***CSE 174 — Fall 2016 — Sections E, F, G, H***

|  |
| --- |
| **Instructor:** Norm Krumpe, 205L Benton Hall  **Course website:** Canvas, http://miamioh.instructure.com (homework turnins, online quizzes, grades), Google Drive for handouts. There is a link to the Google Drive folder from the Canvas site. |
| **Office Hours: Please come in to talk about homework, questions about the class, and so on**  Mondays and Wednesdays: 12:30pm – 3:00pm in Benton 205L  I recommend choosing a meeting time using **http://meeting.krumpe.com**  Can’t make these hours because of a conflict? I will work with you to arrange a time that fits my schedule and yours. |
| **HELP SESSIONS:** These are "drop in sessions" held in BENTON 8. Come in anytime, work on homework. Someone is around to help if you have questions. These hours are subject to change. They are not guaranteed, but we will try to keep them regularly.   * Tuesdays, 6-10pm in BENTON 8 * Wednesdays, 4-10pm in BENTON 8 * Thursdays, 6-10pm in BENTON 8 |
| **Open Labs:** Benton 16, Benton 8 (classes are scheduled in those rooms at some times, but one or the other should be open most of the time) |
| **Text:** *Big Java: Late Objects*, Horstmann  **Clicker:**You will need to purchase a TurningAccount license, which works with an app on your smartphone. If you do not have a smartphone, or would prefer not to use it, you can also purchase a "clicker" that works with the license. Although you can buy licenses and clickers at Miami's bookstore, there is currently a rebate that is only valid if you purchase through the online store. The rebate will allow you to get a 4-year license for free. Your participation with your clicker or smartphone is part of your participation grade, and we will begin using these on the second day of class (Wednesday, August 31). **See the course Google Drive for rebate information.** My personal recommendation: If you have a fairly current smartphone (Apple or Android), just get the license, free after rebate.  **Software (free):** *Java SE Development Kit (JDK)*  *Dr. Java***,** free from <http://drjava.org> but only works AFTER you install the JDK  See [this document](https://docs.google.com/document/d/1qTnAQde-XcOmTnqDmxN-RSyjFgGmw8HpMMnLqP0sttk/edit?usp=sharing) for information about setting up Dr. Java and the JDK. |

# 

# Meeting times:

* Lectures: 10:00-10:55am in KRG 319
* Labs: 10:00-11:50am in BEN 16 (section E), 2 (section F), 6 (section G), or 1 (section H)

# 

# *Fundamentals of Programming and Problem Solving*

This is a course where you learn how to program. You will learn Java, and you will learn how to solve problems with Java. If you have ever wanted to develop mobile applications, video games, or any other type of software, this is the class where you will develop the fundamental skills you need. Expect to work hard. By the time the course is done, you will have written thousands of lines of code, and will likely spend 75-100 hours writing programs. Programming can be challenging and frustrating, but it can also be a lot of fun. Look for ways to make it fun, and you will find that time flies when you are coding.

***Learning Outcomes: By the end of this course, you should be able to...***

1. Write programs in a contemporary programming language
2. Solve programming problems in a clean and robust fashion using a procedural approach
3. Use a contemporary programming environment
4. Trace and explain the flow of programs that include multiple data types, conditionals, loops, console and file I/O, and multiple methods
5. Identify, describe, and eliminate errors in programs
6. Read, understand, and communicate technical information
7. Write code and documentation that emphasizes readability and uses specified style guidelines
8. Understand and apply mathematical concepts and models
9. Communicate a main idea in a logical way with supporting evidence
10. Identify, analyze, and compare multiple solutions to problems

***Important dates***

Fri, Sep 16 Last day to drop this course (no grade assigned)

Fri, Sep 23 **Exam 1**

Fri, Oct 14 No classes - Fall Break

Fri, Oct 21 **Exam 2**

Mon, Oct 31 Last day to withdraw from this course (course grade will be “W”)

Fri, Nov 18 **Exam 3**

Wed, Nov 23 Thanksgiving Break begins - No class

Fri, Dec 9 Last day of class

Wed, Dec 14 **Final exam, 8:00-10:00am**

# *Course policies:*

**Attendance - Read carefully**

* Course grade decreases by 3% for each absence beyond 2 (this is in addition to any deductions due to missed in-class labs/activities). This means, for example, that if your final grade in the course would have been a 95%, but you missed 4 days of class, your final grade would be reduced to 89% (95% - 3% - 3%). Note that absences count for *any reason*. You essentially get three absences that do not count against you. If you miss because you are sick, or sleep in, or have a job interview, and so on, those are all considered "absent".
* More than five minutes late = half absence
* I reserve the right to drop you from the course if you miss more than 3 days. If you cannot afford to be dropped from the course, then do not miss more than 3 days.
* In case of an absence:
* inform me beforehand, if possible
* submit on time any work that is due
* understand that there are no make-ups for missed labs, quizzes, exams or other in-class work

Regarding the idea of “excused” absences, the following excerpt is from Miami’s Student Handbook:

“There are no University-recognized excused absences except for religious observances that require absence from a class session and other required class activities. Students must give written notification to their instructor within the first week of each full-term or sprint class of the religious event that prohibits class attendance and the date that will be missed, if officially known.”

