Australian National University

UML Class Diagrams - Design

Eric McCreath

Plan to Design

Once you have a clear idea of what you want the software to do (requirements) then spend some time designing your code.

The main and most important part of your design is how you break your code up into modules. From an OO persective this will basically be your class diagram (generally expressed using UML).

Spend some time designing your code. However, keep it in perspective of the length of your project.

Bad design will make for a difficult and fragile implementation.

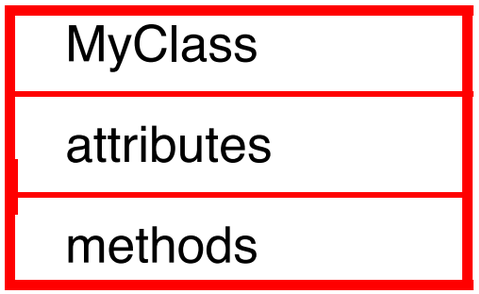
Reasonably good design will produce an implementation that can always be re-designed (with considerable code reuse).

UML

UML (Unified Modeling Language) is a standard way of describing a program's design. We are focusing on UML Class Diagrams in this course.

A box represents a class or interface.

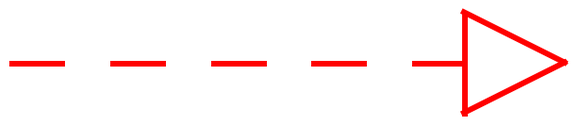
Classes/interfaces may also have the attributes and methods annotated.



Lines between classes/interfaces represents a relationship between them.

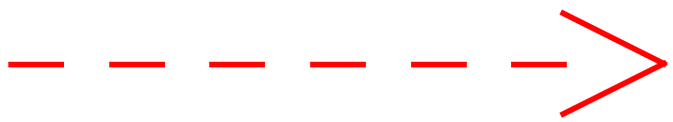
Inheritance "is a"

Interface Implementation "does this"



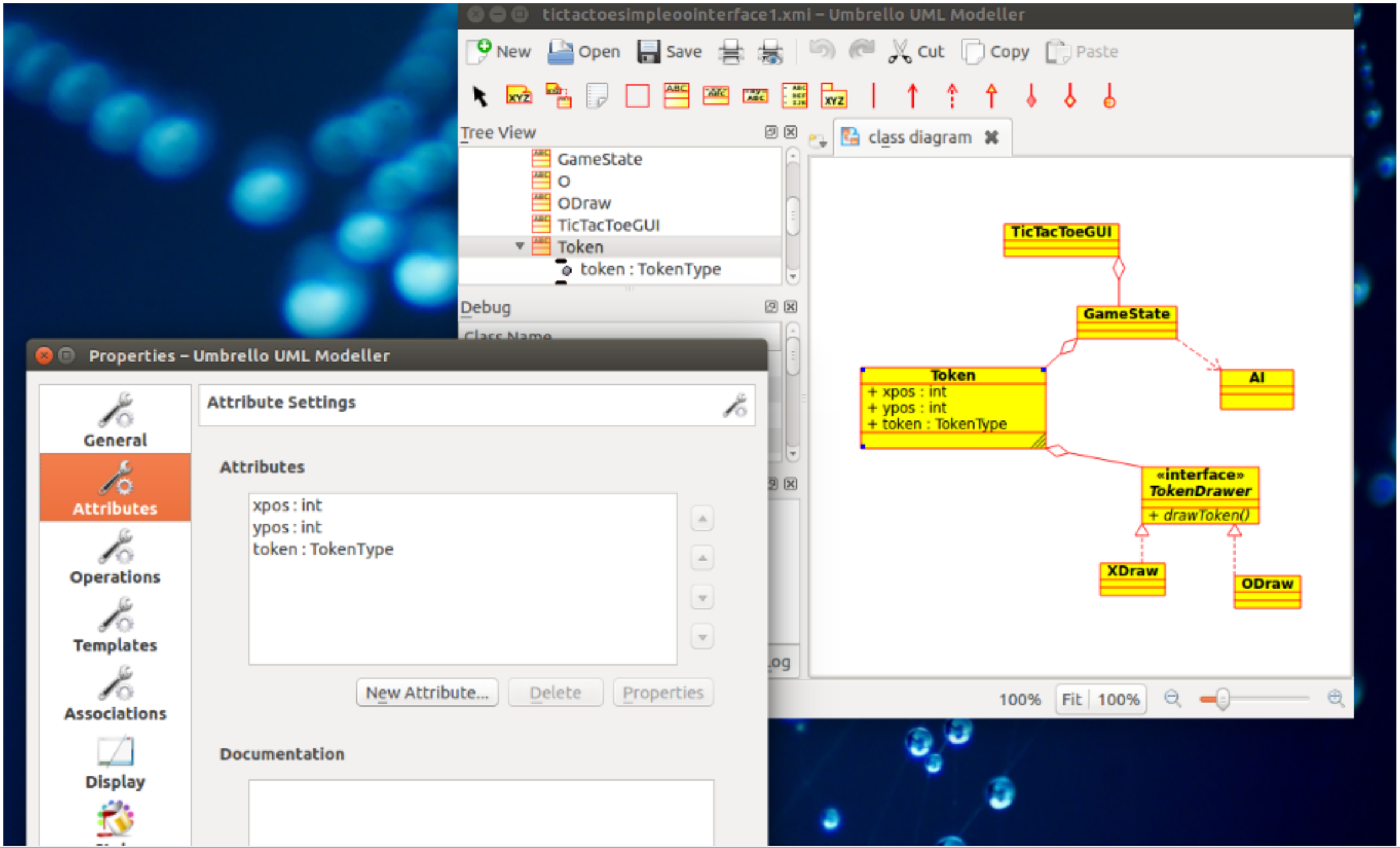
Aggregation "is made up of"

