Duke CPS 108, Fall 1999

- Software Design and Implementation
  - Object oriented programming and design
  - good design helps do away with late night Teer-fests, but some late nights are inevitable
  - your toolkit must include mastery of language/programming and design

- What's in the course?
  - C++ and Java, team projects, mastery exams
  - team projects can be more and less than the sum of their parts
  - high-level abstractions, low-level details
  - patterns, heuristics, and idioms

Duke CPS 108

Program Design and Implementation

- Language independent principles of design and programming
  - design heuristics
    - coupling, cohesion, small functions, small interfaces ...
  - design patterns
    - factories, adapter, MVC aka observer/observable, ...

- Language specific:
  - idioms
    - smart pointers, vectors/arrays, overloaded operators ...
  - idiosyncracies, idiocies
    - must define virtual destructor, stream zoo in Java, ...

Duke CPS 108

Administrivia

- check website and news regularly
  - duke.cs.cps108

- Grading (see web pages)
  - group projects: small, medium, large
  - mastery programs (solo or semi-solo endeavors)
  - readings and summaries
  - tests

- Evaluating team projects, role of TA, UTA, consultants
  - face-to-face evaluation, early feedback

- Compiling, tools, environments, Linux
  - g++ 2.95, Java 2 aka 1.2

Duke CPS 108

C++ idioms/general concepts

- templates
  - generic functions/container classes

- inheritance
  - design guidelines, benefits, drawbacks

- low-level structures
  - C-style arrays and strings compared to standard: STL, Tapestry

- from C to C++
  - function pointers, function objects

- build on generalities, transition to Java
  - vectors, interfaces, stress general principles

Duke CPS 108
Design Criteria

*Good design comes from experience, experience comes from bad design*

Fred Brooks (or Henry Petroski)

- Design with goals:
  - ease of use
  - portability
  - ease of re-use
  - efficiency
  - first to market
  - ?????

How to code

- Coding/Implementation goals:
  - Make it run
  - Make it right
  - Make it fast
  - Make it small
- spiral design (or RAD or waterfall or ...)
  - what's the design methodology?

Design Heuristics: class/program/function

(see text by Arthur Riel)

- **Coupling**
  - classes/modules are independent of each other
  - goal: minimal, loose coupling
  - do classes collaborate and/or communicate?

- **Cohesion**
  - classes/modules capture one abstraction/model
  - keep things as simple as possible, but no simpler
  - goal: strong cohesion (avoid kitchen sink)

- **The open/closed principle**
  - classes/programs: open to extensibility, closed to modification

Design patterns

“... describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice”

Christopher Alexander, quoted in GOF

- **Factory**
  - virtual constructor, have system that doesn’t depend on how items are created

- **Example: iterator**
  - sequentially access elements of aggregate without exposing representation/implementation
  - see DirStream and other classes in libtapestry
  - heavily used in STL, with pointer like syntax
Pattern Essentials

- **Name**
  - good name provides a handle for the pattern, makes it easy to remember and use: vocabulary

- **Problem**
  - when the pattern is applicable, context, criteria to be met, design goals

- **Solution**
  - design, collaborations, responsibilities, and relationships of the classes/design elements

- **Consequences**
  - trade-offs, problems, results from applying pattern: help in evaluating applicability

larger group/company issues

- establish a shared vision
  - what is hyperwag? what can we add?
  - harmonious sense of purpose
- develop a creative environment
  - the more ideas the better, ideas are infectious
  - don’t flip the BOZO bit
- scout the future
  - what’s coming, what’s the next project
  - what new technologies will affect this project

see McCarthy, Dynamics of Software Development