COMPSCI 121: An Introduction to Problem Solving with Computers

Course description

COMPSCI 121 provides an introduction to problem solving and computer programming using the Java programming language. The course teaches how problems can be solved computationally using the object-oriented approach that underlies Java. No previous programming experience is required.

Course objectives

1. To enable you to develop problem solving and programming skills to design solutions to non-trivial problems and implement those solutions in Java.
2. To provide you with knowledge of essential facts, concepts, principles, and theories relating to object-oriented programming and design.
3. To prepare you to develop programming skills that can serve as a foundation for further study in computer science.
4. To equip you with skills to work productively as part of a team and to develop your ability for organization, communication, and collaboration.

Learning outcomes

At the end of the course you should be able to:

1. Demonstrate knowledge and understanding of fundamental programming constructs, variables, expressions, assignments, I/O, control constructs and recursion.
2. Deploy appropriate theory, practices, and tools for problem definition, specification, design, implementation, and testing of programs that use basic computation, simple I/O, standard conditional and iterative structures, the definition of methods, and parameter passing.
3. Use object-oriented design (inheritance, interfaces, polymorphism, abstract classes) as a mechanism for problem solving as well as facilitating modularity and software reuse (refactoring).
4. Design and model recommended programming practices (Java style and documentation, UML diagramming, and testing).
5. Work productively as part of a team and demonstrate your ability for organization, communication, and collaboration in teams.

Course Structure

The course follows a flipped classroom model of teaching and learning. We will be using lectures, labs, and the “Programming in Java (Early Objects)” online textbook from zyBooks.

1. **Lectures:** Participation activities from the textbook are due before lecture and challenge activities are due after the lecture. Lectures are interactive and parallel the text material, but may expand upon or otherwise deviate from the text. In lecture, you can download starter code and develop along with the instructor and use iclickers for group work.

2. **Labs:** You work in a group at developing code and answering questions that reinforces the topics covered in the text and in lecture. Although the lab document is submitted individually, the results may be the same for each group member. See the *Lab Protocol* document in Moodle for more information.

3. **Textbook:** We will use zyBooks' Programming in Java (early objects) state-of-the-art learning material, proven effective, and designed to maximize learning while respecting student time. The online textbook has embedded exercises and assignments. Participation activities are due before lecture and challenge activities are due after the lecture.

This course uses Moodle and Gradescope systems.

1. **Moodle** is a Learning Management System (LMS) used for posting lecture materials and for online exams. The Moodle webpage, when expanded, shows the weekly topics and material covered in lecture and labs. You are enrolled in the Moodle course through Spire.

2. **Gradescope** is a grading management system that allows us to give you feedback for your lab assignments and programming projects. You will receive a course code to create a student account in Gradescope.

Course Grading

Your course final grade is composed of marks from the zyBook assignments, projects, and exams, and attendance at labs and lectures. The exams test the ability to recognize, trace, implement, and translate code. The assignments and projects deal
with higher order programming skills (analyse, adapt, debug, test, apply, design, model, and refactor).

**Grading Categories and Weights:**

- zyBook questions including end of chapter exercises: 10%
- Lab Participation(attendance and document) 10% (the first 2 labs are not counted; the lowest 2 grades are dropped)
- iClicker questions: 5% (based on lecture attendance; the first 2 lecture sessions are not counted; the lowest 2 grades are dropped)
- Programming projects (5 projects): 35%
- Exams (four): 40%. In order to pass the course with a letter grade of D or better, all exams must be attempted and the average of all exam scores must be at least 50% of the total exam points.

<table>
<thead>
<tr>
<th>Learning Outcomes</th>
<th>zyBook questions and exercises 10%</th>
<th>Exams 40%</th>
<th>Projects 35%</th>
<th>Labs 10%</th>
<th>iClicker 5%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recognize, trace, implement, translate, debug code</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Analyze problem, apply, adapt, relate, debug</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Model, design, test and refactor</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Teamwork values</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Your final letter grade is determined by calculating a [course final grade](#) based on a weighted sum of your scores for individual grading categories. This number is mapped to a letter grade. There are no opportunities for extra credit in this course. See the *Attendance and Grading Policy* document posted in Moodle for more details of how assignments and projects are graded.

You may expect the following approximate grading range:
- 90 to 100 -> A
- 80 to 89 -> B
- mid 60 to 79 -> C
- 50 and above -> D
The instructors reserve the right to make adjustments to the grading ranges listed in this document. The instructors may assign a grade based on extenuating circumstances and/or according to his or her judgment.

You can review the grades you receive on programming projects and exams within 3 days release of the grade. It is important that you ask any questions as soon as possible to clarify any points about your grade or the content you submitted. We will not review any grades after 3 days beyond their release date.

All graded materials for this course will be retained until the end of next semester (Spring 2019). If you wish to review them, please come to see the instructors during office hours (or make an appointment).

