Semester Review

14 Apr 2010
CMPT166
Dr. Sean Ho
Trinity Western University

TRINITY WESTERN UNIVERSITY

CMPT166: Java and OO Design

- Java:
  - Basics: access modifiers, static, final, etc.
  - I/O: console vs. file, text vs. object
  - Swing and multi-threading
  - Networking
  - Android (concepts)
- OO Design:
  - Inheritance and polymorphism
  - UML
  - Design patterns

Java basics

- “Relearning CMPT140 but in Java”:
- Basic types, String, arrays, Math
- if/else, while/for, switch
- Packages, public/private/protected
- Exceptions
- Making classes: class vs. instance, attrib/meth
  - static
  - Constructors, copy constructor, this()
- Interfaces

I/O

- Console I/O: System.out/in/err
- File I/O: File
- Network I/O: Socket.getInputStream()

- Text in: Scanner: next(), nextLine(), nextInt(), ...
- Text out: PrintWriter: print(), println(), printf()
- Object in: (File/Object)InputStream, readObject
- Object out: (File/Object)OutputStream, writeObj
  - Serializable interface, transient keyword

Swing

- JFrame, JPanel, widgets (JButton, JTextBox, ...)
- Handling events: ActionListener
  - Using anonymous inner classes
- Layout managers
- Menus, WindowEvents
- Drawing: paint() / paintComponent()
  - Shapes: Line, Rect, Oval, Arc, Polygon
  - Clip
- Swing's thread model: initial thread, event dispatch thread, and worker threads

Networking

- Sockets, IP, host, port
- TCP vs. UDP: pros/cons, example applications
- How to setup TCP server: ServerSocket, Socket
- How to setup TCP client: Socket
- Communicating using streams
- Multi-threaded forking server

Android

- What is it, history, component architecture
  - compare vs. iPhone OS
- Activity, Service, BroadcastReceiver, ContentProvider
- Activity lifecycle: active, paused, stopped, dead
- Views (UI widgets)
- XML configuration of views
- Resources: layouts, strings, drawables
- Event listeners for buttons: OnClickListener

OO design

- Inheritance: "is a kind of"
  - Overriding superclass methods
  - Abstract superclass
  - Polymorphism: designing for extensibility
- Designing class hierarchies

UML

- Use-case diagram: requirements
  - Specifying the bounds of the system
  - Actors, use cases
  - Basic flow, alternate flows
- Component diagram: Class-Resp.-Collab.
- Sequence diagram: messages, flows
- Class diagram:
  - Inherit., assoc., aggregation, composition
  - Direction of dependency; multiplicity

Design patterns

- Creational:
  - Factory Method, Abstract Factory, Builder, Prototype, Singleton
- Structural:
  - Adapter, Bridge, Composite, Decorator, Facade, Flyweight, Proxy
- Behavioural:
  - Chain of Responsibility, Command, Interpreter, Iterator, Mediator, Memento, Observer