Syllabus Highlights/Class Expectations

1325 Object-Oriented Programming

Fadiah Qudah

University of Texas at Arlington

TOC

- Syllabus Highlights/Class Expectations
- General Information
- Virtual Machine
- C++ Standards
- Introduction
- Sample Program

SYLLABUS HIGHLIGHTS

Grading

Homework	7%
Quizzes	20%
Project	10%
Exam 1	10%
Exam 2	11%
Exam 3	12%
Final	30%

Grading

- **DO NOT** ask me to email you your numerical grade at any point in the semester
  - You can calculate this yourself using the information on the previous slide
  - If you have any questions on how to calculate it, stop by my office hours to ask me how to do it-NO GRADES WILL BE RELEASED BY EMAIL
  - The syllabus also mentions what letter grade corresponds to a numerical grade

Time Expectations

- Expected: 10 hours per week outside of class.
  - Some people spend 5 hours/week and get an A.
  - Some people spend 20-30 hours/week and fail.
- Programming assignments will be time consuming.
- This is a class where:
  - Falling behind is easy.
  - Catching up is hard.
- Attention to detail will be crucial.
  - **Code that is 99% correct is 0% useful.** (yes, you will be hearing this a lot in this class).

Re-grading Policy

- Send the TA an email/go to their office hours asking about any grading questions (since they are they one that do they grading)
- If you have further questions about the grading or feel you do not agree with the TA’s grading, feel free to ask me

Re-grading Policy

- Any request for re-grading (for an assignment or midterm exam) must be made within 5 days of receipt of that grade.
- Any request for re-grading the final exam must be made within 3 days of receipt of that grade.
- Re-grading can lead to a higher or lower grade, depending on grading errors that are discovered.
- Usually, around final exams time (when people try to get extra points, to make the threshold for a better grade) students ask us to regrade exams and assignments from two or three months ago.
  - At that point they get shocked to hear about the re-grading policy.

Assignments

- You must submit on Canvas.
- Late penalty: 20 points per hour.
  - No exceptions, except for medical/personal emergencies documented in writing.
  - Network/computer crashes will not be accepted as an excuse.
- Every semester, some people get 0 in an assignment (or more), because they submitted the wrong file on Canvas.
  - Then they ask for leniency, because they did all the work, and just made a silly mistake in submitting.
  - Sorry, there will be no leniency on this issue.
  - Verify your submission every time.

Assignments

- I don’t give out HW solutions
  - If you have questions about HWs, feel free to stop by my office hours or make an appointment
- I give **difficult HWs but make them a small percentage of your grade** so you can LEARN without harming your overall grade
  - This way, you can actually LEARN (the point of a university class)

Regarding Submission Problems

- If, for whatever reason, you cannot submit on Canvas, then you can send us your assignment by e-mail.
- In that case, e-mail **(before the deadline)** your submission files to me and the teaching assistant.
  - If you e-mail us after the deadline, you still get the late penalty.
- Use your UTA e-mail, so that you can prove that you sent your message on time.
- Check with us ASAP to make sure we received your e-mail.
- You will still need to provide ASAP very convincing documentation that you really had problems with Canvas.

Exams and Quizzes

- Most questions will ask you to write code or answer T/F questions
- Exams will closely mirror quizzes
  - Quizzes will be like smaller scale exams
- Some questions may ask you to read code and:
  - Write code
  - Fix code
  - True/false (w/explanation)
  - Write an equation
  - Predict what code will do
  - Write a short answer
  - Identify code that would actually run

Exams and Quizzes

- The material will be heavily based on the homework assignments and the class material (slides/class discussion/code)
- If you do your homework assignments and revise class material, the quizzes (and exams) should not be too difficult
- **DO NOT JUST SHOW UP TO TAKE QUIZZES- AUTOMATIC 0 IF YOU DO THIS**

Project

- There will be a project in this course
  - You will pick a real world problem and code it using object-oriented principles
  - I will release more details about this later in the semester

Talking in Class

- You guys are adults-**DO NOT TALK IN CLASS**
  - **It severely distracts me and causes me to lose my concentration while teaching**

- If talking becomes an issue, I will implement a seating chart and start taking points off your final grade
  - Just don’t talk in class

Syllabus

- A link to the syllabus is posted on Canvas.
- **You are responsible for reading fully and understanding what the syllabus says. If you have any questions or doubts, please ask me**
- At the end of the semester, when grades are posted, it is common to get complaints from students, saying that they were not aware of various aspects of the syllabus, that severely hurt their grade.
  - My only response will be to point them to this slide.

Getting Help

- My office hours are posted on the website
  - I am here to help you guys
- If you can not make it to office hours, please make an appointment
- I am also available by email: fadiah.qudah@mavs.uta.edu
  - Make sure to include mavs- **DO NOT email fadiah.qudah@uta.edu**

Getting Help

- I’m pretty good about answering emails within 24-48 hours
  - If you do not get a reply in that timeframe, feel free to send a follow up email (I may have somehow missed the original email)
- If I am updating a grade or looking over an exam/quiz, I will send you an email to confirm that I received it, but I may not actually do the required action until later on
  - I keep all emails in my inbox until I finish the task required by the email and will send an email when it is completed
  - If a long period of time goes by without an email signaling completion of the task, feel free to send a follow up email

Getting Help

- Additionally, feel free to email the TA with questions
- **Do not expect responses to frantic queries in the last minutes before an assignment is due**.
  - I most likely won't be able to get to the email on time
- This strongly applies to debugging emails (see next page)

Debugging/Office Hours

- If you need your program debugged, send me **(and the TA)** an email with your program and a short description of the problem
  - Due to the number of debugging questions I normally get, **I split the emails with the TA**
  - Do not just send your program and expect us to fix it for you-the reply will point you in the right direction
  - Include what you think may be causing the problem

- **DO NOT** come to my office hours for me to debug a program
  - Debugging can take a considerable amount of time
    - I am essentially having to go through your whole code to find the error-this can take a lot of time
  - It is not fair to other students if I spend the whole office hour period debugging one program

Debugging/Office Hours

- If you come to my office hours with a debugging question, I will ask you to email the program (as stated on the previous slide)
- You are welcome to come to my office hours and work on your assignments
  - Some students like to do this so if they have a question about an assignment or concept covered in class when coding they can ask me
  - **IF A QUESTION BECOMES A DEBUGGING QUESTION, I WILL ASK YOU TO EMAIL THE PROGRAM (previous slide)**

Attendance

- Attending exams is mandatory.
  - Again, exceptions are made only for medical/personal emergencies documented in writing.
  - Transportation problems, malfunctioning alarms, not accepted as an excuse.
- If you do not attend lectures, you are still responsible for understanding the material.
  - Do not expect a private lecture during office hours or by e-mail.

