ISAT 252  
Programming and Problem Solving   
Spring 2010 Syllabus

# Introduction

“[**Computational Thinking**](http://www.cs.cmu.edu/~CompThink/)” 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.

# Course Objectives

By the end of this course you should be able to:

1. Demonstrate a basic familiarity with computer hardware and explain the function of the main components of a computer
2. Recognize when a problem in science, business, or engineering is amenable to a software solution
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 using a modern programming language such as Visual Basic .Net
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 such as use of online and offline reference materials, search strategies, online forums, and integrated debugging tools
9. Document code appropriately to increase ease of maintenance and understanding
10. Read and understand code that others have written and be able to predict the results of running that code and/or re-use/incorporate others’ code when appropriate
11. Describe what a knowledge-based system is and the types of problems and solutions to which it is best suited
12. Describe the architecture of a typical expert/knowledge-based system, describe the methodology used to develop expert/knowledge-based systems, and use the methodology to develop an expert system
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 correct decision trees in an applied problem-solving scenario
15. Evaluate a given problem within its social context and identify the most appropriate paradigm within which to develop a software solution

# Skills to Develop

Over the years, programmers have developed approaches to solving problems that encompass a broad range of human behaviors including, but not limited to:

* best practices for communication and teamwork,
* standards for documentation,
* social norms for how to ask questions, and yes,
* writing code.

As such, the goals of this course touch on a wide range of areas that can be roughly categorized into professional competencies, and programming skills.

## Professional Competencies (aka “soft skills”)

These skills are vital to being successful in your career, no matter what you do, but are rarely addressed explicitly in any course.

### Communication

By the end of this course you should:

* ***demonstrate*** ***ability to select the most appropriate medium*** for communication, e.g. email, phone, flow chart, screen diagram, class diagram, text message, online forum, IM, face-to-face, etc.
* ***demonstrate*** ***ability to communicate clearly*** and appropriately given the chosen medium
* ***communicate*** in a way that demonstrates ***commitment to and consideration of one’s team members*** and classmates (including the instructor)
* ***know*** ***how to route and organize your communications*** so as not to miss important messages from colleagues
* ***overcome the temptation to avoid telling your team/instructor that you are stuck*** and don’t know how to proceed

### Teamwork

By the end of this course you should:

* have ***developed a niche for yourself*** on your team
* have devoted significant energy to ***learn how to work effectively with the team you’ve been dealt***
* ***establish solid and reliable lines of communication*** among team members
* ***develop effective means of managing group resources***, such as papers and files
* ***have figured out how NOT to give into the temptation to procrastinate*** that can be exacerbated by the diffuse responsibility shared by team members
* have behaved in a way that will allow you to ***earn a solid and positive recommendation*** from your teammates
* understand how what you do reflects on the team, and how what your team does reflects on you

### Attendance

By the end of this course you should:

* ***have developed a reputation for showing up on time*** to class and to team meetings, or, barring that
* ***have developed a reputation for timely communication about where you are*** and why you’re not showing up on time

### Documentation

By the end of this course you should:

* have learned to ***generate a “paper trail”*** to document and demonstrate your and your team’s efforts to accomplish tasks
* ***have developed a habit*** of documenting your time and effort on a task
* ***have developed a habit of clear documentation*** (emphasis on clear)

### File Management

By the end of this course you should:

* ***never have to say “I don’t have my files”*** because (pick one):
  + I forgot/lost my jump drive
  + my computer crashed
  + I was working on a lab computer and the files are no longer on the desktop
* know how to use a version control system to ***always have an available copy of every version of any given file***
* understand how to use a version control system to ***collaborate effectively with teammates on a shared file*** or other electronic resource

### Professionalism

By the end of the course you should:

* ***develop a positive attitude*** that contributes to a productive work environment
* ***be in the habit of thinking ahead***, anticipating the needs and actions of others and doing what you can to meet those needs, even if not explicitly requested