**Course Materials**

1. You will need a reliable laptop computer for this class. Chromebooks, Microsoft Surface, and tablets will not work for this course. Your laptop must be configured to access the Eduroam wireless network. Please see or visit the campus OIT office for support. *Lack of working equipment or reliable internet access is not a valid excuse for failure to submit assigned work*. See the *Software Requirements* document posted in Moodle for more details. See: [https://www.umass.edu/it/wireless](https://www.umass.edu/it/wireless)

2. iClickers are required for lectures. You must register your iClicker in the Moodle course page to get points for answering iClicker questions. You must purchase a specific type of iClicker model 2. See [https://www.umass.edu/it/audience-response-system](https://www.umass.edu/it/audience-response-system).

**Course Support**

**Office hours:** See the Moodle page for office hours for Instructors and TAs.

**Tutors:** Individual Tutors as well as group review sessions (SI and EXSEL) are available at the Learning Resource Center, 10th floor in the Main Library (Du Bois). See [http://www.umass.edu/lrc/](http://www.umass.edu/lrc/)
**Communication:** Use Moodle for all course communication with your Instructors and TAs. We will not answer emails. Post your private messages (about grades, absences, or extensions) in the “**Private Student Forum**” in Moodle and we will get back to you as soon as possible. For general questions about the course you can post in the “**Public Student Forum**” in Moodle. You can also answer questions from other students in this public forum. See the **Communications Policy** document posted in Moodle for more details.

**Course Policies**

**Attendance:**

Attendance at lecture and labs is mandatory. The lowest 2 grades for labs and the iClicker questions are dropped. Beyond these, medical conditions, religious or funerary events, university-related event (athletic event, field trip, or performance), extenuating non-academic reasons (military obligation, family illness, jury duty, automobile collision) will be accommodated with written documentation. See the **Attendance and Grading Policy** document posted in Moodle for more details.

**Late submissions:**

Late work will not be accepted. If you need an extension for an assignment, contact us at least 24 hours before the assignment is due. Medical conditions, religious or funerary events, university-related event (athletic event, field trip, or performance), extenuating non-academic reasons (military obligation, family illness, jury duty, automobile collision) that need extension will be accommodated with written documentation. **Problems with computer or internet access, holiday, or family travel are not valid excuses.**

To ensure that you submit your work on time you should begin early and not wait until the last minute to submit. You will be able to submit multiple times so submit early and often to ensure you have something in before the deadline.

**Academic Honesty Statement:**

Students are **required to read and follow** the **Academic Honesty policy** document posted in Moodle. Students who violate this policy may receive a failing grade.

**Accommodation Statement:**
Any student who requires an accommodation due to a disability is directed to contact the UMass Disability Services: http://www.umass.edu/disability/ to obtain the appropriate accommodation forms if they haven't already done so. We cannot provide accommodations without documentation from Disability Services.

It is our goal to provide every student with a high quality learning experience. We invite you to contact us if you have any questions or concerns about disabilities or any issue that may impact the quality of your learning.

**List of Topics Covered (approximate)**

**CODE DEVELOPMENT**
- Coding style
- Commenting and documentation
- Modular code design
- Diagramming techniques
- Skeleton code (stubbing)
- Test your code

**PRIMITIVE DATA TYPES**
- Numeric: int, double
- Logical, character: boolean, char

**ABSTRACT DATA TYPES (Classes)**
- Java library classes: String, Scanner, Random, etc.
- User-defined classes

**OPERATORS**
- Assignment and Arithmetic Operators
- Relational Operators, Increment Operator, Logical Operations, Concatenation

**VARIABLES**
- Initialization of Instance Variables
- Type Conversion, Numeric Cast
- The **final** Keyword
- **static final** Variables

**EXPRESSIONS**
- Assignment
- Flow of Control: if, else
- Null References
- Packages
- Import Statements
RUNNING A PROGRAM
The **main()** Method
Using System Resources
Compiling Java source files
Using a debugger

OBJECT-ORIENTED PROGRAMMING
Defining a Class
Public Class Files
Objects and Encapsulation
Constructors
Instance Members, Class Members and Finalization
  Static Members
Setter and Getter Methods
Member Classes
Local Classes
Anonymous Classes
Nested Top-Level Classes
Casting

METHODS
  Method Signature
    Access level, return type, name, arguments
Method Overloading
Static Methods

INTERFACES
Interfaces and polymorphic behavior

IMPORTING JAVA LIBRARIES

GRAPHICAL USER INTERFACE (GUI)
  The event model
    Java libraries awt and swing

STRINGS
  the **toString()** Method
StringBuffer
EXCEPTIONS
   Handling Errors Using Exceptions
   Common Exceptions
   Checked and Unchecked Exceptions
   Chained Exceptions in Java

I/O
   Standard Input and Output Stream Classes
   File I/O using Scanner

INHERITANCE
   extending a superclass (generalization)
      abstract classes

POLYMORPHISM
   Polymorphism Based on Overloaded Methods
   Polymorphism, Type Conversion, Casting, Etc.
   Runtime Polymorphism through Inheritance
   Polymorphism and the Object Class

ARRAYS
   Arrays of Primitive Types
   Array indexing
   Length of an array
   Arrays of Objects
   2D arrays