Dependency "uses a"



UML

Umbrello is a useful tool for creating UML diagrams.



Challenge

- Have a go at using Umbrello.

Australian National University

GIT - The Basics

Eric McCreath

Basics of git

- Install git, see:

```
https://git-scm.com/book/en/v2/Getting-Started-Installing-Git
```

- Tell git your name and email address

```
$ git config --global user.name "Your Name"
$ git config --global user.email u1234567@anu.edu.au
```

Now you are ready to get started.

- To get help:

```
$ git help
```

You can also get help on a particular command. e.g. getting some info about the "git pull" command type:

```
% git help pull
```

Gitlab

Gitlab provides a powerful web-interface and server for managing a collection of git projects. Along with providing a git repository for each project, it also provides:

- a project wiki
- ability to view/edit files
- visualization of the network of commits
- a way of managing the project team and their individual access rights
- issues and milestones for a project

git - external repositories

- Duplicate (or clone) an external repository:

```
% git clone <url or directory of other repository>
```

- Obtaining updates from an external repository that you have cloned (this will also update the working files and will set this repo as the upstream master):

```
% git pull
```

- Pushing changes you made in your local repository to the repository you cloned from:

```
% git push
```

- Before performing a *push* of updates made, one needs to *add* and *commit* the changes made to the local repository

git - local commands

- Staging a file to be added to the local repository:

```
% git add filename.c
```

Note that you also need to *add* files to this staging area when you make modifications to them.

- Committing the local updates:

```
% git commit -m "comment here"
```

- Obtaining the status of the local repository:

```
% git status
```

- Or for a summary of the status:

```
% git status -s
```

- Looking at the log info:

```
% git log
```

git - ignore files

- Generally only source code is placed under revision control. However, the "git status" command will list other files that have not been added to revision control.

- A list of files can be added to the **.gitignore** file and git will ignore those files.

- *.gitignore* may also contain filename patterns such as "*.class"

Challenge

- Install git on your computer.
- Make a repo on https://gitlab.cecs.anu.edu.au/
- *clone, add, commit,* and *push* some files.

Australian National University

Introduction - Assessment Structure

Software Design Methodologies/Software Construction(COMP2100/6442)

Dr Eric C. McCreath

Details about the assessment structure for this course can be found at:

https://cs.anu.edu.au/courses/comp2100/outline.html

Australian National University

Challenge

- Engage with the course!
- Register for a lab and a workshop.
- Schedule some time for your prep.
- Do a S.W.O.T. analysis of the assessment scheme.

Australian National University

Introduction - Overview of Course Content

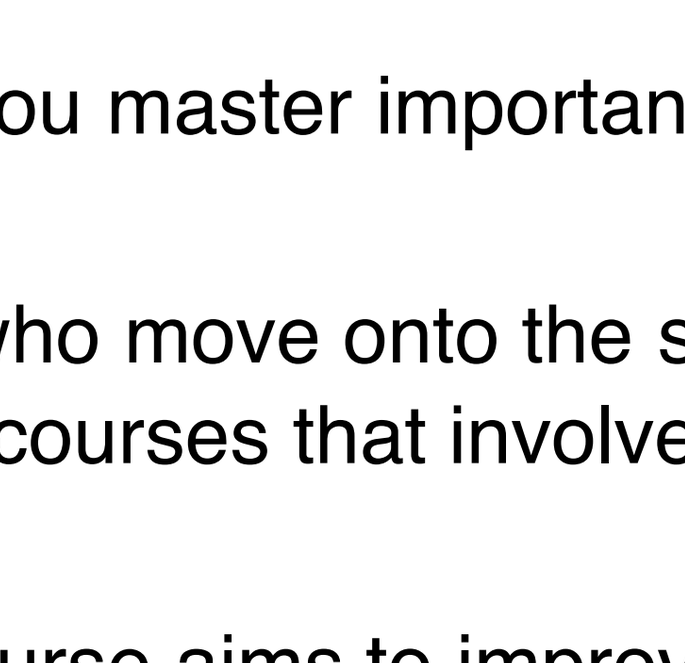
Software Design Methodologies (COMP2100/6442)

Dr Eric C. McCreath

From Understanding to Mastering

- This course will help you master important aspects of software construction.
- This will help people who move onto the software engineering group project and other courses that involve software development.
- More generally this course aims to improve your programming and software engineering skills for university and beyond.

