SE 552: Lecture 1

Overview

Course summary

Administrivia

Introduction to concurrent programming

Diagrams

Example

Summary

Course summary

Fundamentals and techniques of developing concurrent object-oriented applications, using a patterns-based approach.

Concepts covered include: threads, synchronization and object locking, thread blocking and deadlock, safety and liveness, state-dependent action and concurrency control.

Course structure

Weeks 1-2. *Technology*: an introduction to the Java Threads API.

Weeks 3-5. *Exclusion*: object locking and ownership of resources.

Weeks 7-8. *State Dependence*: dealing with failure and blocking.

Weeks 9-10. *Creating Threads*: managing thread objects.

Aims

Understand that concurrency is hard!

Learn techniques for developing concurrent software.

See common pitfalls in concurrent software, and how to avoid them.

Use patterns and OO to formalize solutions.

Objectives

See how to program in a patterns-based style in Java.

Learn the Java threads package for concurrency.

Develop thread-safe packages for concurrent programming.

Administrivia: contact details

Lecturer: Alan Jeffrey

Email: ajeffrey@cs.depaul.edu

Office: CST 840

Phone: (312) 362 8322

Office hours: 3.30-5.00pm Tuesdays and Thursdays

Administrivia: reading materials

Course home page: http://fpl.cs.depaul.edu/ajeffrey/se552/, contains lectures, homeworks, pointers to API documentation, sample source code...

Textbook: Concurrent Programming in Java, 2nd edition, by Doug Lea, Addison Wesley, 2000.

Administrivia: prerequisites

Required:

SE 450: Object-Oriented Software Development

CSC 416: Foundations of Computer Science II

Wouldn't hurt:

CSC 343: Introduction to Operating Systems

SE 430: Object-Oriented Modeling

SE 550: Distributed Software Development

CSC 599: Concurrent System Design

Administrivia: required software

A Java 1.4 or 1.5 compiler, e.g. Sun's JDK.

An editor for Java source, e.g. XEmacs.

Pointers to software are on the course home page.

Administrivia: assessment

Mid-term exam (4 May 2004): 33%

Final exam (8 June 2004): 33%

Homeworks due in weeks 3, 5, 8 and 10 (submitted using Courses Online, best 3 out of 4): 33%

All students must attend the mid-term and final exams.

In-class students must take the in-class exam. DL students must register on-line for an exam (you may select to take the in-class exam).

Late assignments will not be accepted without medical evidence.

Plagiarism or collusion is unacceptable, and will earn an F in the course.

Introduction to concurrent programming

What is concurrent programming?

What is the difference between concurrent programming, parallel programming, and distributed programming?

What is the difference between a host, a processor, and a thread?

What is the difference between a thread and an object?

Why concurrent programming? (For example, MacOS was single-threaded up until MacOS 8; Palm OS 5 has only one user thread.)

Why object oriented programming?

Why Java?

Diagrams

What is a sequence diagram (called an interaction diagram by Lea)?
A message? A return? An asynchronous message?

What is a class diagram?
What is a superclass ('is a')? An attribute ('has a')?
What is the cardinality of an attribute (0, 1, *...)?
What is the difference between an interface and a class?

What is an object diagram?

We will be using UML in this course, for example UML Distilled is a good concise book.

Example: the Jack application

Download the src.zip archive and unpack it.

Compile and run the Jack application with:

```
cd src
javac ajeffrey/teaching/jack/Main.java
java ajeffrey.teaching.jack.Main
```

Gotchas:

- Make sure you're running JDK 1.2 or better.
- Make sure you're in the right directory, and that your CLASSPATH is set correctly.
- You may need to use backslashes "\" rather than forward slashes "/"

How would we design such an application?

Example: the Jack application

'User-centered design': design the GUI first.

Then ask 'What objects do we need for this application?' and 'What methods do those objects need?'.

Example: the Jack application

Here's the main source files:

- Main.java: the main application class.
- GUI.java: the view.
- Logic.java: the model.

The question is: how to implement the model?

Example: the Jack application

Attempt 1: SimpleLogic.java

What's wrong with this implementation? (Hint: try pressing the New button twice.)

Example: the Jack application

Attempt 2: PollingLogic.java

What's wrong with this implementation? (Hint: what's polling?)

Example: the Jack application

Attempt 3: SuspendResumeLogic.java

What's wrong with this implementation? (Hint: try compiling it.)

Example: the Jack application

Final version: GuardedLogic.java

Uses Guard.java

What does the Guard class do? How does it do it?

Summary

In Java, we can write OO concurrent applications using the Threads API.

Next week: the API in more detail.

After that: how to use the API properly!

Reading: Chapter 1 of Lea, Sun's Thread javadoc documentation.