## Programming Skills

Ostensibly, these are the “hard skills” that you’re taking this course to acquire. However, not everyone was born to be a programmer. For most students, this represents their first foray into the world of programming. As such, some reasonable personal goals for the semester might be:

1. To find out what the discipline of “computational thinking” has to offer you
2. To find out whether or not you have the temperament, attitude, and aptitude to be a really good programmer
3. To make a decision about whether or not you want to pursue this field any further

If you ***work hard*** and can achieve the three goals above, then you will most likely have had a very productive and rewarding semester. Please note, however, that if you half-ass this course, you may find answers to the questions these goals raise, but they may be the wrong answers. So, without further ado, here are the programming skills/concepts you should acquire:

1. Basics
   1. What is “computational thinking?”
   2. What is a program?
   3. What is a computer?
      1. Hardware components
      2. Operating system
      3. Software components
   4. How does a computer execute a program?
      1. Computer circuitry and Boolean logic
      2. Machine language vs. Human language
2. Algorithmic Thinking
   1. What is an algorithm?
   2. What kinds of problems have algorithmic solutions?
   3. How do you plan an algorithmic solution to a problem?
      1. Flow charts
      2. Pseudocode
      3. Paper and pencil screen diagrams
      4. Storyboards
      5. Iteration
   4. How do you implement an algorithmic solution to a problem?
      1. Plan
      2. Code
      3. Test
      4. Deploy
      5. Maintain
      6. Iterate
   5. What are the different paradigms for implementing algorithmic solutions?
      1. Procedural programming
      2. Object-Oriented programming
      3. Declarative programming
   6. What tools do programmers use to implement algorithmic solutions?
      1. Standards and Conventions
      2. Paper and pencil
      3. The web
      4. Other people
      5. Visio
      6. Visual Studio ***Integrated*** Development Environment (IDE)
         1. GUI form designer
         2. Code editor
         3. Documentation
         4. Compiler
         5. Debugger
         6. File manager
         7. Collaboration tools
      7. Exsys Corvid
      8. Subversion
3. Object-Oriented Programming (OOP)
   1. What are objects?
      1. Properties (the nouns)
      2. Methods (the verbs)
   2. What are classes?
   3. Is everything an object?
   4. How do you plan an algorithmic solution with objects?
      1. Class diagrams
      2. Hierarchy charts
   5. How do all of these planning documents translate into actual code?
   6. What are the basic, programming constructs of object-oriented programming?
      1. Data types
         1. Numeric
            1. Integer
            2. Rational
         2. Text
         3. Boolean
         4. Complex types
         5. Collections
            1. Arrays
            2. Lists
      2. Symbols
         1. Constants
         2. Variables
         3. Naming conventions
      3. Calculations
      4. Flow Control
         1. Events
         2. Decisions
         3. Iteration (Loops)
         4. Errors
            1. Compilation errors
            2. Run-time errors
            3. Logic errors
         5. Exceptions
      5. I/O (input and output)
         1. Input
            1. Manual
            2. Streams

Files

Network

* + - * 1. Databases
        2. Devices and sensors
      1. Output
         1. Screen
         2. Audio
         3. Streams

Files

Network

* + - * 1. Databases
    1. Data
       1. Text Files
          1. Unstructured
          2. Delimited
          3. Markup