The Pin is Mightier Than the Sword!



From http://en.wikipedia.org/wiki/Pin_%28chess%29

Adding Two Integers

Even for something very simple, such as adding up two integers, there are many different ways of writing code for a "correct" solution.

Given the below interface how many different ways can you think of?

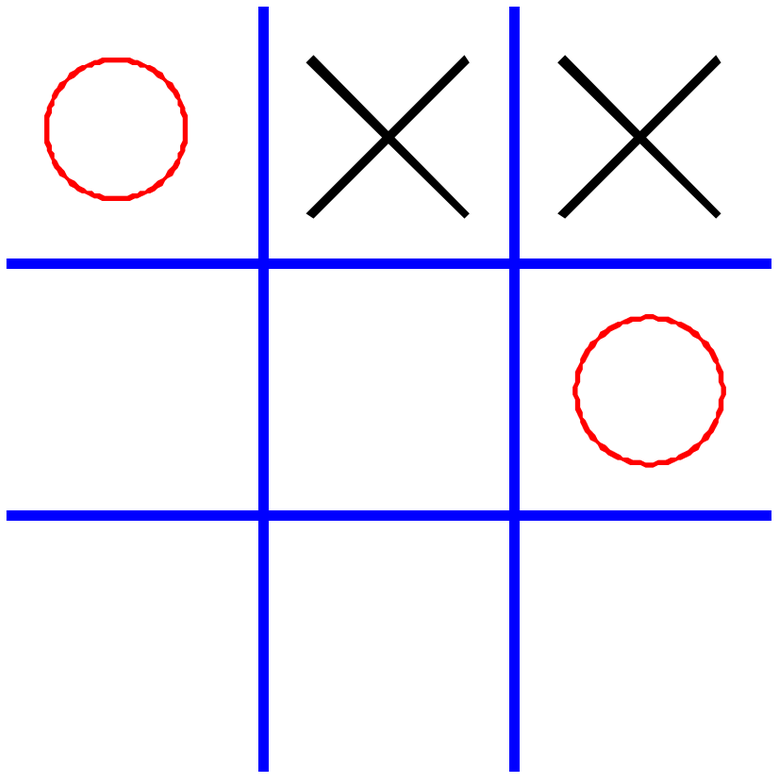
```
public interface IntAdder {
   public int add(int a, int b); // to make it simple let us
                                 // assume non-negative integers
}
```

Design Approaches

Let us consider a more serious design example, the game of tic-tac-toe.

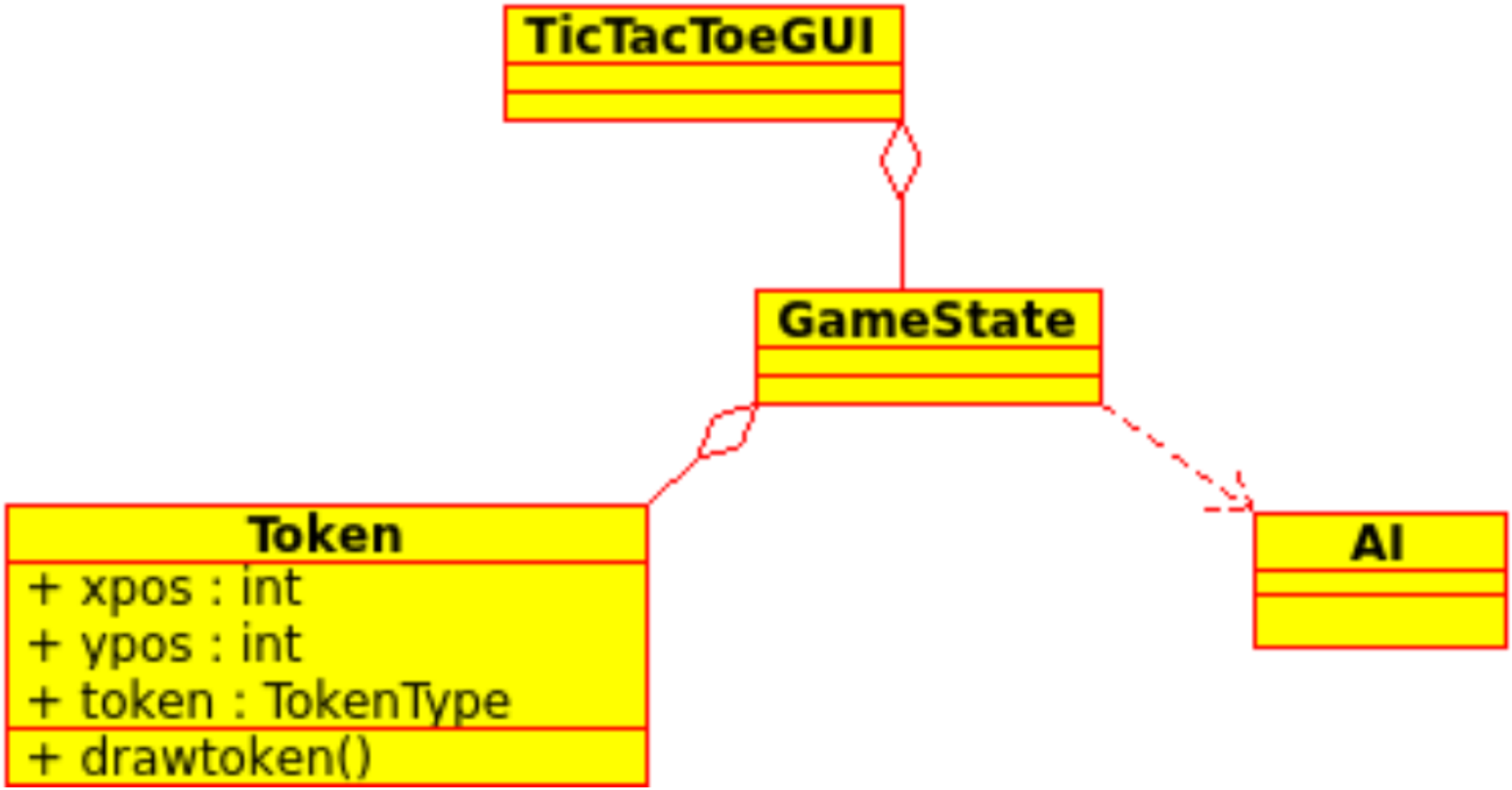
Consider the overall design of a simple GUI for a tic-tac-toe game. What would your overall design be like?