Attendance

- If a student fails the class, I need to report to UTA if and when that student has stopped attending.
- This may have repercussions on the student's financial aid.
- If you want to prove attendance, keep submitting homeworks and keep coming to exams.
  - You can submit empty files, or blank exams, that receive zero points, but still prove attendance.
- **If you stop submitting homework and taking exams, I have to report that you stopped attending.**

Class Participation

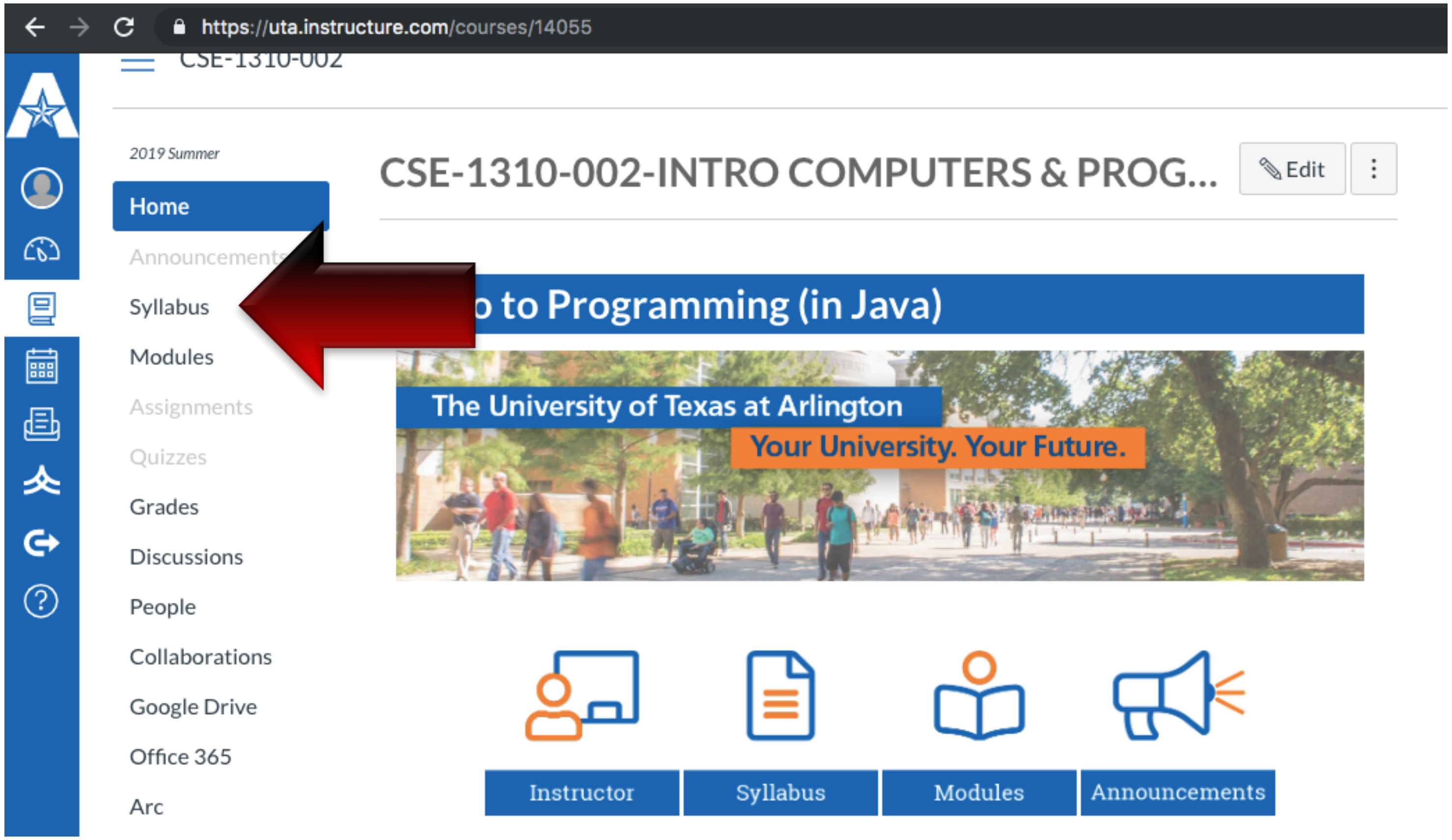
- Class participation is not part of the grading criteria.
- However, asking questions, and trying to answer questions can help you in understanding the material.
  - If you have questions and you do not ask in class, where are you going to get the answers?
- If you do not understand something, always feel free to raise your hand and ask a question.

Class Participation

- If a question is WAY beyond the scope of the class, I will say “Please stop by my office hours or email me and we can discuss this topic further”
  - I’m more than happy to discuss more advanced topics with you, but will have to do so in office hours
- I do this to avoid tangents and possibly confusing your fellow classmates that may not be as advanced as you are in the class

