COURSE SYLLABUS STEM Ed Abroad Program

Course Title: Introduction to Java Programming
Course Semester: Fall
University and Country: Adam Mickiewicz University; Poznan, Poland
Number of ECTS: 6 (lecture) and 2 (laboratory)
Course Designations for Transfer Credit: CSC216 (NCSU)

Content: Laboratory with experiments designed to provide fundamental concepts and an overview of Electrical and Computer Engineering specialization areas including Analog Electronic Circuits, Electric Power, Communication Systems, and Signal Processing. Introduction to standard laboratory equipment including power supply, multimeter, function generator, oscilloscope and spectrum analyzer.

Pre-requisites: Calculus III, Physics I and a basic course in Electrical Engineering.

Aims: CSC 216 is the second course in computing, intended for majors and students in the Computer Science Certificate Program. Emphasis is placed on interpretation of inductive definitions (functions and data types); testing strategies; specification and implementation of finite-state machine; encapsulation; polymorphism; inheritance; class invariants; and resource management.

This is a course on the fundamentals of computer science and programming using Java. Students taking this course are expected to have an understanding of loops, conditional logic, objects, classes, file I/O, arrays, and the basics of Java GUIs (swing and/or AWT).

Upon satisfactory completion of this course, you should be able to:

- Describe the utility of inheritance, abstract classes, interfaces, and polymorphism in object-oriented systems, and design and implement programs that use these language features.
- Identify the phases of a simple model of the software life cycle and employ these phases in developing software.
- Describe basic design modeling techniques, including UML class diagrams, and indicate how and when to use them.
- Identify and compare the basic kinds of software testing, describe when to use each method, and design and implement test code.
- Navigate and extract information from the Java API and employ the Javadoc tool to construct internal documentation of source code.
- Design and implement a finite state machine.
- Identify when recursion is useful and design and implement recursive algorithms and simple recursive data structures.
- Construct and use a stack, queue, array-based list, linked list, and a simple binary tree.

- Proficiently use an IDE (Eclipse) and version control system (GitHub).

Learning outcomes:

The goals for this course are to ...

- learn to design programs with classes that work together with maximum cohesion and minimum coupling,
- implement programs that use abstract classes, interfaces, inheritance, polymorphism, and recursion,
- understand the software lifecycle, basic design using UML, finite-state machines, and basic software testing,
- be capable of working with the basic Java APIs,
- use Javadoc to automatically construct source code documentation, and
- learn to use basic data structures like stacks, queues, linked lists, and binary trees.

Recommended Books: *Big Java, 2nd Edition,* C. S. Horstmann, J. Wiley, Inc., http://www.horstmann.com/bigjava.html

Notes for the online section of CSC 216 are located here. Readings from these notes will be assigned in advance. You are responsible for reading and studying these notes

Instructors: XXX with consultation by Dr. Sarah Heckman (NC State University).

Grading System and Percentage Contribution

Your final grade will be based on the following scheme:

Work	Total
Final Exam	**16%**
Guided Project 1	3%
Guided Project 2	3%
Guided Project 3	3%
Labs (12 different labs)	10%
Project 1: Black Box Test Plan, Design Document	1%
Project 1: Solution code and JUnit tests	12%
Project 2: Black Box Test Plan, Design Document	2%
Project 2: Solution code and JUnit tests	12%
Project 3: Black Box Test Plan, Design Document	2%
Project 3: Solution code and JUnit tests	12%
Test 1	12%
Test 2	12%

You cannot make up missed tests or exams without an official university excuse. Furthermore, we will not accept late programming assignments (except as described below) without an official university excuse.

Code and Submission Tools. You must use Eclipse for project/code development and NC State's GitHub for submission of your work. Eclipse is an industrial-strength Integrated Development Environment (IDE) that incorporates many of the tools that you will use this semester. GitHub is a web application around Git, which is a version control system. You will submit all non-test/non-exam work for grading to instructor supplied repositories in NC State's GitHub. This includes both code and ancillary documents. The initial labs and guided projects have copious information/tutorials on how to use Eclipse and GitHub.

Guided Projects. Guided Projects will introduce you to several tools to facilitate learning software engineering skills, specifically Eclipse, JUnit, GitHub, Jenkins, FindBugs, PMD, CheckStyle, Subversion, and EclEmma. There are three guided projects, with the second building on the first and the third building on the second.