Create a simple UML diagram of your overall design.



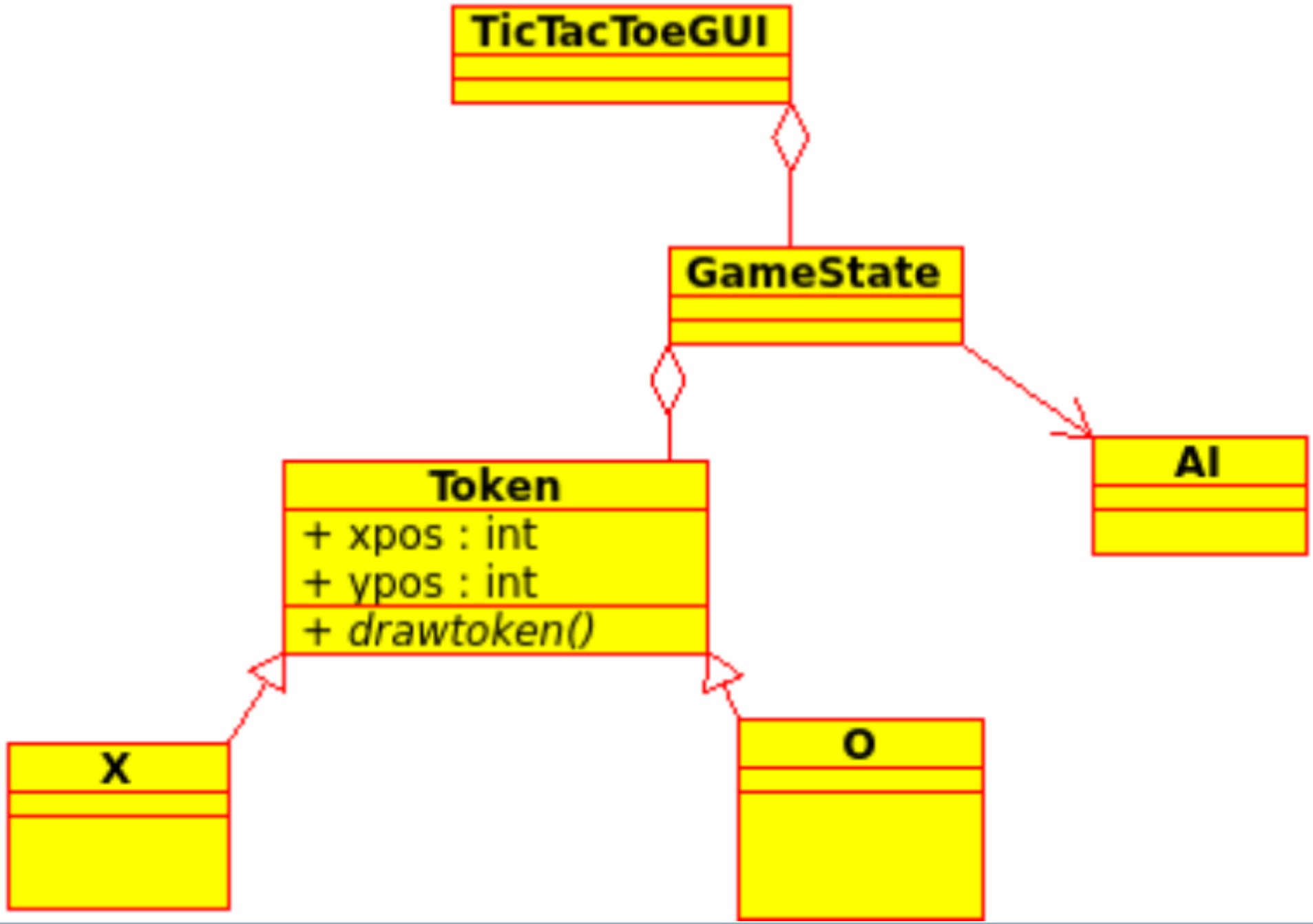
Simple Design

The below approach uses classes to break the code up into modules