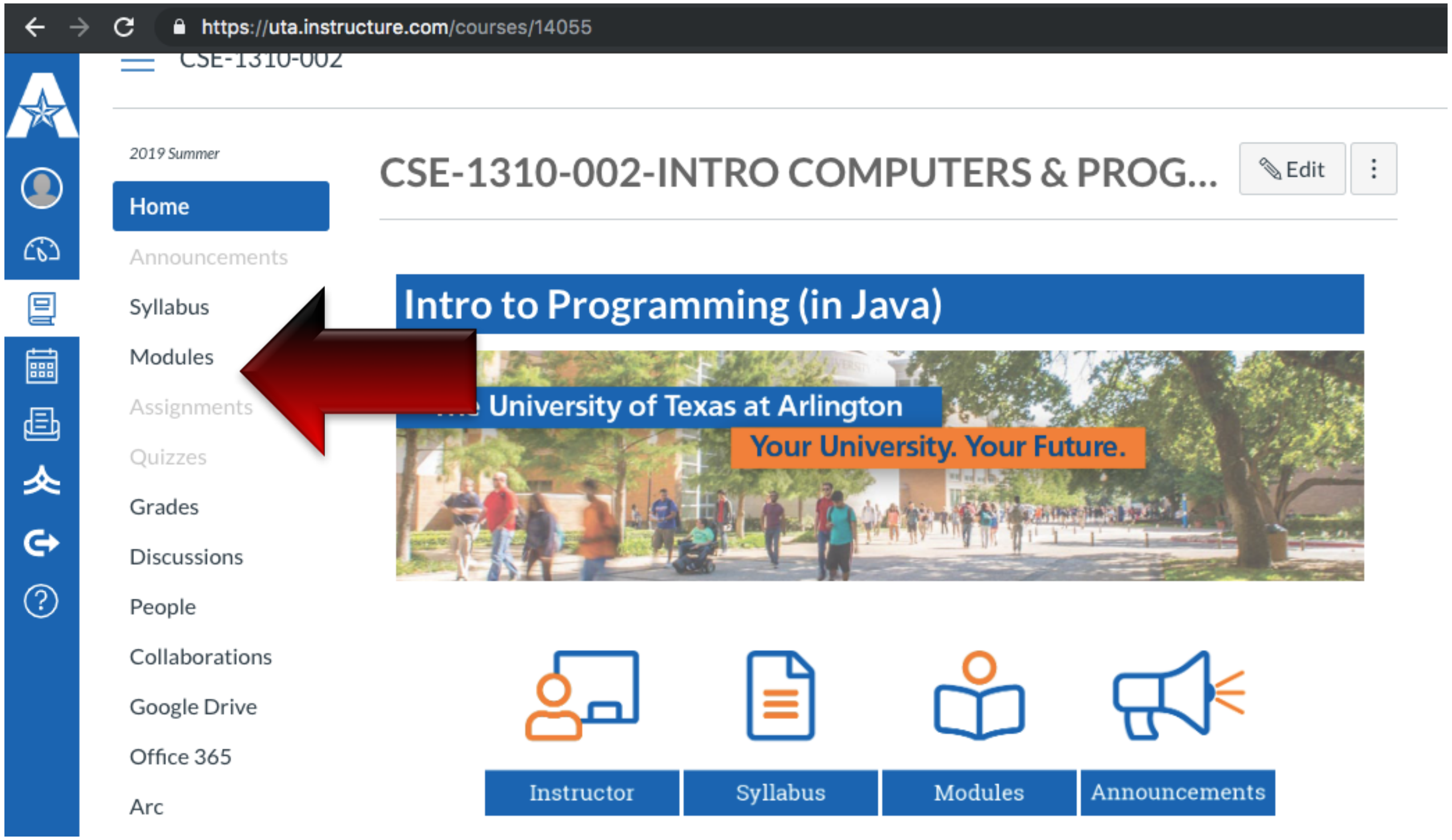
Course Website

- The link to the course website is posted on Canvas under the Syllabus tab



Course Material

- Slides and code will be uploaded to the Modules tab



GENERAL INFORMATION

About the Course

- This course is the third course aimed at teaching you to program
- The topic is difficult
  - It requires very precise, mathematical thinking.
- Unlike other courses, "getting most of the material" will not get you a good grade.
  - If you have even small gaps in the material, it will be very hard to do well in the class.
  - For a program to work, it needs to be 100% correct.
  - **Code that is 99% correct is 0% useful.**

About the Course

- At this point, I am expecting you to have programmed before
  - You should know basic and intermediate programming concepts
  - You should be familiar with problem solving

- I am expecting that you are familiar with Omega

- Find out what in-class techniques work best for you
  - Some students code along with me on a computer
  - Some students take notes
  - Some students simply watch and absorb

Goals of the Course

1. Introduction to object-oriented concepts
2. Introduction to class diagrams
3. Introduction to GUIs

Computer Use

- If you do not have access to a personal computer, please see me immediately so we can set up time for you to work on assignments/practice in the lab
- Not having a computer is NOT an excuse to not complete assignments-we want all students to have equal opportunities and will make sure lab resources are available for you

Study Tips

- Coding a little everyday is MUCH better than a lot occasionally
  - 20-30 minutes daily is better than 5 hours every 2 weeks
- YOU MUST ACTUALLY CODE TO LEARN-not just “review” code!
  - Passive (understanding) vs. Active (doing-producing code)
- Try to solve problems we do in class by yourself (only looking at the solution when you get stuck)
- Different websites online (codingbat.com)

Study Tips

- Learn how you learn!
  - Notice what works for you when you study
    - Some students like to code something multiple times
    - Some students prefer to always code something new
    - Some students like flashcards
    - See what works best for for you-it will make the rest of your computer science career much smoother
  - The one universal study tip for everyone is to CODE OFTEN AND FREQUENTLY (daily if possible)
    - Additionally, solving math word problems also helps you think more like a computer scientist

Study Tips

- Instead of taking notes ABOUT how something works, you may find it helpful to actually write a sample code and then put the notes about it in comments
  - It is more important to actually deal with code
  - Do not make the mistake of always worrying about the concept and avoiding code
    - Try doing the code and learning the concept that way

Study Tips

- Create your own problems to solve and code
  - Look around and see what interests you
  - Examples:
    - If you are interested in eating healthy during the semester, maybe make a program to count daily calories
    - If you are having trouble studying effectively, maybe make a program to help you organize your time
    - Etc.

- Most of the code I wrote for this class is a result of me creating my own little problems/scenarios and writing the code for it
  - The mental process you go through to recognize a scenario, solve it and code is pretty much what you will be doing your whole computer science career

Book

- The book is optional in this course
  - It is a good reference, but not necessary
- The department recommends the following website: https://www.learncpp.com/

- You will be tested on material from the slides, class discussions, code done in class and HWs
- I will also be showing you guys how to use online resources as a reference (since you will most likely be doing this in the future)
  - Knowing how to look up information is important!

Inevitable Feelings

- Every language learner (whether you are learning Java or French) feels frustrated at the beginning
  - The trick is to keep going
  - I’ve never had a student that didn’t give up not get it eventually
- As long as you **consistently** code, you will get there
  - **Actually write code**, don’t just think about code
- I’ve heard multiple people say they ”just woke up one day and got it”
  - This is actually due to consistently doing a little code every day-it all comes together

Approach

- I treat all programming classes as language immersion classes
  - This means I give you **a lot** of code in class
  - I obviously don't expect you to understand everything the first time you see it
- In each class session, I spend most of the time "translating" lines of code to you
  - Imagine that a complete program is a book and every line of code is sentence in that book
  - I'm "translating" these sentences to you

Approach

- I focus on:
  - Teaching you parts of the language
  - Syntax
  - A little “vocabulary” (mostly different methods/functions)
  - How to turn the real world around you into corresponding code
- My goal is for you to see and use enough code in different scenarios so you will be able to derive new code when you need

Approach

- Example in English:
  - If you understand the syntax and parts of speech, you know how to plug in

Nouns:	**Verbs:**
cat	to eat
dog	to drink
bird	to dance
…	…

Syntax example:

The _________ likes _________.

noun *verb*

We can have an infinite list of nouns and verbs and plug in and make a working sentence.

Approach

- Example in English:
  - If you understand the syntax and parts of speech, you know how to plug in

Nouns:	**Verbs:**
cat	to eat
dog	to drink
bird	to dance
…	…

We can have an infinite list of nouns and verbs and plug in and make a working sentence.

Syntax example:

✓ The ___**cat**___ likes ___**to eat**___.

noun *verb*

Syntax example:

✗ The ___**to eat**___ likes ___**cat**___.

noun *verb*

Approach

- I encourage you to type out class code yourselves to learn
  - Copying and pasting code doesn’t help you
  - I know some of you will do this anyways, but just know it helps you more to type out the code by hand

Approach

- Passive vs. Active
  - Passive: reading the code, understanding it
  - Active: writing code, producing it
- You should definitely spend time reading code and understanding it, but to **actually produce code you must actually write code**
- Note that you will most likely be able to understand before you can produce
  - Its like talking to someone that understands English but can't really answer you back correctly

