Intro To CS234C#

- This course is the third in a multi-course sequence of courses that teach students to program in C#
  - C# is a high level language
  - C# is an object oriented language
  - C# can be used to build command line or event driven programs
  - C# can be used to build desktop, web based and mobile applications
  - C# is used primarily to build applications that run in a Windows environment

    There is an open source project called Mono that built a version of the .NET Framework and the Common Language Runtime that will run on other OS platforms.

Intro To CS234C#

- We chose C# as the primary programming language for programming majors and health informatics majors because
  - It's one of the major programming languages used by programmers in all kinds of fields.

    The health care profession uses .NET and/or Java.

    VB.NET is used as well as C# in the .NET environment. If you can learn C# you can switch to VB very easily.
  - It can be used for both desktop, web based and mobile applications.

    Unlike open source languages like PHP that are primarily used for web based applications.

    That means that you can don't have to learn a new programming language to program for the web.

    AND you can also use C# to do mobile development for both the Android and IOS platforms

Intro to CS234C#

- CS133C#(N) – introduces you to fundamental programming concepts using C# as the language and Visual Studio as the IDE. You built desktop applications with graphical user interfaces. Your used classes that are part of the .NET framework but we won't build our own classes.
- CS233C#(N) – introduces object oriented programming in C#. We'll build our own classes and add them to our desktop applications with graphical user interfaces.
- CS234C#(N) – builds on your understanding of oriented programming in C#. We'll build data driven desktop applications.
- CS295A(N) – introduces server side web development in an ASP.NET environment. We'll learn all about the ASP.NET classes that are used to build web based applications and will continue to program in C#.
- CS296A(N) – introduces more advanced components in the ASP.NET environment. We'll build sophisticated web based applications.
- CS 235 AM – introduces mobile application development for android
- CS 235 iM – introduces mobile application development for IOS

Other Programming Courses

- CS 161/162 – is the programming sequence designed for computer science transfer majors. The emphasis is on teaching folks with lots of math background and an interest in theoretical computer science how to program
  - CS161C+/CS162C+– teaches C++ programming to the students who are gaming majors and/or who intend to transfer to a 4 year school where C++ is the language of choice.
  - CS161J/CS162J – teaches Java programming to the students who intend to transfer to a 4 year school where Java is the language of choice. We're not currently teaching the Java version.
- CS 133JS – teaches students with no programming background to add client side programming to web pages in JavaScript
- CS 295P/CS 296P – teaches students with html and JavaScript experience to create server side web based applications in php.

The Course Itself

- Classes.lanecc.edu – “Moodle”
- In class
  - Lecture and Demo
  - Laptops for hands-on exercises and help with labs
- Textbook
  - Is the book that we used in 233N. It ss a “trade book”. Good “how to” information but light on explanation. Lots of “snippets” of code. One or more “complete examples” in each chapter. One or more exercises that are VERY similar to the text example in each chapter.
  - Start with the chapters on building classes which should be review from 233N (12 – 13)

The Course Itself

- Labs
  - You'll be working on a lab every week. 8 total for the term but 1 of them will take us more than one week and is therefore worth 40 points.
  - Turn parts in electronically. Moodle contains the details.

    Let's look at lab 0 in moodle now.
  - In the lab or at home. Need Visual Studio 2015.
- Quizzes
  - Reading quizzes each chapter. Multiple choice. 8 for the term. May ask you to take once in class. You can re-take outside class. I take the highest score. No makeups.
  - "Regular quizzes". 2 per term. After we finish specific content. Mixture of multiple choice, matching and problems. Practice quizzes will get you ready. Makeup only with advance notice.

You Might Want To

- Set aside at least 12 hours each week for the course.
- Read the text book and the lecture notes before coming to class and certainly before attempting to complete the associated lab assignment.
- Read the text book and the lecture notes as many times as necessary to understand the content. It is not uncommon for students to read the text at least twice.
- Take notes as you read about the things you don't understand. Ask those questions in class or in one of the forums for the class. Don't stop asking related questions until you could explain the material to a classmate on a forum or to me on a quiz.
- "Play with" the examples in the book. Type them in, test them, change them trying to answer "what if" questions.
- Work on the lab assignments in several short time segments rather than in one longer block of time on the day before the assignment is due. When you feel like you're stuck, make some notes about your questions, ask your questions in a forum and take a break.
- Form a study group. Talking to other students about the content of the course will clarify your understanding as well as your misunderstanding.

