COM 297, Modern Programming Languages

American University of Central Asia
Software Engineering Department

1 Course Information

Course Code
COM 297

Course ID
5679

Prerequisite
COM-117, Programming II
or
COM-119, Object-oriented Programming

Credits
6

Time and Place
Lecture: Tuesday 15:35–16:50, Lab Room 432
Lab: Thursday 15:35–16:50, Lab Room 432

Course Materials
`https://github.com/auca/com.297`

2 Contact Information

Professor
Dmitrii Toksaitov
toksaitov_d@auca.kg

Office
AUCA, room 315

Office Hours
By appointment throughout the work week (write to your professor to make an appointment)

3 Course Overview

The Modern Programming Languages course will familiarize students with contemporary programming languages currently popular on the market and demonstrate the most common types of applications that people build with those languages. The course will cover the fundamentals of Kotlin, Swift, Dart, JS with TypeScript, Python, Ruby, Go, Rust, C#, Erlang with Elixir, F#, and Haskell. Students will learn the syntax and semantics of the languages, the structural, procedural, object-oriented, and functional paradigms. Students will create apps of different types popular to be developed by the language at hand to practice. We will write a simple mobile app in Kotlin, Swift, and Dart in this course. We will create React front-end apps with TypeScripts. We will build back-end server applications with Ruby, C#, and Go. We will write a driver in Rust for the Linux kernel. We will do a bit of scientific calculations and machine learning interactively with Python and Jupyter Notebooks. We will create a distributed network application to handle thousands of connections with Erlang and Elixir. In the end, we will try to implement several standard data structures and algorithms in the world of functional programming with F# and Haskell.

As a result, students should be able to research and analyze the functioning of the information technology systems, improve their skills using programming languages for software design, development, and maintenance in accord to the goals of the AUCA Software Engineering department and the 510300 IT competency standard (including competency elements ОК 1–7, ИК 1–7, ПК 1–15).

4 Topics

- Week 1–2: Introduction (6 hours)
- Week 3: C# (3 hours)
- Week 4: Ruby (3 hours)
- Week 5: Go (3 hours)
- Week 6: Kotlin (3 hours)
- Week 7: Swift (3 hours)
- Week 8: Dart (3 hours)
- Week 9: JS with TypeScript (3 hours)
- Week 10: Python (3 hours)
- Week 11: Rust (3 hours)
- Week 12: Erlang with Elixir (3 hours)
- Week 13: F# and Haskell (3 hours)

5 Assignments and Exams

5.1 GitHub Checkpoints

Students will have to maintain a personal private GitHub repository with all their works shared with their instructor. Students have to periodically commit and push a specific number of lab and project solutions as told by the instructor. The instructor will check the assignments and give points for the accomplished work.

5.2 Labs and Projects

Students will have a number of laboratory tasks and have to finish several course projects. Students will have to defend their work to the instructor during midterm and final examination sessions.

6 Course Materials, Recordings and Screencasts

Students will find all the course materials on GitHub. We hope that by working with GitHub, students will become familiar with the Git version control system and the popular (among developers) GitHub service. Though version control is not the focus of the course, some course tasks may have to be submitted through it on the GitHub Classroom service.

Every class is screencasted online and recorded to YouTube for students' convenience. The course Zoom link is `http://bit.do/auca-com-297`. YouTube links can be found on the course repository at `https://github.com/auca/com.297`. An ability to watch a class remotely at any time MUST NOT be a reason not to attend at least the online Zoom session. Active class participation is necessary to succeed in this course.

7 Software

Students are recommended to install the following software on their machines.

- Git: `https://git-scm.com`

The compilers, assemblers and debuggers will be available on the remote course server.

8 Reading

Be prepared to read and use official language guides and documentation on this course on your own to do programming exercises.

9 Grading

9.1 GitHub Checkpoints

Your instructor will announce a periodic review of your work. You will be awarded up to the following number of points for such checks.

- Labs (30%)
- Project (30%)

9.2 Exams

- Midterm Exam (20%)
- Final Exam (20%)

9.3 Totals

- 100% is formed from the GitHub submissions (60%) and the two exams (40%).