Approach

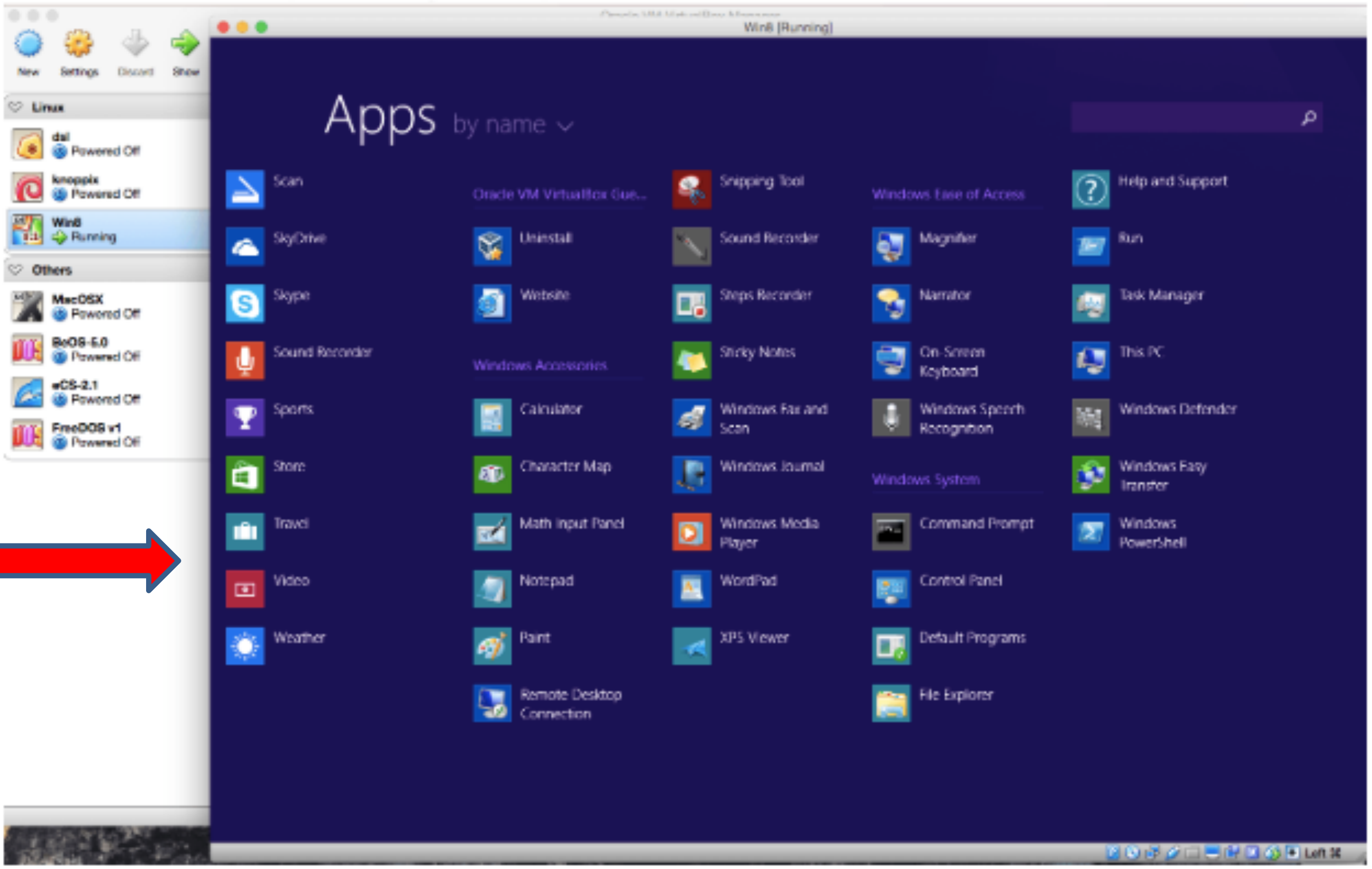
- Conclusion:
  - This is not a history or business class where you focus alone on learning the content and material
    - You will fail if you treat it as such
    - You can't *"cram"* the night before and memorize a bunch of material
      - You can't learn how to speak German in one night can you?
  - It's a:
    - **Language acquisition class** -learning the language
    - **Problem solving class** -how to think and solve a problem before expressing it in the language

VIRTUAL MACHINE

Virtual Machine

- In class, you will sometimes see me using Omega to run programs and sometimes using a virtual machine
  - You can think of this as running a computer within another computer
- We will use something called VirtualBox to allow this to happen
  - We can run multiple operating systems on one computer
  - Normally, our laptops only use one operating system

This is a Mac computer running Windows 8



More info:
https://www.virtualbox.org/manual/ch01.html

Virtual Machine

- All assignments submitted must run on either Omega or the virtual machine (given below)
  - The compiler on Omega is VERY OLD-it is a good idea to go ahead and start using the virtual machine
  - Automatic **60 point** deduction if this is not the case
  - Make sure to specify in your README where to run it
- When we get to GUIs, I **<u>will only be using the virtual machine</u>**
  - It is good to go ahead and get used to it now

Virtual Machine

You will need to download the following two things:

1. https://www.virtualbox.org/wiki/Downloads
   - This is the visualization application that allows us to run a computer within a computer

2. Download one of the following:
   - This is the machine you will be running (takes up a few GBs)

(most recent version)

https://drive.google.com/file/d/1TeaWDVnwbl5F2JDoaQmzfahmxteXtoiP/view

(less recent)

https://drive.google.com/file/d/1AuDlnEGtHNX2ugWtDMjXqtyfgUV9gzmi/view?usp=sharing

Virtual Machine

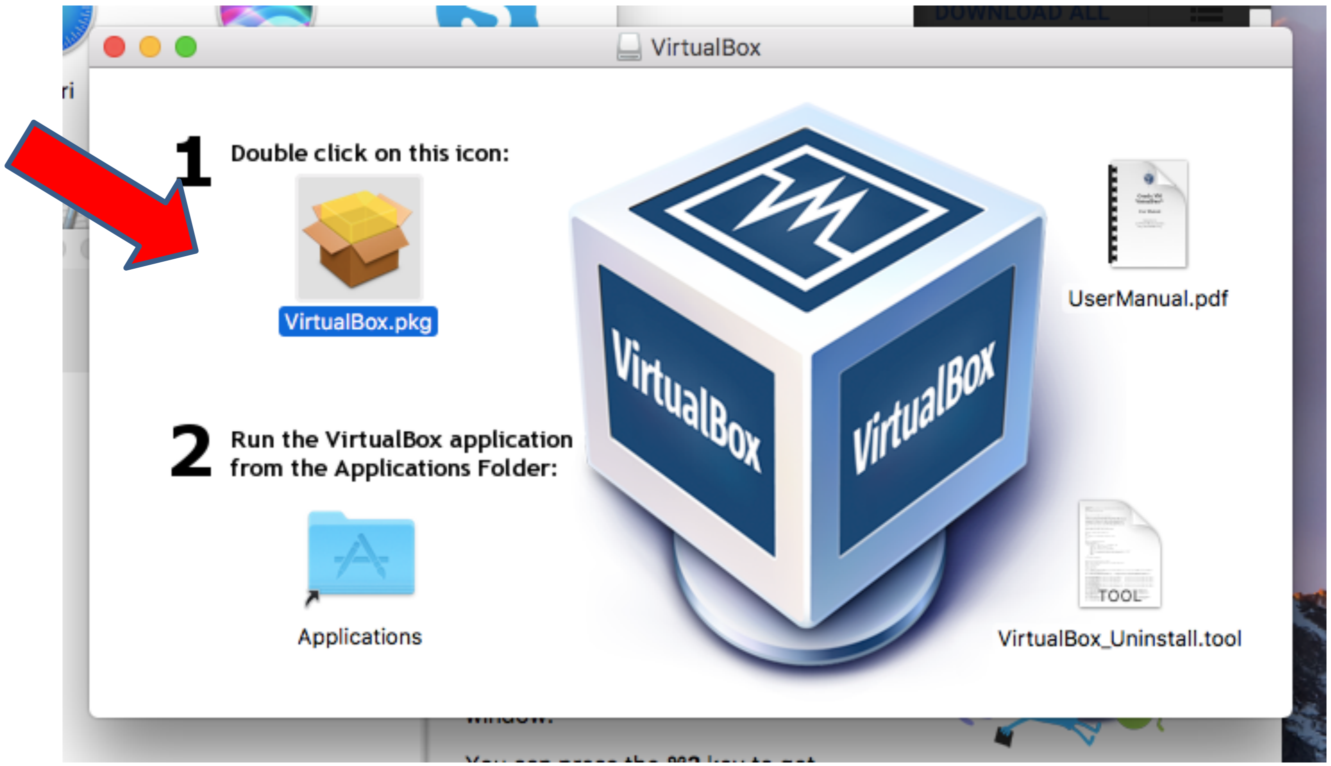
Here is a video from Professor Rice to help you with installation **(VERY HELPFUL!)** :