**E-mail and Announcements**

Regularly check the course website (http://miamioh.instructure.com) and your *Miami* e-mail for announcements.

**Textbook (part of your participation grade):**

Keep up with the reading. Work through the activities in each chapter. Answer clicker questions in class that are based on the reading. Answer online quiz questions. You generally do not need to bring your textbook to class, but there are times when you may want it with you in the lab as a programming reference.

**Taking notes**

I sometimes use in-class electronic presentations to give you basic information. These are not a substitute for taking notes.

* Take notes during lectures, because…
* …lab activities will often depend on you to use what you wrote in your notes.
* "Good notes" does not mean "Write everything". Be selective.
* Focus on writing sample code, diagrams, "notes to self".

**Participation**

Being in the room when someone is talking does not count as participation. Here is how to participate:

* Keeping up with the reading
* Asking questions in class
* Volunteering answers to questions in class (clicker and non-clicker questions, in-class activities)
* Encouraging others to get involved
* Contributing to online discussions (asking good questions, helping provide answers, not posting code)
* **Helping other students during labs**

**Clickers or ResponseWare app (part of your participation grade):**

* We will start using clickers on the second day of class. You will need to purchase a TurningAccount license (free after rebate). If you want, you can also purchase a physical clicker. However, there is a free ResponseWare app that works with your license.
* Register your TurningPoint account by going to the course website and following the instructions in the announcements.
* Your responses are part of your participation grade.
* I assume that everyone will have a couple bad clicker days, and so I compensate for that in your participation grade.

**Homework**

* **Roughly 14 assignments, typically programming assignments**
* **Cannot be turned in late.**
* **Must be submitted using Canvas using the turn-in feature of the appropriate assignment (e-mail submissions are not accepted).**
* Start early…start early…start early…start early…start early.
* WRITE YOUR CODE ALONE…learn to help one another without sharing any code.

**Labs and other in-class activities**

* Half of our meeting time each week will be devoted to labs and other hands-on activities.
* Lab assignments are to be completed during in-class time…they are not homework.
* Lowest lab assignment grade will be dropped.
* There are no make-ups for missed labs.

**Exams**

* There will be three midterm exams, and one final exam.
* All exams are cumulative, closed-book.
* You may have one sheet of notes for each exam. This sheet must be handwritten (and it must be handwritten by *you*), and turned in at the end of your exam.
* **No make-ups for missed exams**. If you are absent for an exam, your grade for that exam will be zero.

**Your grade will be determined as follows:**

4 Exams 70%

Homework 20%

Labs 5%

Participation and other work 5%

Total 100%

**Grading scale**

|  |  |  |
| --- | --- | --- |
| If grade is at least… | but less than… | your grade will be… |
| 92 | 100 | A |
| 90 | 92 | A- |
| 88 | 90 | B+ |
| 82 | 88 | B |
| 80 | 82 | B- |
| 78 | 80 | C+ |
| 72 | 78 | C |
| 70 | 72 | C- |
| 68 | 70 | D+ |
| 62 | 68 | D |
| 60 | 62 | D- |
| 0 | 60 | F |

|  |  |
| --- | --- |
| **Topics**  Tools   * Compiler * Editor * Debugger * Java Virtual Machine   Basic Language Elements   * Types, operators, variables, constants * Strings * Type conversion   Input/Output   * Interactive input/output * Sequential file input/output   Control Structures   * Sequence * Selection (branching) * Iteration (looping)   Testing and Debugging   * Compile-time vs. run-time errors * Testing and debugging strategies * Test case selection | Using Standard Classes   * String, Random, ArrayList, Math, Scanner * Instantiating objects and invoking their methods * Using static and non-static data and methods   Methods   * Designing and implementing methods * Method decomposition * Variable lifetime and scope * Parameter passing and return values   Problem solving   * Functional decomposition * Designing methods   Arrays   * Single and two-dimensional arrays * Arrays of objects   Algorithms   * Searching arrays: linear and binary * Standard array processing algorithms * Non-recursive sorting techniques * Searching and sorting using library routines |

**Official course description:**

Algorithm development and refinement in problem solving. Modular programming using sequence, selection, and repetition control structures. Program debugging and testing. Formatted input/output. Data files. Fundamental data types. User-defined data types: structured and enumerated. Arrays and arrays of structures. Simple sorting and searching algorithms. Character data and string processing. Algorithm efficiency considerations. Classes, objects, and introduction to object-oriented programming.