9.4 Scale

- [92%–100]%: A
- [85%–92)%: A-
- [80%–85)%: B+
- [75%–80)%: B
- [70%–75)%: B-
- [65%–70)%: C+
- [60%–65)%: C
- [55%–60)%: C-
- [50%–55)%: D+
- [45%–50)%: D
- [40%–45)%: D-
- Less than 40%: F

10 Rules

Students are required to follow the rules of conduct of the Software Engineering Department and the American University of Central Asia.

10.1 Participation

Active work during the class may be awarded with up to 5 extra points at the instructor's discretion.

Poor student performance during a class can lead to up to 5 points being deducted from the final grade.

Instructors may conduct pop-checks during classes at random without prior notice. Students MUST be ready for every class in order not to lose points. Students absent without a good reason from such classes with graded work will also lose points unless it is force-majeure circumstances. Instructors must be notified in advance about why a student is absent not to lose points.

10.2 Questions

We believe that a question from one student is most likely a question that other students are also interested in. That is why we encourage students to use the Canvas online discussion board to ask questions in public that other students can see and answer. We discourage students from asking questions through E-mail. If it is a private matter, write direct messages to your instructor there on Canvas too. We will not be answering most E-mail messages this semester (unless it is a severe emergency) to consolidate all the course correspondence in one place on our LMS (Learning Management System).

Do not post the complete source code for any task on the Canvas discussion board. You will get zero for that work for any such public post. Do not ask generic questions about your code to know why it does not work. Please spend some time thinking about your code, debugging it.

10.3 Late Policy

Late submissions and late exams are not allowed. Exceptions may be made at the professor's discretion only in force-majeure circumstances. If you got ill, got severe personal issues, got problems with your computer or the Internet, you MUST notify me at least 24 hours in advance. Otherwise, we will not give you an extension. We will consider that you were procrastinating until the very last day. We will also not be giving more than one emergency extension throughout the course.

Six hours before the deadline for any work on the course, instructors will go into a silent mode. No questions will be answered about the work that has to be submitted, no requests to have office hours will be considered. Usually, it will be Saturday and Sunday (which are not working days for us anyway). However, at any other work time before the deadline, we will try our best to answer your questions and help you through Zoom or in our office.

10.4 Exam Ceremonies

Students MUST follow exam ceremonies. It means they MUST prepare task list forms with all points appropriately calculated. They MUST submit them correctly. They must bring task list forms to the exam. Failure to do so will result in lost points.

Throughout your career, you will have to work with various supporting documents (contracts, timesheets, etc.). It is a good idea to start learning to work with such documents accurately early. We will remove points for not following these rules or even refuse to accept your exam defense. We will also give zero for not following the strict exam timing rules.

10.5 Incomplete

As with late submissions, the grade *I* may be awarded only in exceptional circumstances. The student must start a discussion on getting the grade *I* with the instructors in advance and not during the last week before the final exams.

10.6 Academic Honesty

Plagiarism can be defined as "an act or an example of copying or stealing someone else's words or ideas and appropriating them as one's own". The concept of plagiarism applies to all tasks and their components, including program code, comments, documentation, abstracts, reports, graphs, statistical tables, etc.

The following are examples of some common acts of plagiarism:

1. Representing the work of others as their own
2. Using other people's ideas or phrases without specifying the author
3. Copying code snippets, sentences, phrases, paragraphs or ideas from other people's works, published or unpublished, without referring to the author
4. Replacing selected words from a passage and using them as your own
5. Copying from any type of multimedia (graphics, audio, video, Internet streams), computer programs, graphs or diagrams from other people's works without representation of authorship
6. Buying work from a website or from another source and presenting it as your own work

In addition to being unethical, this indicates that the student has not studied the given material. Tasks written from somewhere for 5% or less will be assessed accordingly or will receive a 0 at the discretion of the teacher. If plagiarism is more than 5%, the case will be transferred to the AUCA Disciplinary Committee.

In this course, teamwork is NOT encouraged. The same blocks of code or similar structural pieces in separate submissions will be considered academic dishonesty, and all parties will get zero for the task.

Students are not recommended to memorize lab and project code before exams, as this is a difficult and inefficient way to learn; and since practice exams may consist of open questions designed to test a student's analytical skills, memorization invariably leads to the fact that the answers are inappropriate and of poor quality.