https://www.youtube.com/watch?v=4fjL8ULadOo&feature=youtu.be

(he also mentions sharing folders between machines)

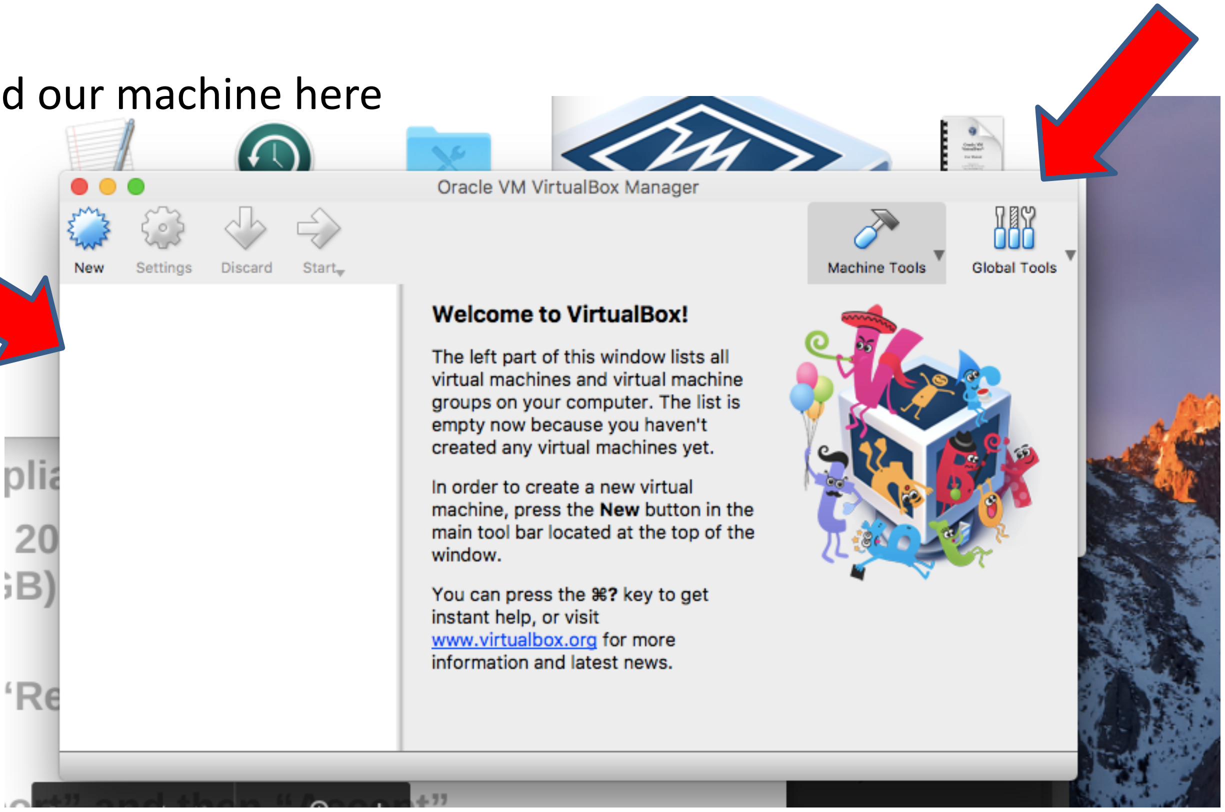
Virtual Machine

Once your have finished downloading both items, you should have a screen like the following (it will look different on a Windows computer):


VirtualBox
1 Double click on this icon:
VirtualBox.pkg
2 Run the VirtualBox application from the Applications Folder:
Applications
UserManual.pdf
VirtualBox_Uninstall.tool

Virtual Machine

We will now add our machine here



Welcome to VirtualBox!

The left part of this window lists all virtual machines and virtual machine groups on your computer. The list is empty now because you haven't created any virtual machines yet.

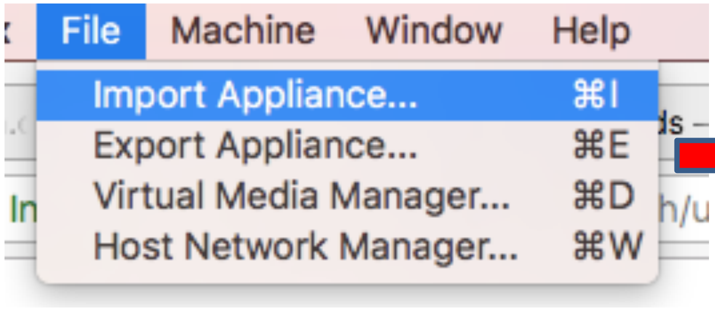
In order to create a new virtual machine, press the **New** button in the main tool bar located at the top of the window.

