ISAT 252—Programming & Problem Solving Spring 2012 Syllabus

Introduction

"Computational Thinking" is a term coined by Jeanette Wing, Chair of the Computer Science Department at Carnegie-Mellon University. According to Wing, Computational Thinking:

- is a way of solving problems, designing systems, and understanding human behavior that draws on concepts fundamental to computer science. To flourish in today's world, computational thinking has to be a fundamental part of the way people think and understand the world.
- means creating and making use of different levels of abstraction, to understand and solve problems more effectively.
- means thinking algorithmically and with the ability to apply mathematical concepts such as induction to develop more efficient, fair, and secure solutions.
- means understanding the consequences of scale, not only for reasons of efficiency but also for economic and social reasons.

If you think about these points, you'll realize that computational thinking is NOT just for programmers, but represents a general approach to problem solving useful in most all disciplines. That is why the ISAT 252 course is a required part of the ISAT curriculum. You should begin to find the skills you learn here to be useful in a broad array of the problem-solving activities you'll encounter in life.

> *The most important thing you will learn in this class is NOT how to program, but rather how to approach a problem, analyze a problem, design a solution to a problem, and implement a solution to a problem using strategies and techniques developed and honed in the programming community.*

You will also learn how to write real, practical, working programs.

Suggested Learning Objectives

Many problems may be solved with computers and there are many skill sets one might acquire in pursuit of this goal. As your instructor, I have two objectives, that you:

1. ***engage seriously with the content***–you (or somebody) spend too much money for you to be here to waste this opportunity
2. ***stretch*** yourself–if it doesn't hurt a little, you're not doing it right

Ultimately, though, your objectives for this class need to be your own. Here are some suggestions of things you might like to come away with, the ability to:

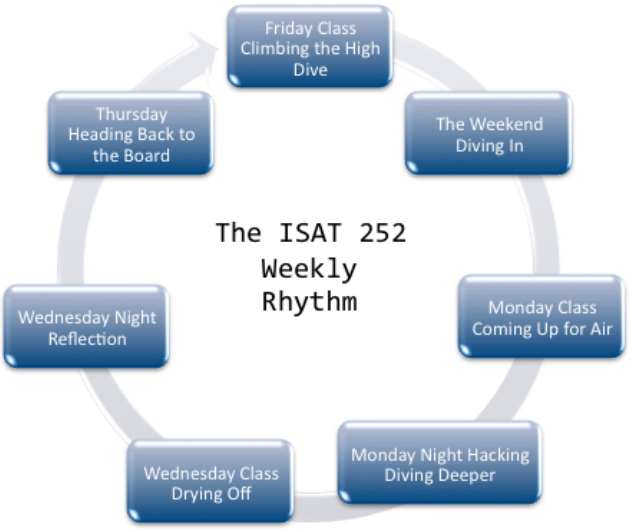
1. Demonstrate basic familiarity with computer hardware functions
2. Recognize when problems in science and business are amenable to software solutions
3. Analyze such problems and generate plans for implementing software solutions including elements such as class diagrams, flow charts, and pseudocode
4. Implement relatively simple software applications with graphical user interfaces
5. Employ procedural programming constructs effectively: variables and constants, conditional expressions, flow control, arrays, modularity with sub-procedures and functions, and input/output with text files
6. Employ event-driven programming constructs effectively: built-in and custom event-handling procedures, catching and handling exceptions
7. Employ simple object-oriented programming constructs: distinguish between objects and classes, identify when classes would be useful in a program, implement a basic class, and use that class in a program
8. Use a range of debugging techniques
9. Document code appropriately to increase ease of maintenance and understanding
10. Read and understand code that others have written with an eye towards code re-use
11. Describe knowledge-based systems and the problem types to which they are best suited
12. Build a simple KBS using appropriate architecture and methodology
13. Use 3 forms of knowledge representation: pseudo code rules, decision trees and rules in a program/shell rule base
14. Extract production rules from a narrative, identify key variables, and construct decision trees in an applied problem-solving scenario
15. Evaluate a given problem within its social context and identify an appropriate paradigm within which to develop a software solution

Weekly Rhythm

The single most important thing you can do to succeed in this class is to dedicate time each week to learning the material. In order to help you do that, our week will follow a very regular rhythm.

