CS 474: Object-Oriented Languages and Environments
Spring 2020

Course Instructor:

Name: Mark Grechanik
Email: drmark@uic.edu
Web Page: http://www.cs.uic.edu/~drmark
Office Hours: SEO, Room 1103, F, 4:30PM-6:00PM or by appointment

Class Meets: TTh 11:00 am - 12:30 pm in the TBH 180F

TA: Mr.Mohammed Siddiq at msiddi56@uic.edu. He will announce their office hours via Piazza.

Course Web Address:

All course materials will be made available to you via Piazza. It is your responsibility to set up your account with the university Blackboard (BB) and Piazza and use it to read course announcements, view your grades at BB and participate in discussion groups at Piazza. If you have problems using the BB or Piazza, then please use the corresponding help desk to resolve them.

Prerequisites:

- Firm understanding of the software development steps. Knowledge of source control systems (i.e., Git) and integrated development environments is required. We will use IntelliJ for homeworks and projects in this course, so each student must obtain its academic free version and get familiar with it. We will use Gradle and Sbt to build and run projects. All submissions will be done via Bitbucket.
- Students are expected to have taken CS 340 Software Design with a grade of C or better or its equivalent from other universities.
- You should have access to a computer with at least 16Gb of RAM with at least a double-core 2GHz CPU and a 200GB+ free space on its hard drive to which you should have the administrative access. In general, as a CS student taking an programming language course like this one you need a more powerful laptop, which you could rent or purchase. NOT having a sufficiently powerful laptop of workstation is akin to NOT having a good quality violin for a practicing violinist! With starting salaries in our area in the range of $90,000, purchasing a $3K laptop is not a luxury, it is a necessary investment in yourselves.
- It is important to come to lectures rested and prepared to participate in discussions. Lecture attendance is optional, however many students attend because they can interrupt the lecture to clarify the material directly from the lecturer. Therefore, if the student does not follow the lecture, there is little point in her/his sitting in the classroom. Some students suffer from micro-sleep (MS) defined as “a temporary episode of sleep or drowsiness which may last for a

fraction of a second or up to 30 seconds where an individual fails to respond to some arbitrary sensory input and becomes unconscious. MSes occur when an individual loses awareness and subsequently gains awareness after a brief lapse in consciousness, or when there are sudden shifts between states of wakefulness and sleep. In behavioral terms, MSes manifest as droopy eyes, slow eyelid-closure, and head nodding.[1]" MS will interfere with your ability to Remember to rest well before you come to lectures.

If you have not met these prerequisites, you must see me.

Required Textbook(s):

Java Language and Virtual Machine Specifications at
https://docs.oracle.com/javase/specs/index.html

Programming in Scala: A Comprehensive Step-by-Step Guide, Third Edition by Martin Odersky, Lex Spoon, Bill Venners. Available on SafariBooksOnline.

Functional Programming in Scala by Paul Chiusano (Author), Rúnar Bjarnason. Available on SafariBooksOnline.

Michael L. Scott, Programming Language Pragmatics, Third Edition, Morgan Kaufmann, 2009. PDF is available.

Robert Harper. 2012. Practical Foundations for Programming Languages. Cambridge University Press, New York, NY, USA. PDF is available.

Also, I will give you reading assignments that include research papers, articles, and handouts. You will be able to integrate your class notes with copies of slides that I use in class and with pointers to papers and web sites relevant to the material discussed in class, all of which I will post on the blackboard.

Optional Textbooks:

R P Nederpelt; Herman Geuvers, Type theory and formal proof : an introduction. Cambridge ; New York : Cambridge University Press, 2014.

Course Content:

Course Goals:

The course will consist of readings, quizzes, two exams, and a final project. The goals of the course are to:

[1] https://en.wikipedia.org/wiki/Microsleep

- Understand the theory of objects;
- Learn principles of OOP;
- Cover various languages and environments for OOP, and to
- Enable students to design OO applications using various tools and methods with popular distributed OO plugins, toolkits, and frameworks.