Labs. There are twelve labs for the semester. Each deals with a specific course topic. Most labs are due the week following their presentations in the lecture material. The due date/time for each lab is given in the syllabus. All submissions will be to GitHub, and all will be graded automatically by Jenkins. In calculating your lab average, we will drop your lowest two lab grades.

Programming Projects. There are three full programming assignments. Each assignment is divided into two parts. Part 1 is devoted to design and black-box test plans. Part 2 is devoted to implementing the design that we supply and unit testing your code. For each project, we supply the instructor design immediately after the Part 1 due date. You have approximately 2 weeks from that time to submit your code for Part 2.

You must electronically submit all project deliverables by the published deadlines and follow the specified formats, submission instructions, and naming conventions. All projects (except Design Proposals and Rationales for Part 1) will be accepted up to 48 hours late. You will lose 1 point every 2 hours the project is late, up to 24 points. No submissions will be accepted after the 48 hour late window without a university excused absence. No late submissions will be accepted through email.

Feedback. Jenkins is a continuous integration program that will automatically compile and test your programs (both with your tests and the teaching staff tests) and provide style feedback. Your grade for Part 2 of each project as well as each lab will be calculated from the last GitHub submission you make before the deadline (even if Jenkins runs after the deadline for that submission) plus additional points for acceptance tests, FindBugs issues, and other related rubric items. The style deductions as derived from Jenkins feedback may be modified by the teaching staff when they manually inspect your code and comments.

Working Individually. All projects must be developed individually rather than as group projects. All programs are to be you own work. See the "Academic Integrity" section of the syllabus for further details.

Grade Appeals. If you feel an assignment was graded improperly, write a request for regrade and explain the basis of your belief. First discuss the grade with the TA who graded the assignment. If you are still unsatisfied with the answer, submit the assignment to the instructor for a regrade. All regrade requests must be submitted to the instructor no later than 2 weeks after the assigned was returned to you. All grade requests for Project 3 must be made at least one day prior to the final exam.

Time and Effort. You are expected to view the taped lectures, read the online course lecture material, and do the non-graded exercises. You should remain current with the course schedule. In addition to reading the html lectures and viewing the videos, expect to spend on average 10 to 16 hours per week preparing and working on graded assignments. In some weeks, especially those around project deadlines, you may spend more than that amount of time. Please plan ahead and use your time wisely. Do NOT wait until the last minute to complete programming projects!!!

AMU Grading system and scale

The grading system used at Adam Mickiewicz University, whose name is abbreviated as AMU or UAM, is as follows:

Tests, exams, homework assignments grading scale

5	100%-91%
4+	90%-86%
4	85%-76%
3+	75%-71%
3	70%-60%
2	59% and less

This translates into the following ECTS (European internationally recognized system) grading scale:

ECTS Grade	AMU grade	Definition
A	5.0	EXCELLENT – outstanding performance with only minor errors
B	4+ / 4.5	VERY GOOD – above the average standard but with some errors
C	4.0	GOOD – generally sound work with a number of notable errors
D	3+ / 3.5	SATISFACTORY – fair but with significant shortcomings
E	3.0	SUFFICIENT – performance meets the minimum criteria
FX	2.0	FAIL – some more work required before the credit can be awarded
F	2.0	FAIL – considerable further work is required

Hours: 3 Lecture hours hour per week. The laboratory component consists of XXX topics listed below. Each laboratory has a duration of 2.5 hours.

COURSE SYLLABUS STEM Ed Abroad Program

Course Lecture Topics:

This course offers an introduction to basic programming concepts using the Java object-oriented programming language. We will cover the fundamental stages that occur during the creation of computer programs, including design, implementation, testing, and documentation. Object-oriented features of Java like abstract classes, interfaces, inheritance, and polymorphism will be presented. Techniques to design, test, and document your source code modules will be explained. We will also begin an introduction to fundamental data structures like stacks, queues, linked lists, and binary search trees.

Below is a tentative course schedule. Please note that time frames and topics will be confirmed in class and are subject to possible changes.

Week	*Topic*	*Chapter*
1	Introduction	1-8
2	Class Design	9
3	Testing and Debugging	10
4	Interfaces and Polymorphism	11
5	Inheritance	13
7	Exceptions	15
8	System Design	17
9	Recursion	18
10	Linked Lists	20
11	Stacks and Queues	21
12	Finite State Machines	Notes
13	Binary search trees	21
14	Selection, insertion sort	19