Our days will be spent as follows:

- **Friday's Class–Climbing the High Dive**
  On Fridays we will introduce new concepts and set goals and tasks for the upcoming week by writing a lab together.
- **The Weekend–Diving In**
  Over the weekend is your time to read the textbook, watch online videos and tutorials, meet with your team, and to get started working on the lab we write on Friday.



- **Monday's Class–Coming Up for Air**
  Monday's class will be largely unstructured Q&A time. You should come to class prepared with questions you want to ask about things you started over the weekend. Your team should be there on time and ready to hit the ground running. Our class TA will be there on Mondays to help you out where you are stuck.
- **Monday Night Hacking–Diving Deeper**
  Every Monday night from 8pm-Midnight the lab will be open and your professor and TA will be present to work with you in a longer, more relaxed setting. This is a great time to spend several hours of focused, uninterrupted coding with your team, or by yourself.
- **Wednesday's Class–Drying Off**
  The lab from Friday is due every Wednesday at class time. We will go over the labs together in code review sessions. Usually one team will present their week's work and receive feedback from everyone in the class.
- **Wednesday Night–Reflection**
  One of the most important elements of success is reflecting on your experiences. Every Wednesday night you'll have a self-evaluation exercise and possibly a peer-evaluation exercise to complete. You'll rate yourself, your classmates, and your instructor on their performance for the week.
- **Thursday–Heading Back to the Board**
  On Thursdays we'll look back on the week and take stock of how far we've come and where we want to go in preparation for our new task on Friday. The cycle is complete. Time to start again!

How much time will this class take per week?

A good rule of thumb is that you should be spending **at least 10 hours per week**per 3-credit course. That includes class time. If you find you like this stuff, though, it will be very easy to spend a lot more time on it. You'll have an opportunity to report the amount of time you're spending during your weekly self-reflection. A good strategy for success is to set up

a weekly calendar in which you set times when you will work on 252-related tasks each week including meeting with your team.

Useful Resources

Here are a number of things you will very likely find useful for this course.

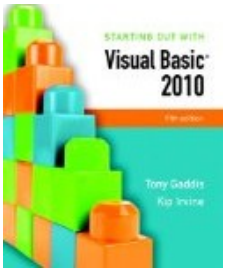
Software

All software required for completing this course is available in the computer labs on the third floor of the ISAT/CS Building. Most of the software is also available for download for free to ISAT students. It is not necessary to download and install all of this software, but if you'd like to work on programming projects at home you will likely wish to do so. All of the software is for Windows. If you are using an Intel-based Mac, then you'll need to install Windows in a virtual machine or via Bootcamp in order to use the programs below. You can obtain a free, legal copy of any Windows OS via the MSDN Academic Alliance site.

- **MultiMedia Logic**
  This is a GUI tool for creating and testing logical circuits that we'll be using early on in the semester to get a better understanding of how computers work.
- **Visual Studio 2010 Professional**
  This is the primary programming environment we'll use for doing Visual Basic projects. It retails for about $700 but you can get a free copy via the Microsoft Developers Network (MSDN) Academic Alliance website. To log in to this site, your username is your full JMU email address, e.g. youreid@dukes.jmu.edu. Your password should have been sent to you via email, but if you've lost it, it can be recovered via the "forgot password?" link on the website.
- **TortoiseSVN**
  TortoiseSVN is a Windows client program for managing files using the Subversion (SVN) version control system. We'll be making extensive use of SVN this semester for managing our files. You'll probably want to download the 32Bit Installer program (first link in the list).
- **Microsoft Visio 2007**
  This software is very handy for creating class diagrams and flow charts. It is available from the same MSDNAA site at Visual Studio (see above).
- **Exsys Corvid**
  Corvid is what is known as "system shell" software that is used to build the knowledge- based systems that we will be studying in the last part of the semester.

Textbook

While it is not necessary for every person to have their own text, I STRONGLY urge every team to have at least one copy of the text. I will not be assigning chapters for you to read, but the book is a fantastic resource for learning how to do this stuff.