XML

HTML

* + - 1. Binary Files
      2. Databases

1. Declarative Programming—Knowledge Based Systems (KBS)
   1. How is declarative programming different from OOP?
   2. What are the core concepts of declarative programming?
      1. Declarative knowledge
      2. Facts
      3. Relationships
      4. Rules of inference
      5. Modus Ponens
      6. Truth tables
   3. What is artificial intelligence and how does it relate to KBS?
   4. What kind of problems are KBS designed to solve?
   5. What do you need to build a KBS?
      1. Knowledge engineer
      2. Domain expertise
      3. End user
      4. Client
   6. What is the architecture of a KBS?
      1. User interface
      2. KBS editor
      3. General knowledge base
      4. Case-specific knowledge base
      5. Inference engine
      6. Explanation system
   7. What is the process for developing a KBS?
      1. Define the problem
      2. Acquire domain knowledge
      3. Represent the domain knowledge
      4. Design a solution
      5. Code the solution
      6. Test and correct
      7. Iterate
   8. What tools do you use to develop KBS?
      1. Exsys Corvid
2. Getting Unstuck
   1. Why are you likely to spend A LOT of time being stuck?
   2. How do you keep from getting stuck?
      1. Start early
      2. Plan, plan, plan
      3. Tackle the problem in small, manageable chunks
      4. Test the small chunks before integrating into the whole
   3. How do you realize that you’re stuck?
   4. How do you get unstuck?
      1. Take a break
      2. Use debugging tools and techniques
         1. Test output
         2. Watch variables
         3. Break points
      3. Ask someone who knows
         1. Instructor/TA
         2. Friend
         3. Online forums
      4. Look it up
         1. Technical reference, e.g. MSDN Library
         2. Tutorial reference, e.g. textbook, tutorial website
         3. Google it
      5. Describe your problem in words
   5. How do you choose a method for getting unstuck?
      1. Cost
      2. Urgency
      3. Availability
      4. Efficiency/Effectiveness
   6. Do these techniques apply to more than just programming?
3. Becoming a Better Programmer
   1. Explore your environment—don’t limit yourself to what you are “taught”
   2. Use common sense—if you want to do it, chances are someone else did too
   3. Read other people’s code—read both “good” and “bad” code and figure out how to tell the difference
   4. Use other people’s code—incorporating others’ solutions into your code is non-trivial and will make you a better programmer
   5. Have code review sessions—discussing alternative solutions and methodologies is one of the fastest ways to get better
   6. Evaluate your plans—did your solution look anything like your plans? if not, why, and how can you make your planning process more effective?

# Useful Websites

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

* <http://isat252.isat.jmu.edu>  
  This is the “official” ISAT 252 website that is shared among all sections of the course. In particular, this will be the place to go to find all course videos.
* <http://isatgeek.basecamphq.com>  
  We will NOT be using Blackboard in this course. Instead we’ll be using a professional online project management system called [Basecamp](http://basecamphq.com/?referrer=isatgeek). Your username is your eID, and your password is your JAC card number. You are ***strongly recommended*** to post any questions you have to the message boards hosted here.
* <http://msdn.microsoft.com/en-us/library/default.aspx>  
  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.
* <http://www.vbforums.com/>   
  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. NOTE: this is only one example of an online forum for asking questions. There are many others and you are encouraged to find the one(s) that is best for you.
* <http://www.google.com>  
  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.

# Text

It is recommended that every group buy at least one course textbook. If you plan to take ISAT 340 (required if you choose IKM as one of your ISAT sectors) we will likely use the same textbook for that class, so DON’T sell it back at the end of the semester.

[Bradley, J.C. (2008) *Programming in Visual Basic 2008*. McGraw Hill. 978-0-07-351720-9](http://www.amazon.com/gp/product/0073517208?ie=UTF8&tag=morphatic-20&linkCode=as2&camp=1789&creative=9325&creativeASIN=0073517208)

# 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.  
  <http://www.softronix.com/logic.html>
* **Visual Studio 2008 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. [bentonmc@jmu.edu](mailto:bentonmc@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.  
  <https://msdn03.e-academy.com/elms/Security/Login.aspx?campus=jmu_cs>
* **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).   
  <http://tortoisesvn.net/downloads>
* **SVN Integration into Visual Studio 2008**You can integrate the SVN tools directly into Visual Studio. The files you’ll need to do so as well as instructions for installing them can be found via the link below. You’ll want to install Visual Studio 2008 and TortoiseSVN before you install these tools.  
  <http://garrys-brain.blogspot.com/2007/07/tortoisesvn-and-visual-studio.html>
* **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.  
  <http://www.exsys.com/DownloadStart.html>

