"CPSC 240 — Object-oriented Analysis & Design"

Professor: Stephen Davies
Spring semester 2019

Class: TR 11am–12:15pm *or* TR 12:30–1:45pm, Trinkle B6
"Lab": F 10–10:50am *or* F 11–11:50am, Trinkle B6

Final exam: **Tuesday, April 30th** *or* **Wednesday, May 1st**,
High Noon, Trinkle B6

Office Hours (in Trinkle B22):

Mon	1–2pm
Tue	2–3pm
Wed	3–5pm
Thu	10–11am

`http://cs.umw.edu/~stephen/cpsc240`

Design vs. code

The precursor to this course (CPSC 220) had the word "programming" in its title. This one, in contrast, features the words "analysis" and "design." Is this just semantics, or is something deeper implied by this choice of terms?

The wording was deliberate, and in many ways it's a matter of *scale*. When we talk about **programming**, we think of individual lines of code that are assembled to create a function or a loop. But the word **design** connotes something larger: a system complex enough that it requires advanced planning and organization to keep it under control. Programming is tactical: your focus is on what's in front of you and on getting the details right. Design is strategic: your focus is on the big picture and on outlining a structure into which all those details can fit together.

Although at this point of your career you've acquired some essential programming expertise, two important things have probably been missing from your experience: (1) big projects, and (2) team projects. The early courses in any computer science curriculum focus on tiny programs that you complete on your own. They may not have seemed tiny at the time, but if you compare them with anything you use on a daily basis – Excel, Firefox, YouTube, Fortnite – you'll realize just how small they really are.

It turns out that once a software project reaches a certain size, it undergoes certain changes that make it unlike anything smaller. That's

not only because of increasing complexity, but also because it's now too big for any one person to hold in their head at once. You now have to depend on an intuitive design and accurate documentation to have a fighting chance of writing the code correctly. This is an utterly new kind of experience.

Working cooperatively with others is also a new kind of experience. You have to depend on someone else's code working right in order for yours to work right. This is nearly always how software development works in the real world, and gaining practice with this paradigm will be a primary focus of this class.

When these two pieces of the puzzle are in place, you will have acquired an important level of mastery. By the end of this course, you will know enough about how software is really built to be professionally hire-able. You should be able to walk in to your average development team as a ground-floor software engineer and basically know what the heck you're doing. It will be challenging, but also worth it in the long run. So hang on and enjoy the ride.

Course objectives

To provide students with:

- opportunities to design, implement, debug, and document original larger-scale programs using techniques of good object-oriented design
- experience using design techniques for identifying classes and methods
- an introduction to UML including class diagrams and sequence diagrams
- opportunities to study the role of encapsulation, abstraction, visibility, inheritance and polymorphism in object-oriented design
- a collaborative, team development experience
- an introduction to the concept of design patterns and the application of common patterns
- feedback to help students improve their writing skills needed in computer science
- a small taste of multi-threaded (parallel) programming concepts, in preparation for further development of these ideas in later courses

Rules of the game

1. There are NO stupid questions. I will never belittle you or make you feel dumb about anything. Your job is not to already know everything before you start, but to roll up your sleeves and work hard to try and learn. Even if your question is, "Stephen, I totally didn't even get that, can you start over and explain it again?", please ask.

2. This class will be interactive. When I point at you in class, say your first name, and be prepared to try and answer questions. (Don't worry if you don't know all the answers.)

3. Except where explicitly indicated, you must complete all work for this class entirely on your own, without any help from any other source (see "The Honor Code and this course," below).

4. Please, **no laptops during lecture**. I've had students claim that they take notes on their laptop during class, but even if it's true, those things are too big a distraction to you and your fellow students to make it worth it. Just stay tuned in, because I move fast.

Books

We'll be using my book this semester, entitled *Blueprints: Creating, Describing, and Implementing Designs for Larger-scale Software Projects.* I will make it available online, but **DO BUY** the coursepack from the Bookstore as well, bring it to class, and mark it up with your own notes. (I get no royalties for this, by the way.)

Calendar

The calendar for the course, complete with assignment due dates, tests, *etc.*, will be maintained on the course website at `http://cs.umw.edu/~stephen/cpsc240`. **In any case where they differ, the website supersedes this syllabus.**

Late policy

No late work will be accepted this semester. Get your stuff in on time, there's no excuse not to!

The Honor Code and this course

I strongly believe in UMW's honor code and scrupulously adhere to it. Here are the rules for this course:

1. The quizzes and final exam are to be solely your own work. They are open book and open notes, but they are both **closed Java compiler** and **closed to other humans.**
2. For the *individual* homework assignments, you must work **entirely on your own.** To be clear:
   - Discussing an assignment with a fellow student in the course is a no-no.
   - Discussing an assignment with a previous student from this course is a no-no.
   - Discussing an assignment with anyone who has never even taken the course is a no-no.
   - Googling for code snippets or design ideas is a no-no.
   - Referencing StackOverflow, or any website other than the Java API, is a no-no.

   All of the above are Honor Code violations.
3. For the *group* activities, you are not only permitted, but mandated, to work with your group openly, frequently, and without reservation. For the *group* activities, it is also permissible to Google for code snippets or design ideas.

Disabilities

If you have a documented disability, please present me your letter from the Office of Disability Resources and I'll be happy to accommodate you.

Grading

Grading this semester will be based on the notion of "experience points" (XP). The basic ideas are: (1) you can never lose points, only gain them; (2) you have to earn what you get (*i.e.*, you don't "start off with a 100%" and lose points based on mistakes you make); and (3) you have some freedom to choose what activities you will perform to acquire experience.