Course Topics:

1. Fundamentals: language, compilation, code generation, runtime. Types of languages. Popularity, TIOBE and IEEE indexes. Language specifications.
2. Variables, procedures, bindings, types. Statements and expressions. State.
3. Abstract data types, procedural abstraction, and object-orientation. Classes and objects. First-class objects.
4. Object-oriented modeling. Subtyping. Ad-hoc polymorphism. Dispatch.
5. Reuse and inheritance. Design Patterns.
6. Intro to functional programming. Functions as objects.
7. Lambda-calculus.
8. Multi-paradigm languages: Scala and Python.
9. Interfaces and mixins. Traits.
10. Errors. Exceptions. Options.
11. Reflection.
12. Parallelism and Concurrency. Asynchronous programming with futures.
13. Type systems.
14. Parametric polymorphism. Using generics to achieve reuse. Wildcards.
15. Phantom types.
16. Actor programming with Akka for building REST web services.
17. Reactive functional programming. Streams. Lazy evaluation.
18. Aggregation functions.
19. Type programming. Enforcing design pattern constraints with types.
20. Type inference. Implicit types in Scala.
21. Javascript and JQuery
22. Functors and monads.
23. Structural and path-dependent types.

Grading:

- Quizzes: 10%
- Homeworks: 20%
- Midterm: 20%
- Final: 25%
- Project: 25%

Warning: in the previously taught CS 474 classes students reported allocating as low as three hours of homework time to achieve the grade of C or better to over 20 hours of homework time per week. Different students have different backgrounds and it required them to allocate additional time to catch up with some concepts. For example, students

who already learned how to use Git and IDE engaged with this course faster whereas students who never used an IDE had a more difficult time. In many cases, there are instructional videos on the Internet that demonstrate to you how to use development tools step by step (e.g., this video shows how to use the entire functionality of IntelliJ). We will concentrate only on core language material in class and we will not waste the precious time on the demonstration of the basic tool functionality. You should estimate the effort required from you to learn in this class and to complete your homeworks and plan your time accordingly. Also, Piazza is available to ask (anonymous) questions whenever you stumble upon a problem – an average response time in my classes is under one hour on Piazza. So plan you time carefully - your instructor will NOT reduce your load or delay the deadline because you signed up for more classes than you can handle.

General:

- You are responsible for knowing the academic calendar and adhering to its deadlines, including but not limited course add/drop dates and payment due dates. All differences between information in this syllabus and the official UIC academic calendar should be resolved in favor of the calendar. Please notify Prof. Grechanik about the differences if you discover them.
- Some lectures will start with a 5-10 min quiz without prior announcements.
- Bonus points will be given in class for active participation in lectures.
- Lecture attendance is not required, you are responsible for learning.
- Eating, sleeping, browsing the Internet in the classroom are permitted as long as these activities do not distract other students and me and do not violate UIC policies.
- Our class does not meet during the Thanksgiving break.
- Please check the academic calendar for the last day to complete late registration; last day to add a course(s) or make section changes; last day to drop individual courses via Student Self-Service without receiving W (Withdrawn) grade on academic record. Make sure that you know the last day to submit Withdraw from Term request via Student Self-Service and receive 100% cancellation of tuition and fees.
- Midterm will be held in our usual classroom, during the regular lecture hours on Thursday, March 12, 2020.
- No classes are held during the spring break week: March 23–27, 2020.
- The day of the final examination will be announced later – the university will schedule it.[2]
- You must attend the midterm and final exams and complete your project to satisfy the baseline requirements set by me to be considered for a grade of C or better.
- No makeup exams and quizzes will be permitted unless authorized by the CS (under)graduate advisor or the department head.
- All homeworks and the course projects should be submitted via Bitbucket. Submissions by email and BB will not be accepted.

[2] The dates for the midterm and the final are subject to change. Stay tuned for announcements.

- The course project submission date is Wednesday, May 6, 2020 at 9:00PM via Bitbucket. Details of the project submission will be announced later. Failure to submit your project by the deadline shall result in losing your project grade.
- You can discuss your grades with me only if you failed to resolve the issue with the TA and if and only if you ask questions about specific points you lost in your quizzes, exams, and the project.

Examples of homeworks:

HW1: using the Eclipse Java parser, create the AST of an open-source Java application and compute Halstead complexity measures enumerated in this wikipedia post https://en.wikipedia.org/wiki/Halstead_complexity_measures.

HW2: create a java agent, unit tests for a chosen Java application and compute the runtime coverage of the application that is executed with these unit tests
Help: https://www.javacodegeeks.com/2015/09/java-agents.html;
http://zeroturnaround.com/rebellabs/how-to-inspect-classes-in-your-jvm/;
http://marxsoftware.blogspot.com/2011/12/estimating-java-object-sizes-with.html;
http://blog.javabenchmark.org/2013/05/java-instrumentation-tutorial.html

HW3: using java reflection, create a test simulation class.
ActualJavaClass obj = TestSimulator(ActualJavaClass.class).whenInvoking("method", p1, p2, p3).return(retval)
assert(obj.method(p1,p2,p3) == retval)

HW4: write a program in scala that takes a top-level directory path as its input and it calculates statistics for all its subdirectories. Use filter and fold functions.

