Lecture 18: Refactoring

Kenneth M. Anderson

Object-Oriented Analysis and Design

CSCI 4448/6448 - Spring Semester, 2005

Credit where Credit is Due

- Some of the material for this lecture and lecture 19 is taken from "Refactoring: Improving the Design of Existing Code" by Martin Fowler; as such, some material is copyright © Addison Wesley, 1999

Goals for this lecture

- Introduce the concept of Refactoring and cover a few examples
- In lecture 19, we will present a tutorial that will introduce a few additional refactoring techniques

What is Refactoring

- Refactoring is the process of changing a software system such that
  - the external behavior of the system does not change
    - e.g. functional requirements are maintained
  - but the internal structure of the system is improved
- This is sometimes called
  - "Improving the design after it has been written"

(Very) Simple Example

- Consolidate Duplicate Conditional Fragments (page 243); This

```
if (isSpecialDeal()) {
   total = price * 0.95;
   send()
} else {
   total = price * 0.98;
   send()
}
```

- becomes this

```
if (isSpecialDeal()) {
   total = price * 0.95;
} else {
   total = price * 0.98;
}
send();
```


Refactoring is thus Dangerous!

- Manager's point-of-view
  - If my programmers spend time "cleaning up the code" then that's less time implementing required functionality (and my schedule is slipping as it is!)
- To address this concern
  - Refactoring needs to be **systematic**, **incremental**, and **safe**