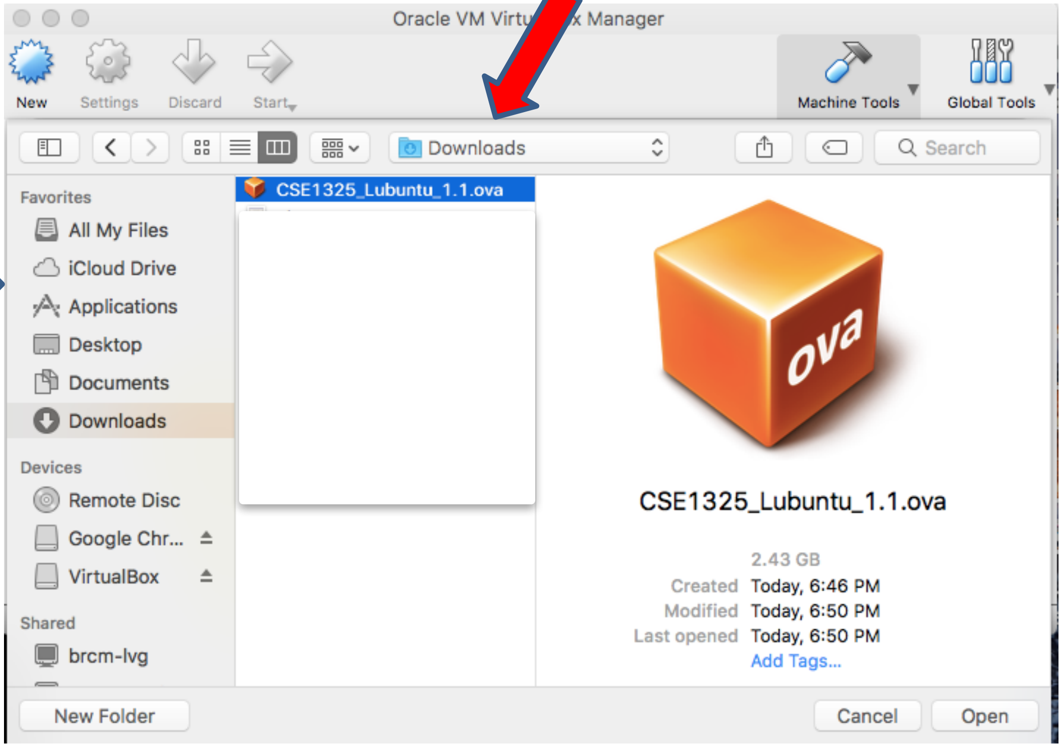
You can press the ⌘? key to get instant help, or visit www.virtualbox.org for more information and latest news.

This is VirtualBox (the application that allows us to run another computer within our computer)

Virtual Machine

This is the second thing you downloaded (the ”other computer”)





CSE1325_Lubuntu_1.1.ova

2.43 GB

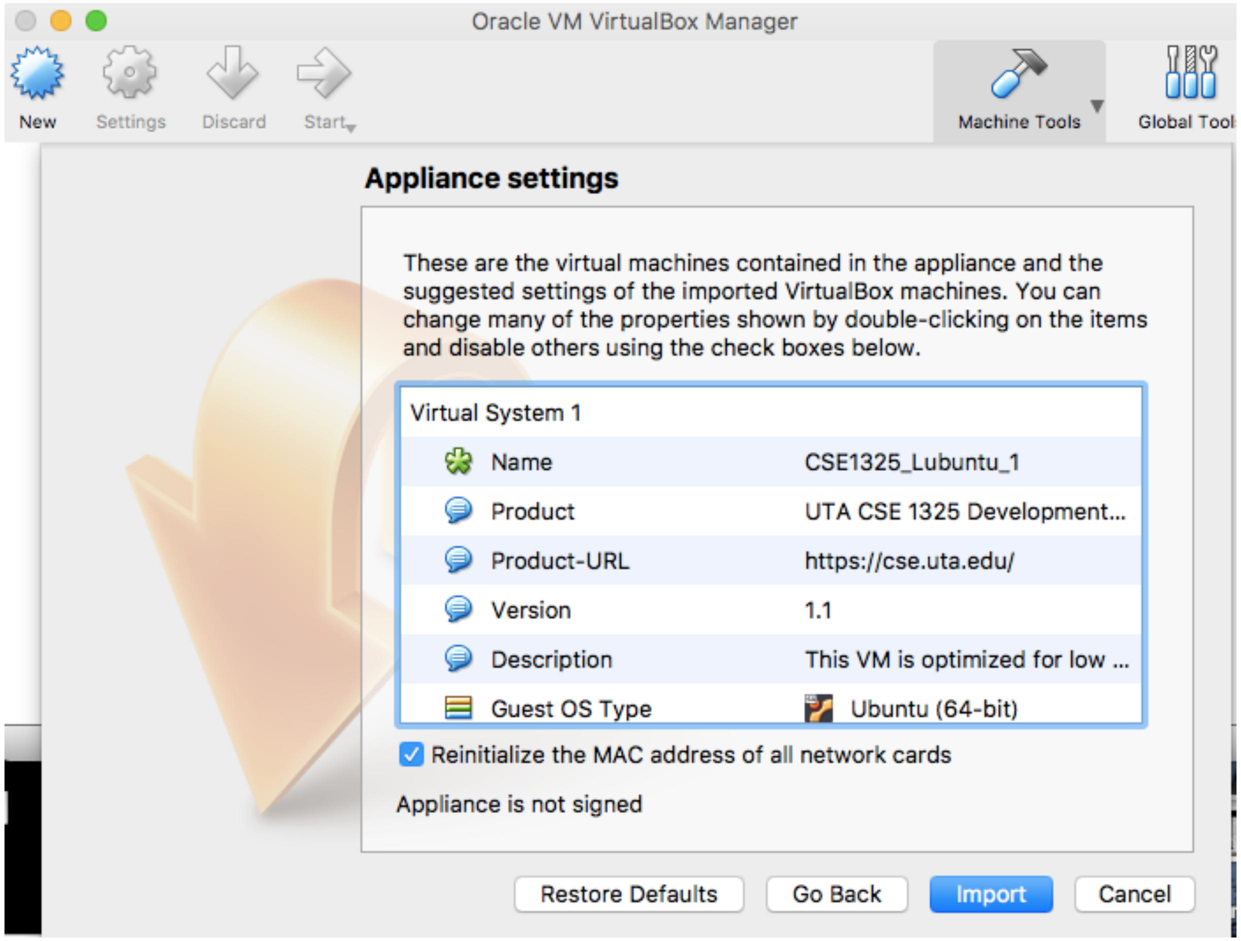
Created Today, 6:46 PM

Modified Today, 6:50 PM

Last opened Today, 6:50 PM

Add Tags...

Virtual Machine



Appliance settings

These are the virtual machines contained in the appliance and the suggested settings of the imported VirtualBox machines. You can change many of the properties shown by double-clicking on the items and disable others using the check boxes below.

