ISAT 252: Programming & Problem Solving – 3 credits (Sec 1, 2, 4)
Spring 2013

COURSE AND INSTRUCTOR INFORMATION

Meeting Times: MWF, 9:05am - 9:55am (Sec 1)
MWF, 10:10am - 11:00am (Sec 2)
MWF, 12:20pm - 1:10pm (Sec 4)
Meeting Location: ISAT/CS 337
Instructors: Morgan Benton, ISAT
Office: ISAT/CS 124
Office Hours: Monday Night Hacking Session, 8pm - usually midnight or 1am
... and I'm in my office *most other times*, so you can drop by whenever
Phone/SMS: 973.495.7736
Email: bentonmc@jmu.edu

NATURE OF COURSE CONTENT

COURSE DESCRIPTION

This course prepares students to analyze complex problems in integrated business, technology and engineering environments by applying **computational thinking** in the context of a **highly participative learning community** that spans multiple coordinated courses.

"Computational thinking" is a term coined by Jeannette Wing, Chair of the Computer Science department at Carnegie-Mellon University. According to Wing, computational thinking:

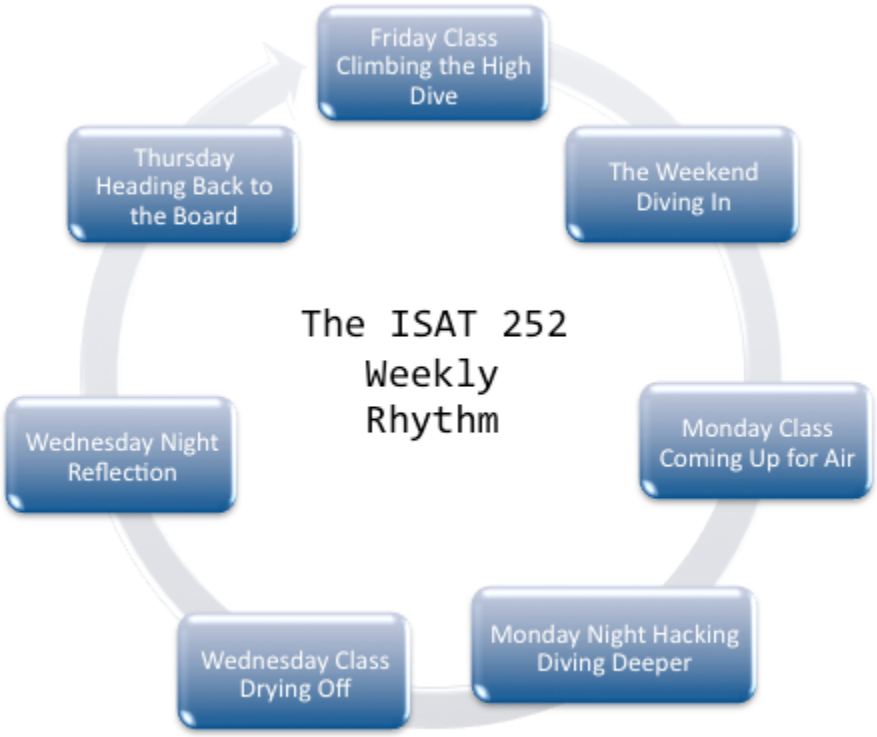
- is a way of solving problems, designing systems, and understanding human behavior that draws on concepts
- fundamental to computer science. To flourish in today's world, computational thinking has to be a fundamental part of the way people think and understand the world.
- means creating and making 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 that is useful in most 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.

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

SCHEDULE

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!

COURSE STYLE & DELIVERY

This course implements the **10 Principles of the Burning Mind Project** as its core value system. (http://www.burningmindproject.org/the-ten-principles/) As a result, the course is somewhat self-directed, blended (integrating online and in-class components), gift-oriented, and synchronously coordinated with ISAT 341 - Simulation and Modeling (Radziwill) and HON 300 - Quality & Process Improvement in Action (Radziwill/Simmons). What this means to you is that you will have the opportunity to *serve* teams in these other courses as subcontracted programmers to deliver real value to the clients those student teams are serving in our community.

GOALS OF THE COURSE

COURSE OBJECTIVES

Many problems may be solved with computer programming 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

METHODS OF EVALUATION

The purpose of this course is to *produce artifacts* that provide value to real people, both inside and outside of our learning community. As a result, participation and professionalism are essential - when you are interacting with your team, your instructors, and (possibly even) the clients for your programs.

GRADING

Because the course supports many different paths through the learning objectives based on your familiarity with programming, evaluation is portfolio-based, and will be tied to the personal collection of artifacts you have compiled to provide evidence of your learning and your contributions to our learning community. **Ultimately, the letter grade will reflect a subjective assessment of your professionalism as you contribute to our learning community and develop your portfolio.** It is your job to recommend the grade you think you should receive in a one-on-one interview with your Prof that will be scheduled during finals week, based on the evidence you provide in your portfolio. There will be a multitude of opportunities throughout the semester to interact and engage with the professor and other students, continually improving the contents of your portfolio.

REQUIREMENTS & POLICIES

REQUIRED TEXTS:

None. We will not be using a textbook this semester. Programmers live online and everything you need to know can be found there. One of the most important skills to develop in this class is the ability to find good tutorials, and to be able to figure out the difference between a "good" tutorial and a "bad" one. But here are a number of other 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.

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.

GitHub: GitHub is a website where you can host and share programming project files using the Git version control system. We'll be making extensive use of Git this semester for managing our files. You'll really want to make one of your first priorities going through a Git tutorial.

Websites: The following websites will be useful in having a good experience in this course. Don't limit yourself to these sites, though, there are many, many, many, many good resources to be found online.

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.

Code Academy
Hands down one of the best places on the web to learn how to program. They have courses in HTML5, CSS, Javascript, jQuery, Python, and Ruby.

Lynda.com
I haven't used this one personally, but I've heard from some students that it is good.

Khan Academy
Looks very promising to me…

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.

ADD/DROP DEADLINES

All of the dates related to adding, dropping, and withdrawing from this course are in the JMU catalog and are posted on the University Registrar's web site. **YOU ARE RESPONSIBLE FOR KNOWING THESE DATES.** Exceptions in the form of either WP or WF may be granted at the instructor's discretion.

COURSE POLICIES AND PROCEDURES

Attendance/Being Late
Participation is one of the primary values of our learning community. As a result, it's important for you to attend and participate in class, group meetings and other sessions. Missing class regularly will negatively impact your ability to get stuff done, learn the material, complete projects, and successfully contribute to the community itself. If you need to be absent for an extended period, please let your Prof know.

Special Needs
To request disability accommodations, many professors ask that you register with Disability Services and to provide them with an Access Plan letter outlining your accommodations. However, this class is part of a very special learning community, where we recognize that everyone has their own strengths and weaknesses, as well as their own unique gifts that they can contribute to the community. Your community is here to support you, and so long as you actively participate, there's really no way to get behind.

Personal Integrity & Honor Code
You are expected to abide by the JMU Honor Code at all times, which is posted on-line at http://www.jmu.edu/honor/code.html. However, here's my interpretation of that Honor Code specific to our class and our learning community.

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-sufficient 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!

Other policies can be found here: http://www.jmu.edu/syllabus