and creates some simple types.



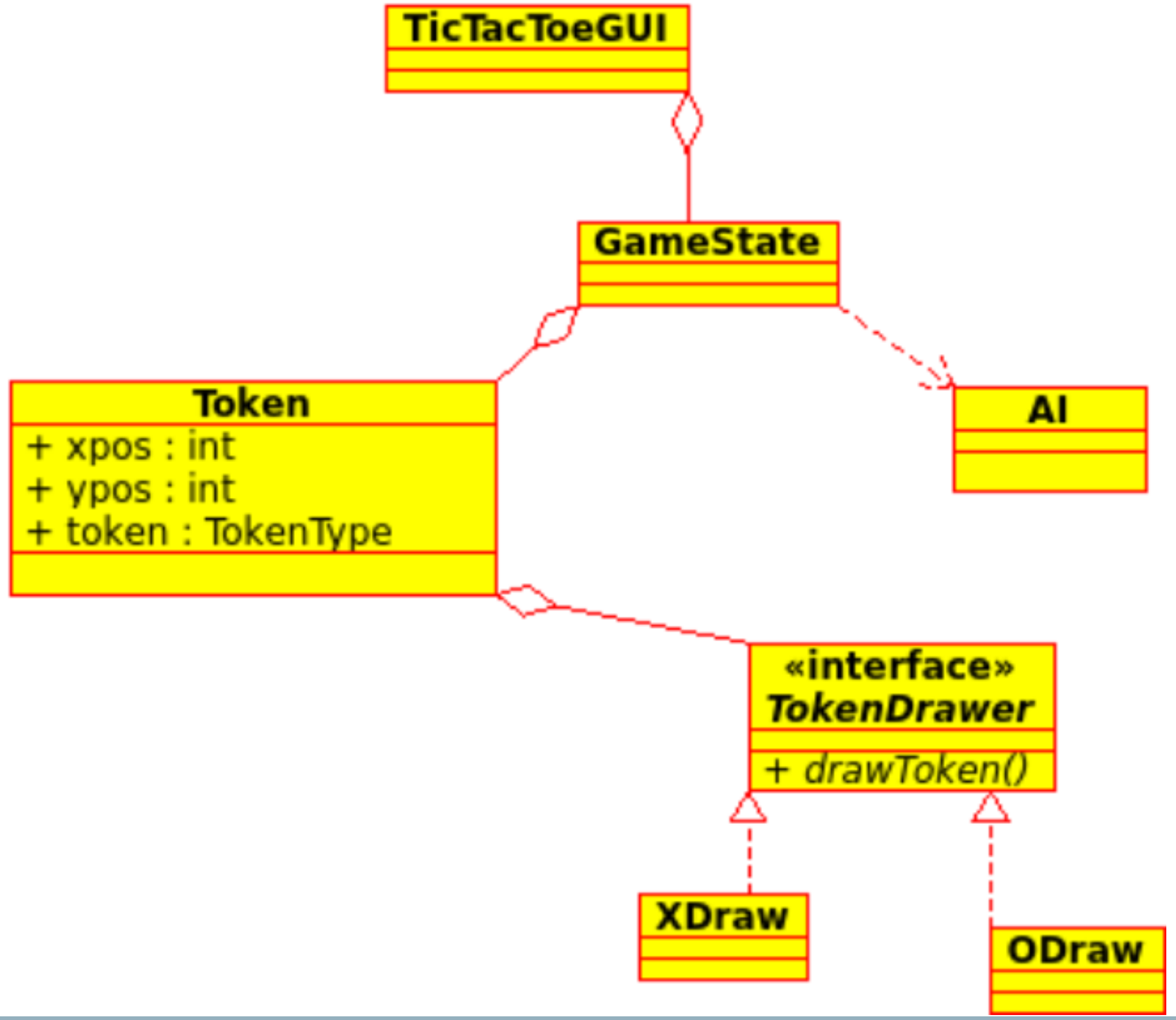
Simple Fixed OO Design

We can get rid of if-else conditionals and let the sub-classes do this work.



