CIS 3260
INTRODUCTION TO PROGRAMMING
SAMPLE SYLLABUS

Prerequisite:
CIS 2010, CSP 1 or Graduate Status.

Required Material


Compiler:  You must have access to the Java compiler, part of Java SDK (Software Development Kit).  You may download it for free at http://www.oracle.com/technetwork/Java/Javase/downloads/index.html.  To develop Java applications and applets, you need the JDK (Java Development Kit), which includes the JRE.  The JDK is a development environment for building applications, applets, and components using the Java programming language.  The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the JavaTM platform.

API- http://docs.oracle.com/Javase/6/docs/api/overview-summary.html

The course syllabus provides a general plan for the course.  Deviations may be necessary.

Catalog Description:
Introduction to Programming. Prerequisite: CIS 2010. Requires a 2.5 GSU GPA and 45 semester hours. CSP: 1, 4, 7. This course provides an introduction to programming using a contemporary object-oriented language. Emphasis is placed upon the development of correct, efficient programs that are easy to maintain. Topics include problem analysis, program design, documentation, testing and debugging. Basic features of the programming language are covered. The specific language will be noted in the course listing for each semester.

Special Considerations:
Introductory programming courses are inherently difficult for most students without prior experience. The use of Java programming language in this course heightens the difficulty level even more, because it is a powerful languages designed for use by professional programmers, not for teaching programming to beginning students. It is the language of choice in today’s software development environment and the ability to program in Java is a valuable job skill, but it can be intimidating to a beginner or even to programmers experienced in other languages. The implication of this for you as a CIS 3260 student is that you must apply yourself diligently in order to do well in the course. This includes attending class regularly, reading the assignments before coming to class and working as many problems in the textbook as possible. You should be aware that missing even one lecture will put you significantly behind your classmates. If you must be absent, arrange to obtain the lecture notes of someone who is an accomplished note-
taker. If you have questions about a specific topic that you missed, see the instructor during office hours, but be aware that he cannot repeat a two-hour lecture for the benefit of one student. This course gets progressively more difficult!

**Method of Instruction:**
Each lecture will cover the same features of Java as the reading assignment in the textbook, but from a different, and usually more advanced, perspective. The lecture will not simply repeat the material in the textbook. Therefore, it is essential that you read the assigned material and work through the problems in each chapter in advance of the lecture. Otherwise, you may find the lectures incomprehensible.

**Class Attendance Policy:**
Roll will not be taken on a regular basis. It is the student’s responsibility to take notes, obtain assignments, and turn in work on time. Absence from class does not relieve you of any of these responsibilities.

**Withdrawals**
Students who withdraw by the official withdraw date will receive a grade of W. Students withdrawing after this date will receive a grade of WF unless a hardship authorization is obtained from the Dean of Students.

**Incompletes**
A grade of I will be given only in exceptional circumstances. A student must have completed all but one of the requirements of the course in order to be eligible to receive a grade of I.

Extra Credit: The instructor reserves the right to give extra credit assignments if the overall (not just an individuals) course performance is not up to the instructor’s standard.

**Programming Assignments:**
Programming assignments will be collected and graded. IMMEDIATELY familiarize yourself with the compiler and use it as a laboratory to conduct experiments that will help you to understand and write programs. DO NOT WAIT.

Note: Save your work often and make backups. Computers do crash, especially if your program misuses pointers. Always save your work before compiling or executing code.

**Exams:**
Generally, questions will involve writing and debugging programs and program fragments, as well as conceptual questions from the textbook and lectures. There can also be some multiple choice and fill in the blank questions.

Makeup Exams: A grade of zero will be assigned on any missed exam unless the instructor is notified in advance, a valid [in the opinion of the instructor] reason for missing is documented [i.e. doctors note, police report], and arrangements are made for how to make up for the missed assignment. Makeup exams will be essay/coding only, no multiple choice or short answers.
Grading Policy:

300 Points: 2 Exams

Exams are closed book, closed notes, with the exception of a single one sided 8 ½*11 hand written sheet of paper that needs to be turned in.
Exams take about one hour and are in general at the beginning of the class period.

350 Points Selected Assignments are graded, and their points prorated for ungraded assignments

Program 1 Pseudocode 10 Pts
Program 2 Input, Processing, Output 10 Pts
Program 3 Decisions 20 Pts
Program 4 Loops 20 Pts
Program 5 Methods 30 Pts
Program 6 Arrays 30 Pts
Program 7 Classes 60 Pts
Program 8 Classes 60 Pts
Program 9 Files 90 Pts
Program 10 Inheritance 20 Pts

50 Points: Participation, other homework and pop quizzes

300 Points Final : Comprehensive final with emphasis on objects, classes arrays, and strings.
The final is open notes, open book.