# Course Description (from the JMU Catalog)

Use of formal logic to represent and assess properties of natural language constructs important to scientific inquiry, with application to the development and use of knowledge-based systems; introduction to procedural programming and its uses for producing and tailoring information systems supporting scientific, technical and business problem solving. *Prerequisite: Sophomore standing or permission of instructor.*

# Grading

Over the course of the semester you will be responsible for accumulating a portfolio of high quality of work. You will get very specific guidance on what kind of evidence needs to be in your portfolio, however, choosing the items that represent your best work will be your own responsibility. At the end of the semester you will prepare your portfolio and then schedule time to present your portfolio to your instructor. Your instructor will assign your grade based on the quality of the portfolio. At a minimum, the portfolio will need to include evidence that you can use to make an argument that you have performed reasonable to high quality work in the hard and soft skills outlined earlier in this syllabus. More information will follow shortly.

# Weekly “Hacking” Sessions

Beginning the first week in the semester there will be a weekly “hacking” session held in ISAT/CS room 337 on Mondays from 8pm to midnight. Your instructor and/or TA will be present during this session to work with you on anything class-related. These sessions are designed to be a fun, relaxed, and very informal. Feel free to come by get extra help on stuff you’re having trouble with, to work with your prof/TA on some of the projects they’re working on this semester, or perhaps to learn some “extra” tricks or secrets on how to be an effective programmer. This is definitely not required, but is the coolest Monday night hang out spot around so I’m sure you won’t want to miss it.

# Academic Integrity

Any breach of integrity will be grounds for immediate failure of the course. I take this very personally. However, I ***strongly*** encourage sharing with attribution. The DRY principle (Don’t Repeat Yourself) is one of the most celebrated cultural elements in the programming community. This includes incorporating other people’s code, where appropriate, and being able to use other people’s code effectively is an extraordinarily valuable skill. So, in other words, I encourage collaboration, but make sure to attribute credit to those people who (knowingly or unknowingly) helped you out. The one caveat is I strongly ***discourage*** you from copying without comprehension. It would be pretty stupid to turn in a bunch of copied projects on which you learned absolutely nothing.

# Important Dates

Tuesday, January 19th, 2009—Last day for add/drop via eCampus  
Thursday, January 28th, 2009—Last day to add a course with signatures  
Friday, January 29th, 2010—Last day to withdraw from JMU with a refund  
Friday, March 19th, 2010—Last day to withdraw with “W” grade or change course credit options

# The Prof

Name: Morgan Benton  
Office: HHS 3224  
Office Hours: Mondays 2-4, or other times by appointment  
Office Phone: 540 568 6876  
Cell Phone: 973 495 7736 (calls and texts are ok within reason)  
Calendar: my [Google calendar](http://www.google.com/calendar/embed?src=morgan.benton%40gmail.com&ctz=America/New_York&pvttk=88fc38002e9de1ffc13d0f8446be94dc) is usually pretty accurate   
Email: [bentonmc@jmu.edu](mailto:bentonmc@jmu.edu)  
Facebook: [yes](http://www.facebook.com/home.php#/profile.php?id=636971996&ref=profile)  
Twitter: <http://twitter.com/morphatic>   
AIM: mcbenton17 (but I’m almost never online—not really a big IM person)

