Semester Review: Object-Oriented Programming

8 April 2009
CMPT166
Dr. Sean Ho
Trinity Western University

TRINITY WESTERN UNIVERSITY

Course topics

- Practical: the toolkits
  - C++ language, OO constructs, and add-ons
  - FLTK and drawing, fractals, etc.
  - Sockets and threading
- Theoretical: the design process
  - Inheritance, class diagrams, templates
  - Unit testing
  - Use cases, actors
  - Design patterns

C++ language

- Files: *.h, *.cpp, *.o: compiling, linking, running
  - extern
- C pre-processor: #include, #define, etc.
- for, while, if/else, switch/case
- Built-in primitive types; pointers, arrays
- iostream, cin, cout
- References (&) vs. pointers
  - const refs (esp. with operator overloading)
- static (at least 2 meanings), namespaces

C++ OO constructs

- Writing classes: *.h and *.cpp
- Declaring attributes, methods
- public / private / protected
- Constructor, destructor
  - Parameters, default values
  - Constructor initializer list
- Subclassing, polymorphism, virtual methods
  - Abstract (pure virtual) methods/classes
  - Compare w/interfaces

C++ add-ons

- Exceptions: try/catch, accessing exception obj
- I/O: <fstream>: istream/ostream, getline()
- STL <vector>,
- STL strings: <string>:
  - +, insert(), append(), substr(),
  - find/replace()
  - length/capacity/reserve()
  - Sorting
- Templates: methods, classes: declaring, using

FLTK and graphical programs

- GUI history: SketchPad, NLS, Xerox, Apple
- FLTK: using Fluid, widgets, compiling
- Structuring a FLTK program: UI vs. core code
- Drawing: <fl_draw.h>
  - Simple shapes: line, rect/f, arc/pie
  - Complex shapes, transforms
- Fractals, using transforms to draw recursively
  - Sierpinski triangle, chaos game
  - Towers of Hanoi, Gray codes

Sockets and threading

- Sockets:
  - Networking: 7 layers, IP (v4/v6), TCP/UDP, DNS
  - Client-server design: send/recv
  - Programming: socket/bind/listen → accept
  - Multi-threaded server: switchboard
- Threading:
  - Concept, shared memory, issues
  - Locks (mutex, semaphore); deadlock
  - PThreads library: thread callbacks

OO design

- Designing inheritance hierarchies
  - “is a kind of”, “has a”, “knows how to”, “is a”
- UML class diagrams
- Design by contract: pre/post-conditions
- Unit testing: testing+coding, assert()
- Use cases: UML diagram, writing use cases
  - Actors, goals, pre/post, basic/alt flows
- Design patterns:
  - Creational, structural, behavioural

Design patterns: creational

- Factory Method (injection mould):
  - "Virtual constructor"
- Abstract Factory (car parts factory press):
  - Platform to create several kinds of objects
- Builder (assembling fast food kids' meal):
  - Director and hierarchy of Builders
- Prototype (biological cell division):
  - Copy constructor / clone / deep copy
- Singleton (office of the President)

Design patterns: structural

- Adapter/ wrapper: Convert the interface of a class into another interface clients expect
- Bridge: split abstraction from implementation
- Composite: organize objects into trees
- Decorator: dynamically add responsibilities / functionality to an object
- Facade: hide complexities behind simple interface
- Flyweight: use sharing to support large numbers of fine-grained objects efficiently
- Proxy: surrogate/placeholder for another object

Design patterns: behavioural

- Chain of responsibility: avoid coupling sender directly to receiver by passing through chain
- Command: make requests into objects
- Iterator: access all elements of a collection
- Mediator: object encapsulating the interactions of a set of objects: promotes loose coupling
- Observer: decouple viewers from the subject
- *(also: Interpreter, Memento, State, Strategy, Template Method, Visitor)*

Where to go from here?

- Computers are tools →
  - Computer scientists are toolsmiths
  - Know your tools!
  - Computing tools are (mostly) free → the only cost is your time and energy
- Languages: the right tool for the right job
  - CMPT360
  - Java (java.sun.com)
  - Python, Ruby/Rails, Scala, etc.
  - Learn by coding a small fun project!