Using Interfaces to Gain Some Flexibility

Flexibility can be gained by using aggregation of an interface.



Perspective on Executing Code

For a programmer what is so important about:

$$10^9$$

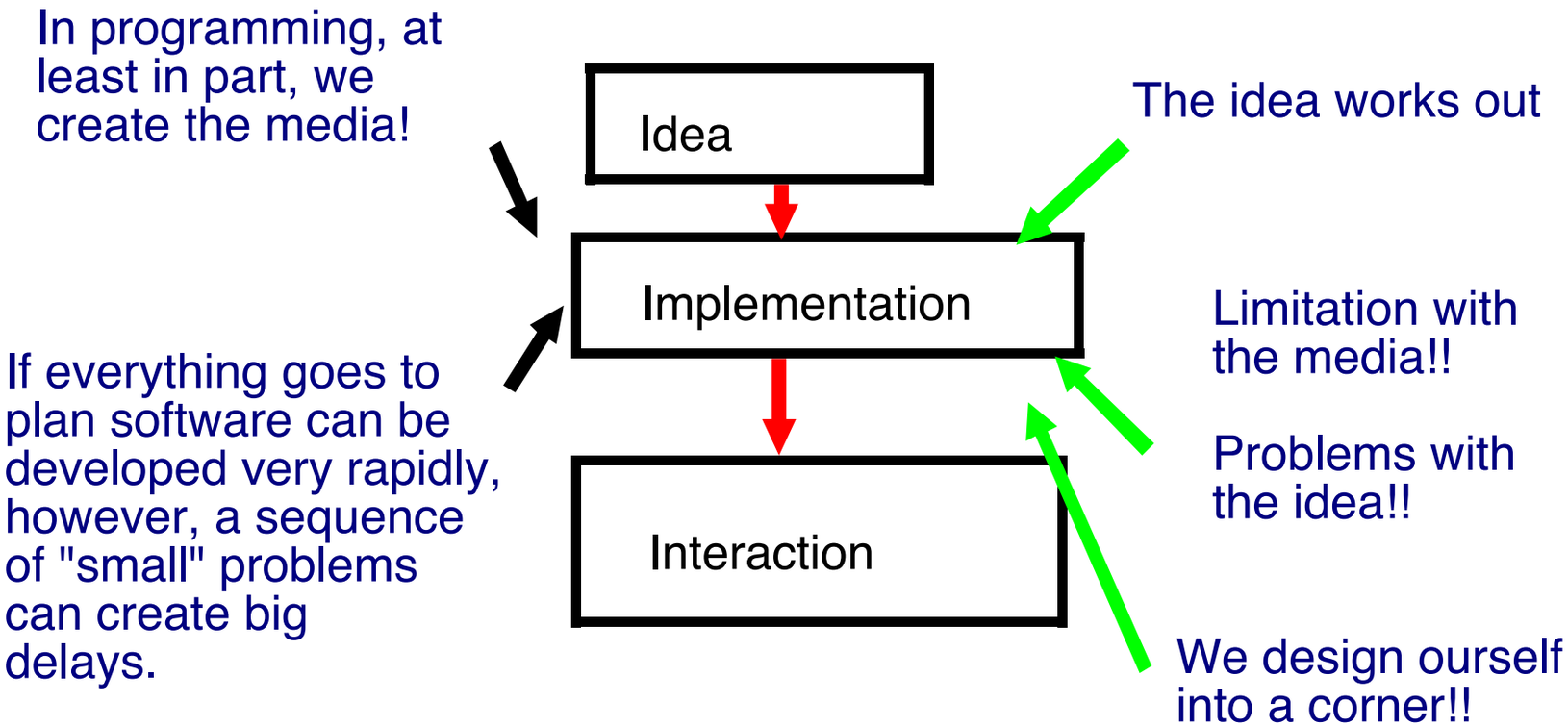
or

$$1000000000$$

?

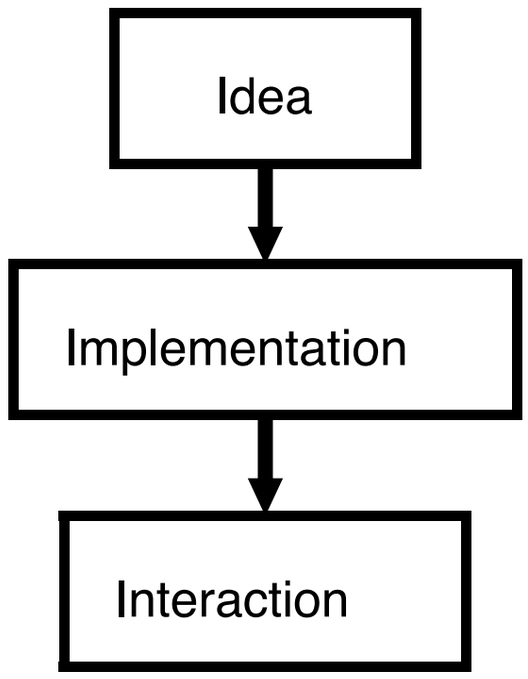
Programming

- Programming is a creative art.
- Software developers are optimists.