Here are the levels you may achieve, together with their semester grade and the number of XP required for each:

Level	XP	Semester grade
Dungeon Master	1300	A+
Wizard	1200	A
Sorcerer	1100	A-
Valkyrie	1000	B+
Warlock	900	B
Paladin	800	B-
Ranger	700	C+
Cleric	650	C
Conjurer	600	C-
Barbarian	550	D+
Rogue	500	D
Bard	450	F
Archer	400	.
Swordsman	350	.
Squire	300	.
Guild Master	250	.
Artisan	200	.
Apprentice	150	.
Merchant	100	.
Footsman	50	.
Commoner	25	.
Non-Adventurer	0	.

Writing assignments

This is a Writing Intensive (WI) course, and hence will involve writing English text: sometimes responsive essays on Computer Science topics, sometimes technical artifacts such as requirements descriptions or API documentation. The most important thing to emphasize right away is that in our discipline **writing well does actually matter and is something you should actually care about**. This comes as a surprise to many CPSC students. But our discipline is full of ideas, and ideas must often be expressed in writing.

I will spend a great deal of my time this semester helping you become better writers. It is a nuanced skill that is worthwhile to acquire. In the recent past, however, I have received an alarming number of outright ungrammatical submissions

from UMW students. They have so many syntactical errors that they're not really ready for a deeper analysis. Because of this, I will institute the **three strikes policy** this semester: I will read your submission and correct it only until I encounter the *third* grammar mistake, at which point I will stop and return it to you. You are always welcome (and encouraged) to fix your mistakes and resubmit it.

Earning XP

There will be opportunities to earn XP throughout the course. Some of these will be spontaneous as the mood strikes me. Some may be in response to impressive things I see you do. Some will be available from in-class team activities. The following opportunities, however, are *guaranteed* to be available to you:

Individual learning opportunities:

Activity	Possible XP	Due date
Faceboard	10	Jan 18
*Responsive reading #1	50	Jan 22
Basic Java program – Star Wars name generator	50	Jan 25
Background project research	25	Jan 28
Canvas quiz #1	30	Jan 31
Zork I (Basic scaffolding)	100	Feb 8
*Responsive reading #2	50	Feb 15
Canvas quiz #2	30	Feb 20
Zork II (Persistence & hydration)	100	Feb 22
*Use case design – Zork dungeon builder	50	Feb 28
Canvas quiz #3	30	Mar 1
Canvas quiz #4	30	Mar 31
Canvas quiz #5	30	Apr 17
Canvas quiz #6	30	Apr 26
Zork individual dungeon	50	Apr 30
Final exam	200	May 1
Various and sundry others	varies	varies

Group activities:

Activity	Possible XP	Due date
Zork III (Items & inventory)	100	Mar 15
Zork++ game engine – UML design	100	Mar 25
*Zork++ game engine – API documentation	100	Apr 1
Zork++ game engine – implementation (part 1)	100	Apr 17
Zork++ game engine – implementation (part 2)	100	Apr 24
Various and sundry others	varies	varies

("*" denotes writing-intensive assignment, with opportunity for revision and resubmission.)

You will be assigned to **teams of three** in mid-February. (The assignment of students to teams is at my discretion, but if you feel strongly about someone you do not want to work with, you may mention this preference to me.)

All group activities will be performed as a team, and **XP will be awarded uniformly to all team members.**

Procrastination

Procrastination will kill you. Do. Not. Do. It.

If you want to succeed in college, your career, and life, **you must overcome the tendency to procrastinate.** Unfortunately I'm not sure what the cure is; the remedy probably depends on the individual. But in this class particularly, it is vital to get over the hump. There are going to be long stretches (2 weeks or more) during which you are expected to be working towards the next project milestone. Leaving the whole project until the day or two before the deadline is a recipe for disaster. **You must figure out a way to get yourself (and your teammates) to work regularly and consistently over the time period allotted.** If you suspect you may have trouble with this, please come to office hours to discuss it and I'll be happy to help you figure out what intermediate deadlines you might set for yourself to overcome this problem.

Basis for determining mid-semester reports

If you haven't acquired 200 XP by fall break, I'll submit a "U" for your mid-semester grade. Please don't hesitate *at all* to come talk to me about this so we can figure out how you can do better in the course.

Guidelines for class participation

I believe that students learn best when they participate wholeheartedly in all aspects of the learning process. Hence while your grade will not be partially determined by any "class participation score" *per se*, it is very much to your advantage, and very much recommended, that you join in during class discussions, ask questions, and make comments.

How to reach each other

I will post announcements to the course website often, so be sure to subscribe to its RSS feed and check it in your feed reader at least once a day! Also, make sure to check your UMW e-mail every day!

To get a hold of me, come to office hours, see me after class, or e-mail me at `stephen@umw.edu`. I'm happy to hear from you any time!

Road map

Week	Topics
1	Motivating OOA&D; the development environment
2	OO principles: classes and objects; memory diagrams
3	Encapsulation; UML class diagrams; exceptions
4	The Singleton pattern; UML sequence diagrams
5	Persistence & hydration; polymorphism and inheritance
6	Polymorphism and inheritance, cont.; Java odds 'n' ends
7	The Factory pattern; team software development
8	Discovering the design
9	Software requirements: Use Cases
10	Documenting an API; design patterns
11	Design patterns, cont.
12	Design patterns, cont.
13	Introduction to parallelism and concurrency
14	Introduction to parallelism and concurrency
15	Introduction to parallelism and concurrency