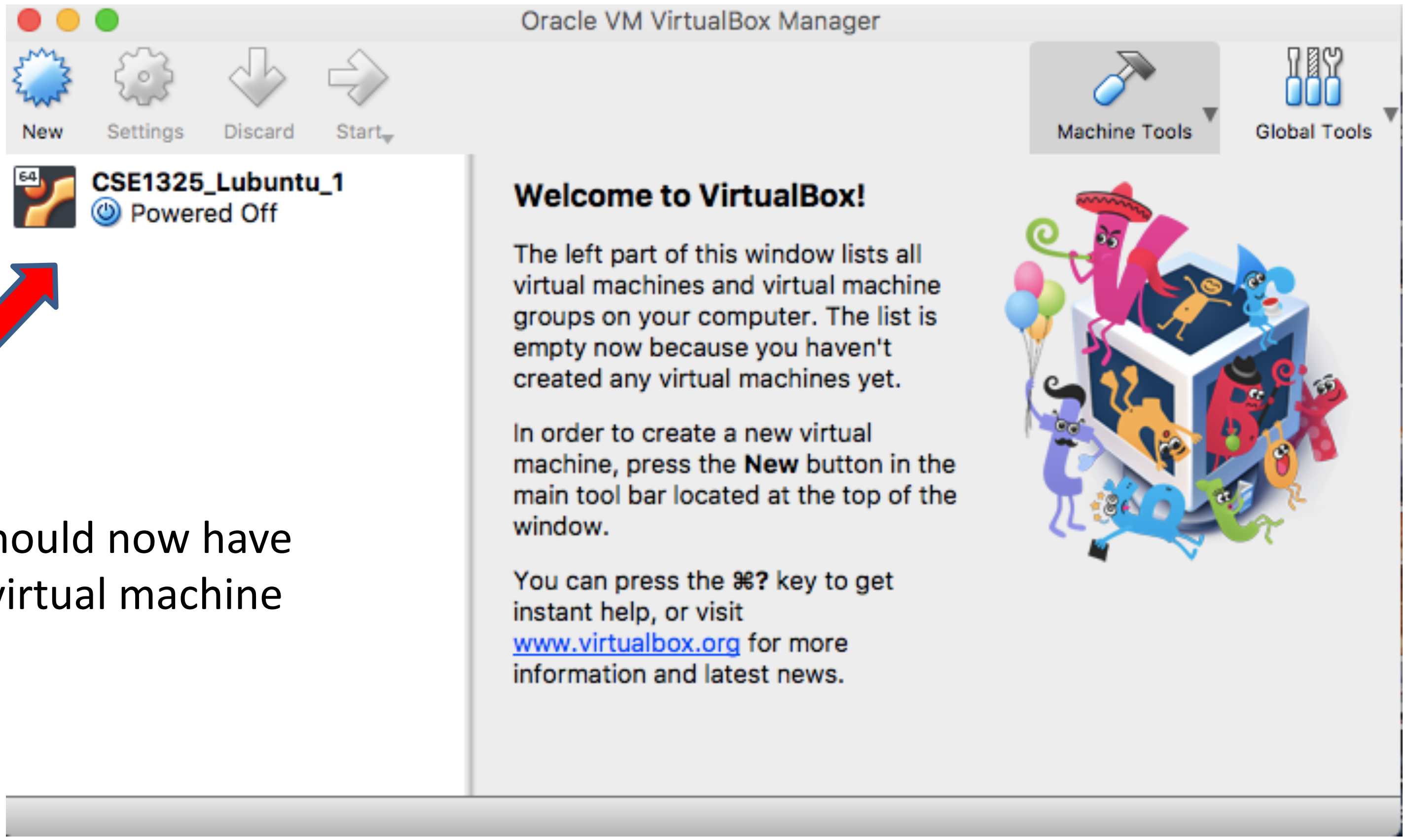
Virtual System 1

Name	CSE1325_Lubuntu_1
Product	UTA CSE 1325 Development...
Product-URL	https://cse.uta.edu/
Version	1.1
Description	This VM is optimized for low ...
Guest OS Type	Ubuntu (64-bit)

✓ Reinitialize the MAC address of all network cards

Appliance is not signed

Virtual Machine



Welcome to VirtualBox!

The left part of this window lists all virtual machines and virtual machine groups on your computer. The list is empty now because you haven't created any virtual machines yet.

In order to create a new virtual machine, press the **New** button in the main tool bar located at the top of the window.

You can press the ⌘? key to get instant help, or visit www.virtualbox.org for more information and latest news.

You should now have your virtual machine

Virtual Machine

- I will now show you an example of the Virtual machine on my computer

C++ STANDARDS

C++ Standards

- Much like human languages, programming languages evolve and change over time
  - Different features are added to the language
- We use compilers to convert our programs written in C++ to a language the computer understands
- These two points mean that our compilers will have different versions to handle different evolutions of the language
  - Omega has an older version than the virtual machine
  - https://gcc.gnu.org/projects/cxx-status.html

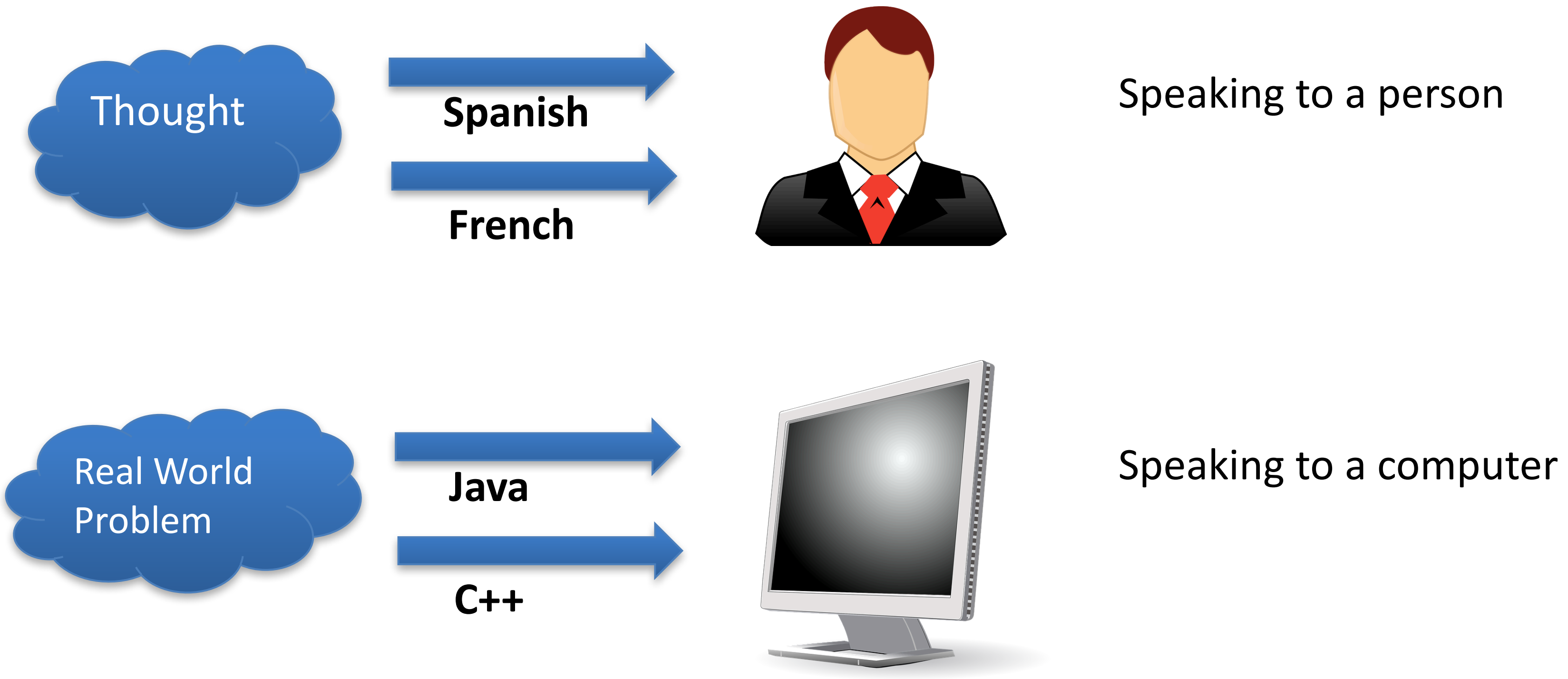
INTRODUCTION

Introduction

- Welcome to your third programming class!
  - *Note*-You will still learn other languages and concepts in future classes
- It is assumed at this point you are comfortable with the idea of programming
  - Remember, all your classes up until now have been about introducing you to programming principles and problem solving
    - You have been implementing these principles in the programming language specified by the course
    - In theory, we could have used any programming language for any of the courses