**Calendar (subject to change):**

|  |  |  |
| --- | --- | --- |
| **Week** | **Content** | **Textbook Chapters (and self-check)** |
| Aug 29-Sep 2 | The programming process  The structure of Java programs  Programming tools: editor, compiler, Java Virtual Machine  Programming practices: formatting, commenting code  Programming errors: compiler errors, runtime errors | Chapter 1: Sections 1.1-1.6 |
| Sep 7-9 | Console output (using printf)  Primitive data types: numeric, character, boolean  Getting user input: Scanner  arithmetic operators, Math methods | Chapter 2: Sections 2.1-2.3 |
| Sep 12-16 | Variables  String introduction  String operations  Combining different data types  Java API documentation | Chapter 2: Section 2.5 |
| Sep 19-23 | Boolean operations: and, or, xor, not  Decision making: if statements  **Exam 1 - Friday, Sep. 23** | Chapter 3: All sections |
| Sep 26-30 | Decision making: if, else, switch  Problem solving with if, else  Flowcharts |  |
| Oct 3-7 | Methods  Programming tools: javadoc  Using CodingBat to implement methods | Chapter 5: Sections 5.1-5.8 |
| Oct 10-12 | Programming tools: debugger  Loops: while and do loops | Chapter 4: Sections 4.1 and 4.4 |
| Oct 17-21 | Loops: for loops  Common loop algorithms: adding, printing, counting  **Exam 2 - Friday, Oct. 21** | Chapter 4: Sections 4.3, 4.7-4.9 |
| Oct 24-28 | Object-oriented programming: using objects  Classes vs. objects  Using object constructors and methods  Java API documentation revisited |  |
| Oct 31-Nov 4 | File input and output  Arrays | Chapter 7: Sections 7.1 and 7.2  Chapter 6: Section 6.1 |
| Nov 7-11 | Arrays and loops  Common array algorithms  The "for each" loop | Chapter 6: Sections 6.2-6.4 |
| Nov 14-18 | ArrayList  Two-dimensional arrays and beyond  **Exam 3 - Friday, Nov. 18** | Chapter 6: Sections 6.7-6.8 |
| Nov 21 | Search algorithms: linear and binary search |  |
| Nov 28-Dec 2 | One lecture, no lab - Thanksgiving Break begins Wed.  Sorting algorithms: bubble sort |  |
| Dec 5-Dec 9 | Wiggle room |  |
| Dec 12-16 | Final exam is Wednesday, December 14, 8-10am |  |

**Computer Science and Software Engineering**

**Academic Integrity Expectations for**

**Individual and Group Problem Solving Assignments**

**The Department of Computer Science and Software Engineering is committed to maintaining strict standards of academic integrity. The department expects each student to understand and comply with the University’s Policy on Academic Integrity:** <http://www.miamioh.edu/integrity> and the [undergraduate student handbook](http://miamioh.edu/_files/documents/secretary/Student_Handbook.pdf) and [graduate student handbook](http://miamioh.edu/documents/graduate-studies/Graduate_Handbook.pdf).  **Students may direct questions regarding academic integrity expectations to their instructor or to the department chair.** All work submitted must be **original** for that class. Submitting the same project for two different classes is grounds for charging a student with academic misconduct unless prior written permission is received from **both** instructors.

“Problem Solving Assignments” are assignments that involve **programming, math, proofs, derivations, and puzzles**.

The purpose of a problem solving assignment is for you to develop the skills necessary to solve similar problems in the future. To learn to solve problems you must solve the problems and write your solutions independently.

It is worth reiterating that the important aspect of the assignment is that you actually create the solution from start to finish; simply copying a solution and then **understanding it after the fact is not a substitute** for actually developing the solution.

The notion of academic integrity can be confusing in courses with substantial problem solving because certain forms of collaboration and investigation are permitted, but you are still required to complete your assignment independently. The following scenarios are meant to help distinguish between acceptable and unacceptable levels of collaboration and research, but are not all-inclusive:

ACCEPTABLE:

* Consulting solutions from the current course textbook, but not from other published sources.
* Seeking help on how to use the programming environment such as the editor, the compiler, or other tools.
* Seeking help on how to fix a program syntax error or how a certain language feature works.
* Discussing strategies with a fellow student on how to approach a particular problem. This discussion should not include significant sections of completed work or source code (including printouts, email, viewing on a monitor). Discussions should begin with a clean sheet of paper and end with conceptual drawings and/or pseudo-code.

UNACCEPTABLE:

* Looking at another solution including those written by current students, past students, or outside sources such as code or solutions found on the Web, or in publications other than the current class textbook.
* Using another solution as a starting point and then modifying the code or text as your own work.
* Providing a copy of your solution or a portion of your solution, in any form (electronic, hard copy, allowing another student to view your code on a monitor), to another student.
* Giving or receiving code fragments to fix a problem in a program.

If you are stuck on a problem and you are tempted to search for a solution on the Web or to look at another student’s solution **STOP** and email or ask your instructor for help.