Lab 0 – Mexican Train Dominos

- The first lab of the term gives you practice building a class. It will also give you the opportunity to learn about unit testing.
  - Begin by dividing yourselves into groups of 3 or 4. This is your "team" for this lab. You will each do your own programming but you are "responsible" for the success of everyone in your team.
    - When you complete your peer evaluation for the lab, you'll give each other feedback about how "constructive" you were as part of a team. You can also "give each other extra credit point" for being exceptionally helpful and/or for asking lots of good questions that caused other people on your team to have to really think about what they were doing.

Lab 0

- As a team
  - Familiarize yourself with Mexican Train Dominos
    - https://en.wikipedia.org/wiki/Mexican_Train
    - http://www.domino-games.com/dominoes-online/mexican-train-dominoes-dillydally.html
  - Identify the classes that you'll need to build the application. For each of the classes, develop a set of fields, properties and methods that you think you'll need to implement. Document your classes by creating a sketch of a class diagram for your solution.
  - Download the starting files for the lab. I have given you several TEXT files and a class diagram.
    - Domino and BoneYard (DominoDeck) classes
    - nUnit unit tests for Domino

Lab 0

Lab 0

- As a team
  - Compare the classes that you designed with the classes on my class diagram.
    - Which methods/properties did you have?
    - Which methods/properties did you miss? Do you have questions about any of those?
    - Examine the source code for the 2 classes. Could you have written them yourself? Questions?
- Let's create a solution together
  - Create a new solution that contains a class library.
    - Name the project MTDClasses. Name the solution MTDominos.
    - Rename class1 to Domino in the solution explorer. Copy the contents of the Domino class from the text file into VS.
    - Add the BoneYard class. Copy the contents from the text file.

Unit Testing

- One of the problems with the kind of testing we've been doing is that it requires that a human being read the output from each test and compare what's expected to what is in the output.
- Wouldn't it be lovely if there was software that did that part for you? Then you could have a less experienced programmer who's responsibility it is to do quality control do your testing for you!
  - There are lots of testing tools out there. I'm going to show you how to use a tool called nUnit from within the Visual Studio IDE.

Unit Testing

- Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. Unit testing is often automated but it can also be done manually. This testing mode is a component of Extreme Programming (XP), a pragmatic method of software development that takes a meticulous approach to building a product by means of continual testing and revision.

Lab 0

- As a team
  - Use the Extensions and Updates menu choice from the Tools menu to add the nUnit Test Adapter
  - Add another class library, not a console app, for Unit Testing to your solution. Name it MTDUnitTests
    - Add a reference to your class library.
    - Manage NuGet packages to add the nUnit framework to your test projects.
    - I've given you a test fixture for the Domino class in the starting files. Rename the one class that's in your test project to DominoTests. Copy and paste the body of my class into your DominoTests class.

Lab 0

- As a team
  - In DominoTests
    - Notice the TestFixture and Test Annotations in the test class.
    - Notice that each test tests ONE thing.
    - Notice that instead of writing to the console, the tests “assert” that something will be true after executing some of the code in the class.
    - Open the TestExplorer window. Build your solution and then run all of the tests. Do they pass? How do you know?
    - Notice how I test exceptions. There are 2 versions. One that uses try and catch blocks. One that Asserts that an exception is thrown. That version uses delegates and lambda expressions. If you can’t “mimic” it without understanding it, use the other strategy.
    - Notice the instance variables and the SetUp annotation.

Lab 0

- As a team
  - In DominoTests
    - Add a class for testing the BoneYard class.
    - What tests will you have to write? Write “stubs” for each of those and in each assert that the test fails.
    - Let’s write a couple of tests together.
    - Finish the rest of the BoneYard tests with your team.
  - At this point you should be ready to implement and test JUST THE TRAIN class.
    - Let’s talk about what I want for the Train class in more detail using the class diagram as a guide.
    - Write all of tests BEFORE you write the class. Write a method and test is. Repeat for the next method until you’re done.

Lab 0

What's Next

- Peer Evaluation for Lab 0
  - BoneYard tests
  - Train class + tests
- Reading Quiz 1 – Review
- Reading Quiz 2 – Unit testing
- Chapter 14 and Lab 1 and Reading Quiz 3
- Chapter 15 and Lab 2 and Reading Quiz 4
- Quiz 1