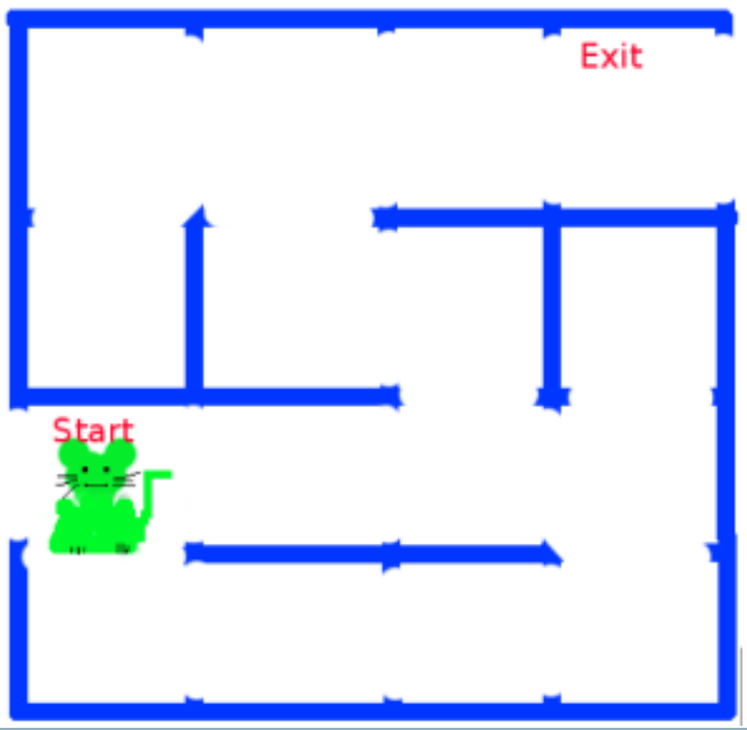
Programming

- To move from 'Idea' to 'Implementation' you need to have a clear mental model of computation.
- Within this mental model you need to work out a sequence of steps (a plan) that will solve the problem. This will often involve creating or using data structures.
- Also it will involve having a good understanding of the syntax and semantics of the programming language.



A Mouse in a Maze

- Programming is like constructing a list of instructions that would enable this mouse to get out of the maze.
- Note you have the following:
  - a mental model of how this constrained world works;
  - a clearly defined goal; and
  - a language that defines the instructions the mouse can execute (up, down, left, and right).



What is Software Engineering?