Gaddis, Tony and Kip Irvine. 2010. Starting Out With Visual Basic 2010, 5th Edition. Addison Wesley, 896p, 0136113400

NOTE: Do NOT buy an older edition of this book. Programming textbooks go out of date virtually by the time they are printed and the older version will lead you astray

Websites

The following websites will be useful in having a good experience in this course:

- **Microsoft's MSDN Library**
  This is the MSDN Library which contains very thorough documentation of all of the features of the Visual Basic programming language that we'll be learning this semester. In addition to reference material there are also some tutorials. While thorough, this documentation is admittedly targeted toward experienced developers and can be a bit technical and heavy on jargon. Despite this, it is strongly recommended that you spend the time necessary to learn how to make effective use of this website.
- **VB Forums**
  The VBForums website is an excellent place to ask questions of real people and get answers about how to do stuff using Visual Basic. If you do ask a question there, you might want to let people know that you are a "newbie" (i.e. a novice VB user), so that they will be more tolerant of really basic questions.
- **Stack Overflow**
  Stack Overflow has rapidly become THE place for asking development questions. Personally, I've found this to be one of the BEST places to get questions answered about how to perform specific tasks, especially when the official documentation is lacking in a particular area.
- **Microsoft's Developer Forums**
  These are Microsoft's official forums for developers. Searching through the responses here is typically a great way to find answers. Don't be shy about asking your own questions here.
- **Google**
  Hey, you use Google for almost all of your other questions in life, so why not programming? This should be one of your first stops whenever you get stuck on a programming project. A good hint is to include the word "tutorial" in your search queries, which will make it more likely for you to find information targeted at beginners.

Personal Integrity

First:

> *If I catch you cheating, or doing anything else dishonest, you will fail the course. Period.*

Second, that being said, ***I strongly encourage sharing and collaboration in most every aspect of the course***. That means that I think it's a smart idea for you to:

- Download code you find on the web (include the URL of where you found it and some notes about how you got there)
- Download your classmates' code and use it, even before an assignment is due
- Pay someone to help you write code
- Get code from upperclassmen or people in previous semesters
- Ask your neighbor to give you a hint on a question on a test that you're stumped on
- Use whatever notes, websites, books, or other materials you need to complete most any assignment or test

You'll note that many of the above behaviors would be considered "cheating" in many or most other courses. Here are some guidelines I'd like you to follow:

- **Never EVER copy without attribution**
  Even on tests, if someone or something helped you out, acknowledge it. Make notes in your code if you got it from someone or somewhere else. Copying without attribution is stealing and is a breach of integrity. If you got the code off of the web, there should be a URL and some notes about how you found it. If you paid someone to help you write it, say so.
- **Never copy without understanding**
  The point of the class is to learn and understand stuff. Since you don't get any grades on individual tests or assignments, it's pretty stupid to copy something that you don't understand. Think about it. What point could it possibly serve?
- **Be very hesitant to copy an ENTIRE project**
  While there's a lot to be gained by incorporating parts of your classmates' code in your own project, copying someone else's entire project doesn't really provide you much of a learning experience and wastes people's time.
- **Try to figure it out yourself first**
  90% of writing programs is learning how to write them, and this will stay the same throughout your entire programming career. Being a self-sufficent learner is one of the primary goals of the course.

> *Code re-use is a HUGE part of hacker culture. What hackers hate more than anything is not understanding stuff. I want you to get a sense for what it's like to be a part of the fun world of professional hackers.*

Okay, so what do I consider a breach of integrity worthy of failure?

- **Lying about anything to anyone in the class**
  It could be as trivial as the reason why you didn't show up for class or do your part of a group assignment. Everybody screws up sometimes. Don't compound the mistake by lying about it. We can forgive mistakes but it's VERY difficult to regain trust once it's broken. Swallow your embarrassment or fear and fess up.
- **Stealing anything–this includes copying without attribution**
  Stealing is just wrong, and since you have a blanket license to copy most any code you can find, there's no reason not to give people credit for the work they did. Passing someone else's original work off as your own is frankly disgusting.
- **Threatening, antagonizing, or intimidating anyone in our learning community**
  This is unacceptable behavior and will get you at least fired, if not sued in most every company you'd ever work for.

