, peek and empty

Maps

- **Map**
  - an interface
  - that represents key to value mapping
- **HashMap**
  - implements Map interface
  - grows as needed
  - can be saved to a file (serializable, implements deep toString)
- interface **SortedMap**
  - class **TreeMap**

Enumerations and Bitsets

- Enumeration
  - an interface
  - used in many places to return an enumeration
    - **public boolean hasMoreElements()**
    - **public Object nextElement()**
- BitSet
  - provides representation of a set as a bit vector
  - grows as needed (like HashMap)

Collection Classes

- interface **Collection**
  - interface **List**
    - class **Vector** (and **Stack**)
    - class **ArrayList**
    - class **LinkedList** – doubly linked
  - interface **Set**
    - class **HashSet**
    - interface **SortedSet**
      - class **TreeSet**

Other libraries

- java.lang.Math
  - abstract final class – only static members
  - includes constants *e* and π
  - includes static methods for trig, exponentiation, min, max, …
- java.text
  - text formatting tools
    - class **MessageFormat** provides printf/scanf functionality
  - lots of facilities for internationalization

Multithreading and Synchronization

What is a thread?

- It's a program counter and a stack
- All threads share the same memory space
  - take turns with the CPU, for a uni-processor
  - run concurrently, on a multiprocessor
- Example: Web browser
  - one thread for I/O
  - one thread for each file being downloaded
  - one thread to render web page
- Running thread might
  - yield, sleep, wait for I/O or **notify**, be pre-empted

Writing Multi-threaded Code

- Need to control which events can happen simultaneously
  - e.g., update and display methods for a class
- Usually only covered in OS/DB courses
  - so few programmers have lots of training
- Can get inconsistent results or deadlock
  - problems often not easily reproduced
- Easy to get multi-threading, without trying
  - Graphical User Interfaces (GUI's)
  - Remote Method Invocation

Extending class Thread

- Can build a thread class by extending **java.lang.Thread**
- Must supply a **public void run()** method
- Start a thread by invoking the **start()** method
- When a thread starts, executes **run**()
- When **run()** returns, thread is finished/dead

Simple thread methods

- **void start()**
- **boolean isAlive()**
- **void setDaemon(boolean on)**
  - if only daemon threads running, VM terminates
- **void setPriority(int newPriority)**
  - thread scheduler might respect priority
- **void join() throws InterruptedException**
  - waits for a thread to die/finish

Simple static thread methods

- Apply to thread invoking the method
  - **void yield()**
  - **void sleep(long milliseconds) throws InterruptedException**
  - **Thread currentThread()**