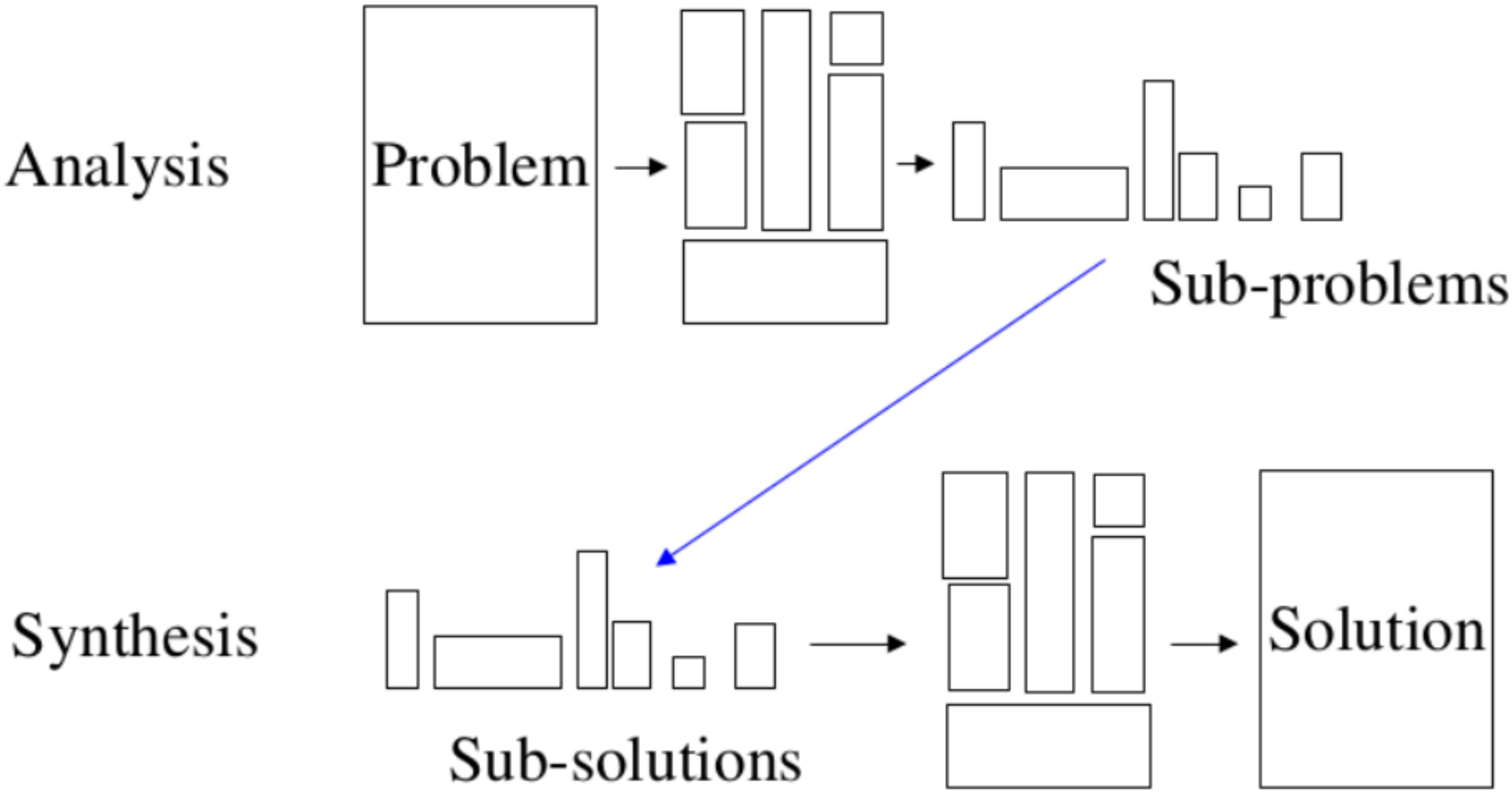
- Software Engineering is the discipline of developing, deploying, maintaining, and retiring of software. Its key focus is quality and the discipline encompasses all the phases of the software's life cycle.
- *"Software engineering attempts to combine sound engineering principles of analysis, design, management, testing, and so forth with scientific concepts of computing in the production of reliable and cost-effective computer software that meets its users' needs."* , Data Structures and Software Development - Tremblay and Cheston

An Engineering Approach

- Few people would build a house without a plan! Software is no different. Detailed and clear plans are required.
- An Engineering approach is needed.
- Given the complexity of software this process is difficult.
- However, the flexibility and modularization of software provide ways of addressing complexity in software development.

Analysis and Synthesis

- The design/implementation process may be thought of as analysis and synthesis. This maps a problem to a solution.



The Software Life Cycle

The software life cycle includes the following:

- Requirements (analysis and definition)
- Planning and management *
- Overall design *
- Implementation *
- Testing (unit, integration, and system) *
- Delivery
- Maintenance (debugging, supporting, and modifying)
- Retirement

This course will focus on items marked with * .

Challenge

- When do you make plans in life? How are they recorded?
- Come up with your own design for the tic-tac-toe game. How would you separate model-view-control?
- Write a Java program. Any!

Software Design Methodologies

(COMP2100/6442)

Eric McCreath

Acknowledgement

" We acknowledge and celebrate the First Australians on whose traditional lands we meet, and pay our respects to the elders of the Ngunnawal people past and present. "

- The official ANU Acknowledgement wording see: http://reconciliation.anu.edu.au/

Welcome

As the title suggests this course is about software design methodologies, previously it was called software construction which also captures the course's purpose.

I hope that in this course you will:

- improve your ability to design, implement and test software,
- you will be able to reflect and analyze different designs,
- you will gain or improve skills in the use of tools for software development (eclipse, ssh, JUnit, git, etc..), and
- be able to understand moderate sized software systems and make modifications/additions to them.

Flipped Course

- This course will run in a flipped mode.
- Lectures are pre-recorded and you are expected to watch them sometime during the week they are scheduled.
- Group programming activites will be done in the large main meetings. These meetings will also be used for quizzes. There are participation marks.
- Individual lab programming tasks will be done and evaluated during the 3 hour lab sessions.
- The usual reading, reflecting, reviewing and programming practice will be done outside class time.

In the end I hope you will find this course interesting, challenging, and useful.

Admin Details

- Course Convener/Lecturer : Dr Eric McCreath - eric.mccreath@anu.edu.au, CSIT Building Level 2
- Tutor/Lecturer: Dr Michael Curtotti - michael.curtotti@anu.edu.au
- Contact hours (the hour after the large meetings). These are in my office. Note you can also drop past at other times.
- Web site - https://cs.anu.edu.au/courses/comp2100/
- Labs start Week 2 - register via https://cs.anu.edu.au/streams/
- No text book.

Challenge

- Reflect on your strengths and weaknesses in programming.
- Think about the largest software project you have developed.
  - What was the design like?
  - What did you refactor?
  - What was good about the design?
  - What where the designs limitations?