Introduction to the Course

Course Goal

The overall goal of the course CMSC 433 has traditionally been to explore useful coding abilities that programming languages support, and learning how to use them effectively.

In short, this class always looks to:
- Introduce advanced programming technologies.
- Deconstruct relevant programming problems.
- Solve them using the advanced technologies.
Area of interest this semester...

Concurrent/Parallel/Distributed Programming
- Issues in multi-threaded algorithm design
- Concurrent collections and shared memory
- Single versus multiple machines running tasks
- etc.

400-level CMSC courses...

How many 400-level CMSC courses have you taken before?

How many are you taking now?

Have you taken 412 or 417?
Which values are possible for i and j? (click all that are possible)

0% 1. i=1 and j=0
0% 2. i=0 and j=1
0% 3. i=1 and j=1
0% 4. i=0 and j=0

x = 0; y = 0;
Spawn thread #1 {x=1; j=y;}
Spawn thread #2 {y=1; i=x;}
Wait for them to end.
Output i and j

Course approach...
The course will contain both:

• Big Ideas
  – We will discuss not only syntax but also high-level concepts in class. Class discussion will actually have discussions from time to time and some in-class practice activities, so even with any slides that are posted, attendance will be an important thing.

• Big Implementations
  – Projects will be individual efforts. You will be able to ask questions about the concepts and have access to some coding example (details with the projects) but you will not be allowed to share code or debug each other’s code.
**Textbook: Optional**

The book site has many code examples even if you don’t buy the book.


How do you decide whether to buy it?
Let’s discuss this now...

Links to other sources of examples are likely to appear on the class website as we cover topics.

You will be allowed to search for online examples of concepts. If you use code snippets in your project you will need to cite the source, even if you modify the code you found to fit the project.

---

**Projects**

Plan is to have six projects.

Weighting varies between projects (see syllabus).

Some starter files will likely be provided though you will find you are building much from the ground up.

Some test scenarios might be provided, but you are **strongly urged** to design your own project test cases as well. We tend to have to hand-test many things in the projects.
**Project Submission**

Each project will be due by Midnight (23:59:59) on the due date.

- You should use the Linux lab cluster `date` command to determine the exact time of day as we know it.
- You must submit a good-faith effort for each project.
- Only your **last submission** of each project will be graded and if a test scenario “works” some of the times it is executed but not all times, it will generally be considered to be failing that test.
- If you do not have a multi-core machine you should test your project on the Linux lab machines too.

---

**Project Grading and Class Accounts**

We will use the Submit Server system for project submission (submit.cs.umd.edu) but much (if not all) of the testing will be “secret” and might be done by hand during the grading process.

Linux lab accounts will be available

- You will need to use Eclipse to submit your project and will likely find it useful to develop there. There are times you might want to test on machines with more CPU cores. You can use your own campus *NIX accounts for course work too if you’d prefer that.

We will post scores to the grades.cs.umd.edu server.

Some reading or reference materials might be posted to the University ELMS site (elms.umd.edu).
**Development Environment**

You will be asked to use Java 6 or 7 and Eclipse 4.2 (Juno) and Junit 4 for projects but we won’t be using features that are new to Java 7.

It will often be useful to bring your laptop to class when we examine source code examples there.

If you go to TA office hours with a project coding question, it would again be best to bring a laptop with your code.

If you go to faculty office hours with a conceptual question a laptop probably won’t come in handy.

---

**Written Exams**

Two semester exam anticipated dates:  
March 4\(^{th}\), April 15\(^{th}\)

Final exam:  
Wednesday, May 15th 1:30pm-3:30pm
Grading (see syllabus for full details)

<table>
<thead>
<tr>
<th></th>
<th>#</th>
<th>Total value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Projects</td>
<td>6</td>
<td>50%</td>
</tr>
<tr>
<td>Semester Exams</td>
<td>2</td>
<td>30%</td>
</tr>
<tr>
<td>Final Exam</td>
<td>1</td>
<td>20%</td>
</tr>
</tbody>
</table>

Discussion and Questions

Visit the class web site regularly – class-wide announcements will appear there as well as code examples and lecture slides.

By default, no project code or pseudo-code is allowed to be posted! If in doubt, e-mail me *first* to ask!

You can discuss ideas on how to test your projects as well, but again please do not post actual code without checking with me first.
**Contact Info**

Evan Golub: egolubUMD@gmail.com, 1115 AVW

William Goh: 1112 AVW

Specific hours will be posted and kept up to date on the class web site.

I am asking that project coding-level questions mostly be reserved for in-person Q&A with the teaching assistant. I am happy to discuss class examples and topics in general at the conceptual level in office hours. Please reserve e-mail messages for short questions.

**Excused Absences**

Inform me directly of any religious holiday dates or other University-approved excuse dates before February 5\textsuperscript{th}.

Medical emergencies:

- Must provide written documentation from a medical professional stating what dates/times you were incapacitated with contact information.
- Self reporting is not sufficient.
- Since projects are assigned over an extended period of time, any requests for extra time on projects due to medical issues will need to be based on extended issues as well.
Let's consider a “simple” example of a shared resource: a printer.

(Dates loosely based on my own experiences and not generalizable but still of use as a progression.)

1980: Printer connected directly to a PC. No sharing. No problems, but expensive...

1990: Lab full of computers where each row was a chained network with a printer at the end of it. All saw the printer as if it were local. To print you would go to the printer, make sure nothing was being printed, call out “I’m about to print” and go back to your PC and select to print, and then hope no fool sent something to the printer until you were done...

2000: A printer could be connected to a PC on the network and requests to print would go to that PC which would then queue things up. If the PC was turned off, no network printing...

2010: The printer is itself right there on the network with its own IP address and queuing software built in. It can be found by others on the network or maybe even the world. This could be a problem...