Administrative Issues

EECS1022 Sections M & N:
Programming for Mobile Computing
Winter 2021

CHEN-WEI WANG

Instructor

- How may you call me?

  “Jackie” (most preferred),

  “Professor Jackie”, “Professor”, “Professor Wang”, “Sir”, “Hey”, “Hi”, “Hello”
- When you need advice on the course, speak to me!
- Throughout the semester, feel free to suggest ways to helping your learning.

If You Are Not Enrolled Yet

- Send me an email ASAP requesting access to the course eClass site, with your *name*, *student number*, *York Passport ID*.
- Still keep up with lectures and tutorials.
- Still complete labs and tests (***no extension***).

Writing E-Mails to Your Instructor

- Think of me as your **colleague** who is happy to help you learn.
  - ***formality*** is unnecessary
  - ***courtesy*** is expected
- This sounds ***very rude*** (and may be delayed, if not ignored):

```
On the link you sent us for our mark
my mark for lab0 did not appear on it
and i submitted lab0 during my lab session
```

- This sounds ***much nicer***:

```
Hello Jackie, the link you sent didn’t work.
I did submit my lab0. Could you please look into this?
Thanks! Jim
```

Course Information

- Two eClass sites
  - *LE/EECS1022 M, N, O -- Programming for Mobile Computing (Winter 2020--2021)*
    - Syllabus
    - Common announcements for all Sections M, N, O
    - Course forum
    - Lab instructions
    - Programming Tests
    - Exam
  - *LE/EECS1022 M & N -- Programming for Mobile Computing (Winter 2020--2021)*
    - Announcements for Sections M & N only
    - Written Tests
- Check your emails regularly!

Required Study Materials

- Lecture materials (recordings, iPad notes, slides, example codes) will be posted on my website for you to *re-iterate concepts and examples*:

  `https://www.eecs.yorku.ca/~jackie/teaching/lectures/index.html#EECS1022_W21`

- The *course syllabus* is also posted in the above lectures site.

Course Syllabus

Let’s go over the *course syllabus*.

Need Accommodation?

- Please contact me via email as soon as possible, so we can make proper arrangements for you.
- We will work out a way for you to gain the most out of this course!

Why this Course? (1)

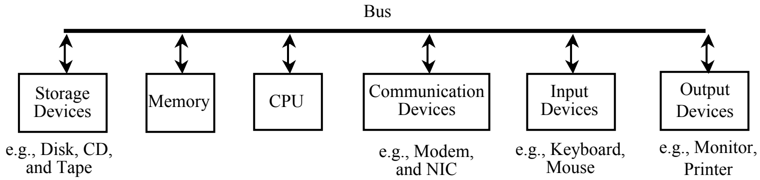
- It is a *pre-requisite* to:
  - ***EECS2030***: Advanced Object Oriented Programming
  - ***EECS2011***: Fundamentals of Data Structure

[the “job interview course”]

Why this Course? (2)

- *Computational thinking (CT)* is a fundamental skill for **everyone**, not just for computer scientists.
  - Reference: Wing, J.M., 2006. *Computational thinking*. Communications of the ACM, 49(3), pp.33 – 35.
  - Thinking like a computer scientist means **more than being able to program** a computer. It requires **thinking at multiple levels of abstraction**.
    - ***Level of Java Code***: How Programs Behave at Runtime
    - ***Above the Level of Code***: *Logical rationale* behind some *functioning*/*malfunctioning* code.
- Being able to think *abstractly* without seeing changes on a physical device is an important skill you are expected to acquire when graduating.
  - Think of programming interviews at Google: Given problems described in English, solve it on a whiteboard.

What Is Course About? (1)



A computer includes both:

- *Hardware*
  - visible, physical, tangible (peripheral) devices
  - *repeatedly* and efficiently executes given instructions
- *Software*
  - invisible, abstract, intangible task-control instructions
  - reflects programmers' *intelligence*

Does the notion of ***stupid computer*** really make sense?

What Is Course About? (2)

- What computers read is difficult for humans, and vice versa.
  - Computers are good at processing *machine language* (0s and 1s).
  - Human beings are good at *abstract thinking* for problem solving.
- *Assembly language* is a big step forward for humans to specify steps of primitive instructions (e.g., memory loads/stores, arithmetic operations, etc.).

  Say `$t0`, `$t1`, `$t2`, `$n`, `$i` are addresses; `$n` stores value *N*:

```
lw      $t0, $n           # fetch N, store in $t0
mult    $t0, $t0, $t0     # store N*N in $t0
lw      $t1, $n           # fetch N, store in $t1
mult    $t1, $t1, 3       # store 3*N in $t1
add     $t2, $t0, $t1     # store N*N + 3*N in $t2
sw      $t2, $i           # store N*N + 3*N in $i
```

  - *Level of abstraction* of the assembly is still *too low* for humans.
  - The above is equivalent to a line of Java code: `i = N*N + 3*N`
  - You will have fun with programming in assembly in EECS2021!

What Is Course About? (3)

- *High-level programming language* (e.g., Java) is even closer to our natural way of thinking (i.e., closer to “writing an essay”).

```
Scanner keyboard = new Scanner(System.in);
int weight = keyboard.nextInt();
int height = keyboard.nextInt();
int bmi = weight / (height * height);
System.out.println("BMI (Body Mass Index) is: " + bmi);
```

- You will study fundamentals for *Computational Thinking*:
  - assignments
  - conditionals
  - loops
  - 1D and 2D arrays
  - classes and objects
  - attributes and methods

Is This an Easy Course?

This may ***not*** be an easy course.

- You need to work HARD and STEADILY in order to perform well.
- Hardware experiment (e.g., Android Tablet, Phidget board) is only meant to be a way to have you engaged.
- Acquiring the *programming* and *problem-solving* skills is the key to success in this course.

But this will ***be*** a course for you to acquire solid computational thinking and programming skills.

Study Tips

- Plan steady, gradual study of:
  - Lecture videos [≈ 2 hours]
  - Java tutorial videos [≈ 1.5 hours – 2 hours]
- *Ask questions!*
- Take (even incomplete) notes, which will help when re-iterating lectures.

General Tips about Studying in a University

- To do well, ***inspiration*** is more important than ***perspiration***.
- Hard work does not necessarily guarantee success, but no success is possible without ***hard work***

  ⇒

  - Don't be too satisfied just by the fact that you work hard.
  - Make sure you work hard both on ***mastering "ground stuffs"*** and, more importantly, on ***staying on top of what's being taught***.
  - Be *adventurous* about going beyond lectures (e.g., CodingBat).
  - Be *curious* about why things work the way they do.
  - Always *reflect* yourself on *how things are connected* .

Index (1)

Index (2)