If you are in doubt about something, please ask your prof. Please feel free to come speak to your prof in confidence about anything in this course that troubles you. So far at JMU I've never had a problem with anyone's integrity (that I know about). Don't be the first group to ruin my perfect record. Thanks!

Grades and Evaluation

A Revolution in Education?

We take grades and the grading system for granted. Watch this:

http://www.ted.com/talks/view/lang///id/865

Simply Put: Grades Undermine Learning

The evidence to support this claim is ample, robust and has been growing for the past 40 years (cf. 2001–Deci, Koestner & Ryan–Extrinsic Rewards Reconsidered). While it's possible to use grades in a way that minimizes their negative influence on learning, rather than jump through those hoops, your instructor has decided to do away with them altogether. It's simpler, less work, and causes much less stress for everyone involved with the process. Incidentally, it's also a much more effective way to foster learning.

Yeah, but grades motivate me to study!

Yes, that's exactly the problem. They aren't really doing what you think they're doing. Watch this:

http://youtu.be/G59KY7ek8Rk

If you are genuinely interested in learning the material in this course, why do you need grades to motivate you? And if you aren't interested in the material covered in this course, why are you here? I mean really! Why would you pay money to do something you don't want to do?

I, your instructor, don't have any desire whatsoever to force feed this material down your throat. I love the material and I do this job because I truly love sharing my passion with others. I can be a good guide to you but I have no interest in dragging you along against your will. What does either of us gain by that? If you still don't know yet, whether or not you like this stuff, I'm also happy to engage with you to help figure out the answer to that question. If you give the material an honest effort and find you still don't like it, that, in my opinion, is a valuable use of both your time and mine.

Sure, but employers want people with good grades…

Maybe not. Research shows that recruiters frequently ignore GPAs or even select against people with high GPAs. Employers want independent problem solvers who know how to create and deploy technology effectively. They want people who know the difference between good work and crap. People who can figure out for themselves how to achieve desired outcomes. Why in the world would a potential employer care about the letter grade on a transcript if you've got a resumé chock full of real live software projects that you've completed? Even though grades are designed to be a proxy for showing what you can do, they are never as good as the real thing. Take advantage of your time in this class to build skills and products that will adequately show what you can do.

Okay, but at least the registrar wants a grade. What will you tell them?

Whatever you want me to (within reason). JMU doesn't force me to follow any sort of grading distribution or curve. JMU doesn't charge me $100 per A, $50 per B, $25 per C and so on. It has little impact on me whatsoever if I give all A's or all F's. Personally, I think you should give yourself an honest grade that reflects what you can do. I'm more than happy to help you arrive at that determination. At the end of the semester you and I will have a one-on-one meeting in which we discuss everything you've done this semester and figure out together what to tell the registrar.

That doesn't really sound fair…

Fair to whom? Concepts of fairness only really come into play in the context of a competition between people over scarce resources. We've already established that there's nothing scarce about the supply of A's in the world. The only competition in which there's a potential for unfairness is in competitions for things like jobs, internships, admission to graduate school, and scholarships. The truth is that none of these competitions are really fair anyway. Furthermore, my way of distributing grades is, while unorthodox, no less valid than anyone else's method. For an accessible discussion on the validity of grades I refer you to Alfie Kohn's 2002 article on grade inflation in the Chronicle of Higher Ed. When you really dig into the dank underbelly of how grades are distributed, you're going to find that most of us faculty are just making it up as we go along anyway. Rather than bicker with you over something about which I don't care, my choice is just to let you choose for yourself what you want. Instead, I'll spend my time sharing really juicy content with you, and trying to find out what makes you tick.

Alright already, so if there are no grades, how will I know if I'm doing well or not?

Now we're talking.