HW5: using type programming in scala, create an implementation of the house builder simulator that not only ensures at compile time that all build methods are called, but also that they are called in a specific order. Document your constraints and the design.

HW6: design a system that can be modeled using types of all variances. explain your design choice, create a model of the system and implement it.

Example of a course project:
Using an open source dataset or live data stream (see some suggestions below), design and implement a functional reactive application with Akka using Scala or Java to process the incoming data, extract some of its attributes, save them in the local datastore, aggregate values of these attributes, feed them programmatically in Weka, a machine-learning software to learn patterns in data, and present results. The project should contain detailed documentation that explains the requirements, the semantics of data, OO modeling and design, and you should use functional reactive programming to implement

your application. For open-source data you can search on the Internet for available resources. You can use https://github.com/caesar0301/awesome-public-datasets#social-networks to bootstrap your search. For example, OpenSteetMap dataset contains open-source map data[3], and DBLP dataset contains information about many articles, conferences and journals where these articles were published, and their authors[4].

Fairness:

It is a fundamental right of each student who takes this course (and other courses) to be treated fairly. Especially, in a large class we have students with different backgrounds, family/work situations and various personal restrictions: there are students who are single mothers and fathers with full-time jobs who need to care for their children, military and other civil servants who risk their lives as part of their jobs, and students with disabilities who need special accommodations, to name but a few. My job is to ensure that this course proceeds smoothly and it is your responsibility to do your assignments and attend lectures and perform all course duties by organizing all your activities to ensure that all submissions are done on or before the deadlines and you take all exams in a pre-arranged manner according to the schedule.

Fairness is violated when you ask me to allow you to submit your assignments late or make up for an exam for you, specifically, whatever reason. It means that if I allow you to do it, I would treat all other students unfairly, because they managed to organize their lives to postpone other important tasks, e.g., to spend time with their children, at some nontrivial expense to them (e.g., hire a babysitter) to complete the assignment on time. Unfair treatment of students breeds resentment and prevents us from having a healthy learning environment.

The only reason to postpone the submission deadline can be done by offering solidly documented and verifiable emergency that occurred to you, e.g., the verifiable documentation of bodily injury from an emergency room with the waved HIPAA rights, so that our department personnel can obtain an independent verification of this emergency. Once a student gave me a jury service notification where the date of his appearance fell on the midterm. I wrote a letter for him explaining that he had a midterm on the date, and his jury appearance was rescheduled. Please take the issue of fairness close to your heart when you think about asking me to postpone a submission deadline.

Emergencies and Deadlines

> As the deadlines for homework/course project and the exams get closer, I get emails from some students about their visits to doctors/emergency rooms with requests to postpone the deadlines for them. Here are some general rules. To accommodate you, I need to go through a process of verifying this situation. This process is the same for all students and it involves a few steps. If you provide a

[3] http://wiki.openstreetmap.org/wiki/Planet.osm
[4] http://dblp.uni-trier.de/xml/

doctor's note to me, be aware that I can contact the doctor or hospital to verify if you were seen on the date(s) in question. If your doctor is unable to verify your visit without a signed HIPAA authorization form, you may need to complete a HIPAA authorization for release of health information form in order for me to verify that you were seen by the doctor on the date in question.

In general, I do not need to see a document that describes a medical condition that brought you to a doctor. The attending doctor, however, should provide official evidence that a student's condition prevented the student from taking a given exam, including potentially via the UIC dean of students. Once this verification is done, we can discuss the accommodations based on its outcome.

Know Your Computing Environment and Learn to Debug It!

Many students in this class will experience difficulties setting up the development and deployment environments for your homeworks and the course project. I would like to offer a few tips to alleviate this problem, and I start with an analogy to compare each of us with a violinist or pianist or some other musician. Or we can use an analogy with a surgeon.

Imagine a violinist who complains that s/he spent a lot of time fine-tuning her/his instrument because s/he didn't know how to tweak different knobs on her instrument. Or a surgeon who complains that instead of cutting flesh and replacing a bad heart valve s/he had to figure out how to set up and use tavi-midcab retraction system on a patient's heart. You can reply that musicians must learn to fine-tune instruments before playing complex pieces and surgeons should not be allowed to cut flesh until s/he learns how all instruments work and how to use them. Using this analogy, it will leave us with the following question: how much cloud computing material we could cover if lectures are spent on showing how all those tools work in practice.