1000 Points total: 1000-900 A, 899-800 B, 799-700 C, 699-600 D, <599 F

Pop Quizzes: The instructor reserves the right to give unannounced pop-quizzes. These quizzes are closed book, and last about 10 minutes. There is no makeup for pop-quizzes

Grading of Assignments:
1. Assignments are due at the beginning of class on the date specified. Five % will be deducted for programs handed in after the lecture begins. NO programs will be accepted after the end of class on the due date, no exceptions. Do not wait until the last day to get printouts of your programs.

2. To submit: (1) A printed copy of your program code (=listing) should be submitted along with (2) printed evidence of the performance of your program (screenshot). This printed evidence should display on paper everything that appeared on your screen when you ran the program, including data that you entered via the keyboard. See instructions in following pages.

3. Testing is a critical part of the programming process. The burden of proof that a program works always rests with the programmer. Whether or not test data are provided, you should adequately test your program to insure that it works correctly in all cases. When test data are provided you must turn in your assignment using that same test data.

4. Many programming assignments specify that a particular approach be used. Read the assignment carefully to be sure that you understand how the problem should be solved. If you use the wrong approach, the fact that your program produces the correct output is immaterial.

5. Programs must be well-structured, readable and efficient. Use meaningful names, indentation, comments, and other elements of style discussed in the course. An unreadable program is not maintainable and is worthless even if it produces the correct result.

6. Output should be neat, properly aligned and have useful headings. Requests for interactive input should be preceded by a prompting message.

7. Programs must conform to the ANSI standard and should use only those features of Java which have been covered in the course at the time of the assignment.

In general, the following point system will be used as a guideline for grading of code:

**Overall Deductions:**

- 20% Not using the provided test data
- 20% Improper use of advanced Java features
- 20% Not following the instructions regarding the required approach
- 30% Your name does not appear in your code/output as specified:

**Standard program head for all homework submissions:**

// Programmer: your name
// Filename: have your filename here  xxx.cs
// Last modified: have your date here

// Description: a short description of what the program does

**Performance**

- 0% Program runs correctly and produces the correct output
- 10% and more Program has minor error (e.g. typo in formula or text, …)
- 20% and more Program has major error or many minor errors
- 30% and more Program does not run due to syntax error
- 40% and more Only program fragments turned in, program crashes,

**Maintenance**

- 20% Poor naming of variables/functions
- 20% Improper indentation
- 20% Poor program structure
- 20% No/poor documentation in code

**Input/Output**

- 10% No prompting for input
- 10% No headings
- 10% Output alignment problems (e.g. table headers)
- 20% No output provided

**Original Work**

Do not jeopardize your grade by allowing others to copy your work. Program and project submissions imply a wholly original work by the student. Academic credit is not available for copied program code. Blocks of code from external sources may NOT be inserted into student programs. Where project code has been adapted from external sources, comments must indicate the exact source of the original code.
Note: When you are stuck with a problem, it is permissible to ask a co-student/lab assistant to help find the error. You are also encouraged to email me your code - I return it corrected within 24 hours. If you do not understand the wording/meaning of a particular assignment, you can also consult with others or email me.

However, you are NOT allowed to copy someone else's code, nor to discuss solutions. This is not considered original work. The penalties for giving and receiving help are the same, and follow department and university policy and can result in dismissal from the course, department, and/or university.

**Submitting Evidence of Program Execution**

As in all the homework assignments, you should turn in a computer printout of your program (called listing) as well as computer generated printed evidence of the running of our program.

**Listing**  A listing can be usually obtained by selecting FILE-PRINT-LISTING.

This printed evidence should display on paper everything that appeared on your screen when you ran the program. In future assignments, this will include data that you entered via the keyboard.

**Windows 95… XP Environment with Word Processor:**

After you have run your program in a window, click with the mouse once on the window to make it the active window (change in frame border color). Hit the “Alt-PrintScreen” (hold down the Alt key and hit the PrintScreen key). This makes sure that the current window as seen is copied to the clipboard. Now open your word processor (e.g. Word for Windows), and from the top menu select Edit/Paste. This inserts the screenshot from the clipboard. Resize it so that the output is readable. Save that wordfile under a unique name, e.g. prog1-screen1. Since it is “difficult” to work with multiple screenshots in a word document, print out each of these screenshots one at a time. Proceed with the other screenshots if necessary.
# TENTATIVE CLASS SESSION SCHEDULE
(Can change at any time)

<table>
<thead>
<tr>
<th>CLASS</th>
<th>TOPIC</th>
<th>CHAPTER</th>
<th>PROGRAM DUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction to programming</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Elementary Programming</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>Selections</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>Loops</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>Methods</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>Single Dimension Arrays</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td><em>Exam 1</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Break/No Class</td>
<td>NONE</td>
<td>NONE</td>
</tr>
<tr>
<td>8</td>
<td>Multidimensional Arrays</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Objects/Classes</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>10</td>
<td>Thinking in Objects</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td><em>Exam 2</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Strings and Text I/O</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>12</td>
<td>Inheritance and Polymorphism</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>13</td>
<td>Exception Handling</td>
<td>13</td>
<td>NONE</td>
</tr>
<tr>
<td>14</td>
<td>Final</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>