CPS 108, Spring 1998

- **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
What’s ahead in programming and design

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

- **Language specific:**
  - idioms
    - smart pointers, vectors/arrays, overloaded operators ...
  - idiosyncracies, idiocies
    - must define virtual constructor, comma operator ...
Administrivia

- check website and news regularly
  http://www.cs.duke.edu/~ola/courses/cps108.html
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++ 8.0 released 11/14/98, read website
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
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 ...) see *DOOM*
Design Heuristics: class and program rules-of-thumb

(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
Roulette: Classes, Responsibilities, Collaborators

- **Class diagram in UML**
  - unified modeling language (see class web page)
  - system at a glance, relationships, heuristics, discussion

- **What does inheritance get us here? (see the non-OO version)**

- **Why does a bet need a wheel? other issues?**

- **What are responsibilities of bankroll?**
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

- **Example: iterator**
  - sequentially access elements of aggregate without exposing representation/implementation
  - see DirEntry, List class, Map class
  - heavily used in STL, with pointer like syntax

- **Factory**
  - virtual constructor, have system that doesn’t depend on how items are created
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
hyperwag: Week-at-a-glance in hypertext

- hypertext version of week-at-a-glance, see web page for complete details
  - read file in wagalang (other languages too)

```plaintext
# this is a fun week
m  9,12  sleep
th 9,10  cps 110
w  9,10  breakfast
f  9,10  running
mtwhf 12,13 lunch
wf 10,12 cps 108
t 10,12 breakfast
```

- produce HTML table

- classes? problems? issues? groups? ideas?
- C++ questions, design questions
larger group/company issues

see McCarthy, *Dynamics of Software Development*

- 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