As computer science students, you should know your hardware (your laptop or your workstation) and all the software that runs on it. You should also learn about well-known tools that automate various tasks. Just like a violinist knows her violin and constantly experiments with its sounds and knobs (it is a right term?), you should know your tools and constantly learn about new ones.

Given the number of tutorials and discussion board and youtube videos and safaribooksonline database, it is not justifiable for professors to use the classroom time to go over the tools and explain basic and well documented operations on the Internet, e.g., how to set a breakpoint or how to debug an external Java process in IntelliJ. Understanding how source control systems work is important. In one situation, a student pushed the code into the master repo mistakenly and the other student pulled this code into his local repo overwriting his local settings. Both situations resulted from the lack of attention, lack of consideration, mental laziness, and the desire to cut corners instead of figuring out how Git works in

detail. As a result, one student lost a few days of work, so he paid with his time for the light attitude towards the complex systems.

It is a part of the learning experience where you develop and refine your abductive reasoning skills. You observe symptoms, recall the set of actions that led to the appearance of these symptoms and then you create hypothesis of what action can cause these symptoms. Then you think of experiments where you modify the system and see how the symptoms change. Slowly, you will converge to a solution.

In case of the example above, this approach would work as the following. The student will delete the non-working project and create a new virgin project and add libraries and write some basic code. Once he sees it is working, he will pull the repo and add the working code to the repo and push it. Then he will pull the code fromt he repo again in a new dir and experience the problem. Now, there are two scenarios: the virgin project works, and the one merged with the repo doesn't. The same idea applies to the other problems you have experienced.

Of course, to do that you need to apply your brain to design a systematic approach to your engineering complex applications. If you attention span is short and you are looking for shortcuts, you will end up very frustrated in this course.

Selecting Teammates for the Course Project

Your course includes a course project. Regretfully, every time I teach this course I receive complaints shortly before project submission deadline that some team members didn't do anything and the working team members want justice by excluding the freeloaders from the team. This is the reason that I am sending this message - to warn you about possible dangers of forming teams and how to ensure that your course project work is productive and enjoyable.

1. Ask your prospective teammates to give you read access to their homework repos and in response you give them access to yours. This way you can see their previous deliverables and gain confidence in their skills.
2. Define project milestones early on and determine who is responsible for what. Do it in writing via email or set up a project in basecamp (you can get an academic license).
3. Document progress. If your teammates do not respond to your email inquiries about their progress on milestones, notify me after two or three attempts to contact your teammates by email. Include your email exchange and CC to your nonresponsive teammates. With this evidence, I can intervene and break up the team if your teammates did not deliver as agreed. To make it effective, you need to set up milestones early in the project lifecycle and keep the documentation to prove the pattern of nondelivery.

Interviews:

On more than a few occasions, I received emails from students or verbal requests to delay quiz/homework submission/exam/... because these students had scheduled interviews or they planned to attend career fairs (with possible interviews there). In some cases, these requests were given to me in a tone that suggested that since a company like Google selected a student for an interview, I should have acknowledged the greatness of the process and humbly step aside with my pestering course assignments. Responding to every request like this takes time and effort, so I want to do it in this syllabus.

You are invited to interviews, most likely because you are studying at UIC CS and taking courses and doing assignments, so your interviewers think that you stand out from the crowd because of your education. They will respect that you take your courses seriously and they will accommodate you gladly, since it shows that you value your commitment to your current projects more than your hunt for future projects. When I asked my friend, a manager at Google about students postponing interview because of the coursework, this is what she replied to me: "In my own experience the recruiter would ask for the student's availability before schedule (sic) an interview. Even after that the date is not carved in stone. Unless the student is facing deportation by USCIS without a job offer by certain date, I don't see a pending interview as a reason to ask for submission deadline extension."

Cooperation policy:

I encourage you to discuss the problem sets and programming assignments with your colleagues. I welcome discussions of possible interpretations of questions, solution approaches, and relevant technical details. You are also welcome to use existing public libraries in your programming assignments. Such activities qualify under approved collaboration practices and you are welcome to take advantage of them.

Note that cooperation is not the same thing as cheating. It is OK to ask someone about the concepts needed to do homework or project assignments. However, ***copying*** other people's code or solution sets is strictly prohibited. The quizzes, project assignments, and exams must be the work of students turning them in. Students who violate University rules on scholastic dishonesty are subject to disciplinary penalties, including the possibility of failure in the course and/or dismissal from the University. Because such dishonesty harms the individual, all students, and the integrity of the University, policies on scholastic dishonesty will be strictly enforced. Acts that exceed the bounds defined by the approved collaboration practices will be considered cheating.