The whole point of my approach is to make you forget about grades altogether and instead focus on learning how to embrace computational thinking in your life. So, let me try to use some web media to help shed some light on this rather unorthodox approach. The first question is, what should we be studying? Watch this:

http://youtu.be/cL9Wu2kWwSY

Okay, so this video is guilty of a bit of hyperbole, a little out of date, and should be taken with a grain of salt, but in my mind here's the money quote:

> *We are currently preparing students for jobs that don't yet exist, using technologies that haven't been invented, in order to solve problems we don't even know are problems.*

This goes double in a programming class. I've been programming for 10 over years now, and I've seen the state of technology and the practices of development change completely several times within that time frame. What that means is that we have to have a sense of humor about the content on the syllabus. Although the topics serve as an interesting framework around which we can spend our time together, it would be somewhat naïve of us to really believe that knowing how to do any of the things there is really going to serve us well for more than three or four years. As such, I believe that a practical and rational set of goals for you to have when you take this course are:

1. Figure out if you really like programming
2. Figure out what your strengths and weaknesses are in this area

If it turns out that you love doing this and are also likely to be good at it, you may be in what Sir Ken Robinson calls your Element. I think all of your "higher education" should be about finding your element. But don't let me convince you, have a listen to Sir Ken:

http://www.ted.com/talks/view/lang///id/66

So, if you take what Sir Ken says seriously (and I do), here's what I think it means for us in this class. There are three areas in which your work in programming might be evaluated:

1. **Did you enjoy it?**
   I don't think you need my help in figuring out whether or not you like this stuff. You should be able to evaluate that for yourself.
2. **Is it effective?**
   Good programs, at least at the surface level, should be relatively easy for you to recognize without my help. I mean, a piece of software is effective, or it isn't. Although we'll talk about this some, I generally expect that you can decide this for yourself as well. I can give you pointers in how to develop a better sense for aesthetics in your own creations.

3. **Is it technically sound?**
   Under the surface however, there are some technical aspects of software quality that may only be apparent to a trained eye. I don't expect you to know anything about these aspects and that's where I can bring the greatest amount of my own expertise to bear in this class. This is where I see my efforts as having the greatest impact. My goal is to train you to be able to evaluate the technical aspects of software on your own.

At the end of the day, the responsibility for determining the quality of your work is your own and here's why:

> *If you can't, on your own, tell the difference between high and low quality work, then you will never ever be able to reliably produce high quality work by yourself because…how would you know?*

Lastly, I see my role as your instructor as a guide and cheerleader to help you find your passion. Actually, I really see myself as more like your personal trainer. So what do I mean by that?

1. You have hired me to help you get stronger at software development
2. I can show you what exercises to do, but you are the one who has to put in the time and do the heavy lifting
3. I still get paid, regardless of what you do, so why not get your money's worth?

I really love this stuff, so why not put in everything you've got and get something out of what this class has to offer.

Important Dates and Policies

At the behest of the registrar, a list of dates you may wish to take note of:

- Tuesday, January 17th: Last day of add/drop
- Thursday, January 26th: Last day to add a class with Department Head signature
- Friday, January 27th: Last day to withdraw from JMU with charges canceled

So if I scare you off, get out early. Or conversely, if I turn you on, join us soon!

My academic integrity policy is different from JMU's standard policy, but I will adhere to JMU's standard policies listed on the JMU Syllabus Information for Students page with respect to add/drop, disability accommodations, inclement weather and religious accommodations.

The Prof

My name is Morgan. I grew up in Lynchburg, VA and was an undergrad at the University of Richmond where I studied Sociology and Leadership Studies (with a minor in physics). I lived and taught English to junior high kids in rural northern Japan for 5 years after college. I got married there and had two daughters. In 2001 I moved back to the US with my family and started a PhD in Infomation Systems at New Jersey Institute of Technology. I came to JMU in 2006 and have been teaching here ever since.

My research mainly involves coming up with pedagogical alternatives that maximize student motivation and learning. Being a tech geek, web-based technology plays a pretty heavy role in what I came up with.

My favorite part of my job is getting to hang out with students and play with technology. Feel free to come see me any time. My info:

Office	ISAT/CS 124
Office Hours	Make an appointment
Office Phone	540-568-6876
Mobile	973-495-7736 (calls and texts are ok within reason)
Email	bentonmc@jmu.edu
Facebook	http://www.facebook.com/morgan.benton