I’m usually on campus 9-7 Mon-Sat. are research days, so I’d appreciate your help in keeping that “me” time clear, but otherwise meeting with students is the most enjoyable part of my job.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Week/Theme | | Monday | Wednesday | Friday |
| 1 | Course Intro  Computer Hardware  Bits, Bytes, and Boolean Logic | 1/11 | 1/13 | 1/15 |
| Topics: Course Intro Hacking Session—8PM | Topics: Computer Hardware | Topics: Boolean Logic  Assign Lab #1—Logic Gates |
| 2 | 1/18 | 1/20 | 1/22 |
| **MLK Day—No classes** Hacking Session—8PM | Lab #1 Due: 9AM  Topics: Grade Lab #1 | Topics: OOP Intro  Assign Lab #2—Visual Studio Intro |
| 3 | Object Oriented and Procedural Programming | 1/25 | 1/27 | 1/29 |
| Lab Day  Hacking Session—8PM | Lab #2 Due: 9AM  Topics: Grade Lab #2 | Topics: Classes  Assign Lab #3—Your First Object |
| 4 | 2/1 | 2/3 | 2/5 |
| Lab Day  Hacking Session—8PM | Lab #3 Due: 9AM  Topics: Grade Lab #3 | Topics: Planning Documents  Assign Lab #4—Planning a Project |
| 5 | 2/8 | 2/10 | 2/12 |
| Lab Day  Hacking Session—8PM | Lab #4 Due: 9AM  Topics: Grade Lab #4 | Topics: File Management with Subversion  Assign Lab #5—Implementing a Plan |
| 6 | 2/15 | 2/17 | 2/19 |
| Lab Day  Exam Review  Hacking Session—8PM | Lab #5 Due: 9AM  Topics: Grade Lab #5  **Midterm Exam #1—6PM** | Topics: Grade Midterm #1  Assign Lab #6—Extending Your Reach  Assign Semester Project |
| 7 | 2/22 | 2/24 | 2/26 |
| Grade Midterm #1  Hacking Session—8PM | Lab #6 Due: 9AM  Topics: Grade Lab #6 | Topics: Debugging  Assign Lab #7—String Processing |
| 8 | 3/1 | 3/3 | 3/5 |
| Lab Day  Hacking Session—8PM | Lab #7 Due: 9AM  Topics: Grade Lab #7 | Topics: Code Review  Assign Lab #8—Working with Files |
| SB | 3/8 | 3/10 | 3/12 |
| **Spring Break!** | | |
| 9 | 3/15 | 3/17 | 3/19 |
| Lab Day  Hacking Session—8PM | Lab #8 Due: 9AM  Topics: Grade Lab #8 | Topics: Mid-semester Status Check  Assign Lab #9—Control Structures & Collections |
| 10 | 3/22 | 3/24 | 3/26 |
| Lab Day  Hacking Session—8PM | Lab #9 Due: 9AM  Topics: Grade Lab #9 | Topics: Getting a Job in a Programming Field  Assign Lab #10—Declarative Programming Intro |
| 11 | Declarative Programming and Knowledge Based Systems | 3/29 | 3/31 | 4/2 |
| Lab Day  Exam Review  Hacking Session—8PM | Lab #10 Due: 9AM  Topics: Grade Lab #10  **Midterm Exam #2—6PM** | Topics: Grade Midterm #2  Assign Lab #11—Decision Rules |
| 12 | 4/5 | 4/7 | 4/9 |
| Grade Midterm #2  Hacking Session—8PM | Lab #11 Due: 9AM  Topics: Grade Lab #11 | Topics: KBS Development  Assign Lab #12—Decision Trees |
| 13 | 4/12 | 4/14 | 4/16 |
| Lab Day  Hacking Session—8PM | Lab #12 Due: 9AM  Topics: Grade Lab #12 | **ISAT Senior Symposium**  Assign Lab #13—Corvid Program |
| 14 | 4/19 | 4/21 | 4/23 |
| Lab Day  Hacking Session—8PM | Lab #13 Due: 9AM  Topics: Grade Lab #13 | Declarative Programming Wrap-Up |
| 15 | Project Presentations Course Wrap-up | 4/26 | 4/28 | 4/30 |
| Project Presentations  Hacking Session—8PM | Project Presentations | Course Wrap-Up  Projects Due |
| Final | | 5/3 | 5/5 | 5/7 |
| Exam Review | **Common Final: 6PM** |  |