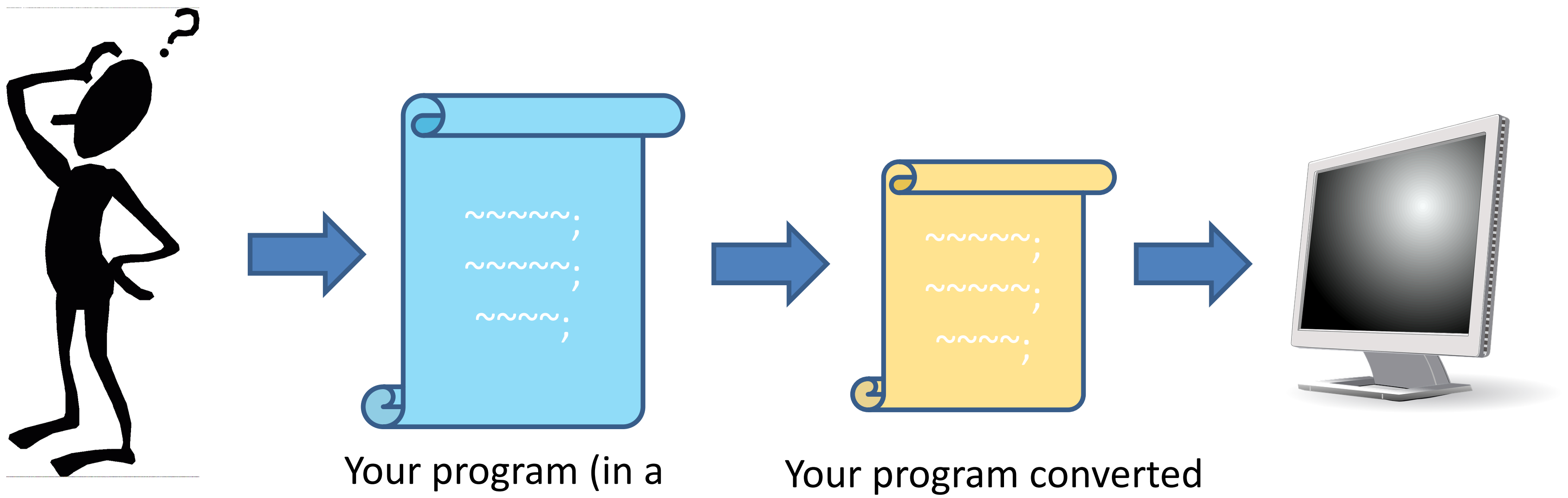
Introduction

- Remember that a programming language is just a way for you to "speak" to the computer



Introduction

- Also remember that your program is just a list of instructions telling the computer what to do (step by step)

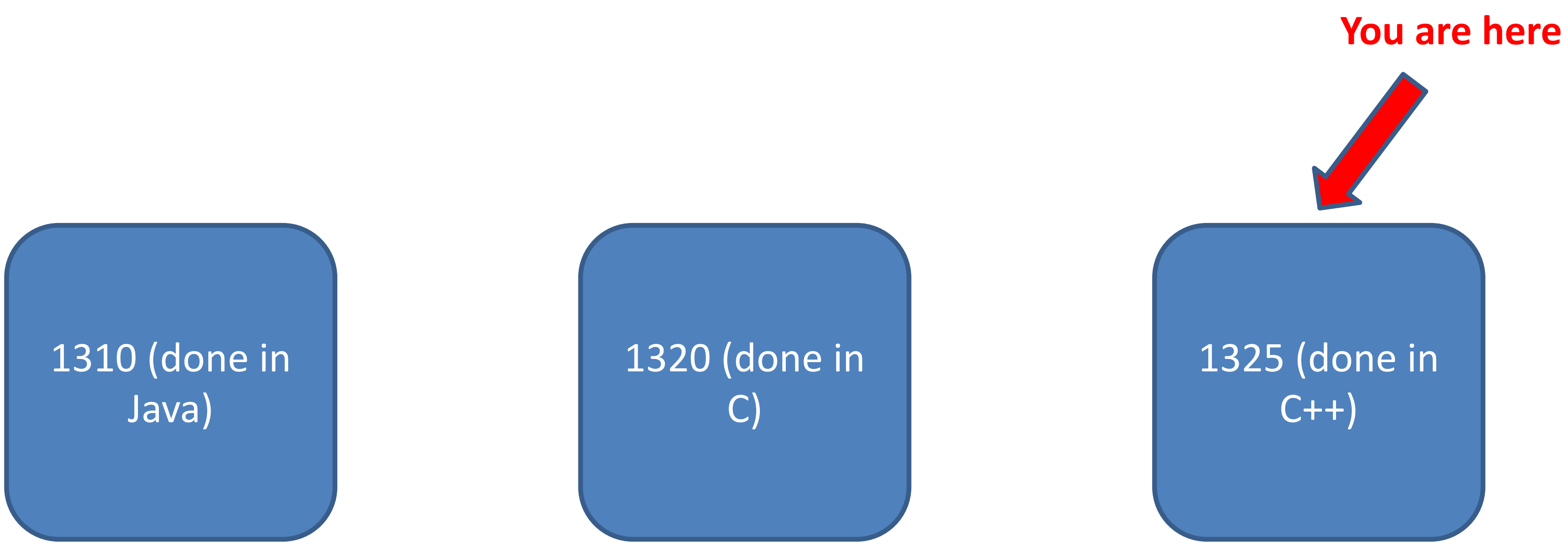


Your program (in a language like Java or C++)

Your program converted into the language the computer understands

Introduction

- Finally, remember that:
  1. Most of your time as a computer scientist will be you solving real world problems and organizing your thoughts pertaining to these problems
  2. You need to be ”fluent” enough in your programming language of choice to communicate the steps you need your computer to take
     - Your computer is a tool for you to use-you just need to tell it what to do!



Intro to programming

This class was about introducing the idea of programming and basic programming principles.

Intermediate programming

Concepts learned in 1310 were a foundation for this class. This class went much more in depth with programming principles.

Object-Oriented programming

This class will continue with more advanced programming concepts. You will be introduced to the object-oriented programming paradigm.

SAMPLE PROGRAM

Sample Program

- See class code
- NOTE: in class I am usually using an editor called Atom