COMP 401
CONCLUDING REMARKS

Instructor: Prasun Dewan
**Course ➔ Future**

- Project
- Topics
- Praxes
- Diaries
- Quizzes
- Exam(s)
- Advanced Courses
- LAship
- Internships
- Research
- Building an app
# Quick Review of Design Patterns

<table>
<thead>
<tr>
<th>The design patterns taught in Comp 401</th>
<th>Prasun Dewan, Teaching Inter-Object Design Patterns to Freshmen. Proceedings of ACM SIGCSE. 2005</th>
<th>PPT</th>
</tr>
</thead>
</table>
FUTURE OPTIONAL COURSES

- Scanning
- Objects
- Overloading
- Properties
- Interfaces
- Shape objects
- Composite objects
- Collections
- Inheritance
- MVC, Observer
- Toolkits
- Graphics (painting) views
- Assertions
- Animation
- Command Objects
- Threads
- Synchronized Methods
- Wait and Notify
- Abstract Classes
- Recursive Parsing and Grammars
- Trees, DAGs
- Generics
- Factories, Adapter, Delegation
- Exceptions

- Human Computer Interaction
- Software Engineering (Design Patterns)
- Operating Systems
- Compilers/PL/Internet Protocols
Large-Scale Object-Oriented Programming!

Large in terms of number of number of classes/types

Defined by you (Programmer-defined)

Count them!

Army of “paint listeners”

Largest (in terms of components) you may write at UNC or elsewhere

Refactoring
**PROGRAMMING PROCESS AND TOOLS**

**Class path and libraries (Eclipse)**

**Following Requirements**

Comp 401 - Assignment 12: Wait Notify, Generics, Etc. (Last One!)

Date Assigned: Tue Nov 8, 2016
Completion Date: Wed Dec 7, 2016 (No late submission!)
Early Completion Date: Mon Dec 5, 2016 (Monday)

assignment will change as we cover new material. The required part addresses wait notify and generics. There is extra credit for exceptions, recursive descent, undo and redo.

**Debugging (Eclipse)**

**Using a Test Suite (JUnit)**
STYLE CHECKS (CHECK STYLE) → SECURITY

```java
package main;

Multiple markers at this line
- expectedDeclaredSignatures: (Assignment1.java:3) In type Assignment1, missing declared signature: processInput: -> void
- typeDefined: (Assignment1.java:3) Class/Interface Assignment1 matching tag main.Assignment(.* ) defined
```
Praxis: Active Learning, Maintenance, Version Control

public class AConsoleReadingUpperCasePrinter {

/**
 * MAIN METHOD HEADER
 * Syntax of main method shown below.
 * Methods correspond to procedures and functions in other languages.
 * Method names should be camel case starting with lowercase letter.
 * Everything before the first curly brace is the method header.
 */

public static void main(String[] args)
{ /*
 * What happens if you use the following header instead, can you execute the program?
 * Comment out the header above and uncomment the following to see what happens?
 * What is the difference between the two headers?
 */

// public static void main(String args) { 

/**
 * METHOD BODY
 * The code between the outermost curly braces is the method body.
 */
INTERPRETER OF PROGRAMMING LANGUAGE
Use Methods

- Drag method names from the details window Methods area to world.my first method.
- Can Group Methods:
  - Do in order: One after the other
  - Or Do together: At the same time

You built an Alice-like visual programming environment that can be used by others to program.

Beginning programmers learn how to use Alice.

Barbara Ericson ericson@cc.gatech.edu
Georgia Institute of Technology
BUILDING AN APP

- Handling of User and Internal Errors
- Concurrency
- Flexible, Multiple User Interface
- Scanning
- Parsing
- Geometry Processing
- Use of Window Systems and Toolkits
Logical Data Structure Visualizer, Style-based Tool, Training Wheels (ObjectEditor)

```java
public Line getXAxis() {return xAxis;}
public Line getYAxis() {return yAxis;}
public StringShape getXLabel() {return xLabel;}
public StringShape getYLabel() {return yLabel;}
public int getAxesLength() {return axesLength;}
public void setAxesLength(int anAxesLength) {
    axesLength = anAxesLength;
    xAxis.setWidth(axesLength);
    yAxis.setHeight(axesLength);
    xAxis.setX(toXAxisX());
    xAxis.setY(toXAxisY());
    yAxis.setX(toYAxisX());
    yAxis.setY(toYAxisY());
    ...}
```
DIARIES AND Q/A

Written and oral skills

Abstraction
QUIZZES AND EXAMS

Make you (and me) think about and identify what you have programmed
MODULARITY

Class Decomposition into Methods: Recursive descent programming

Program Decomposition into Classes: Design Patterns

Process Decomposition into Threads: wait, notify, synchronized methods, command objectys
WHAT did you not LEARN

- Distributed Systems
- Efficient and Complex Data Structures and Algorithms (Implemented by Java)
- How a computer system works on regular and mobile computing (Architecture, OS, Compilers)
- Proving things about what your program can (not) do
- Team Programming