CSC207 Fall 2017
Software Design

Larry Zhang

The Teaching Team

- LEC0101: Arnold Rosenbloom (coordinator)
- LEC0102: **Larry Zhang (me)**
- LEC0103: Sadia Sharmin

TAs: Arsala Bangash, Fatima Nadeem, KyoKeun Park, Luke Sawczak, Michael Szpakowicz, Prateem Shrestha, Santiago Ordonez Arias, Swapnil Shah, Veerpal Brar

Today's Outline

- **Why** do we take CSC207?
- **What** do we learn in CSC207?
- **How** to do well in CSC207?
- Start learning stuff...

Why CSC207?

If CS meant Carpentry Skills ...

After CSC148 you can do these:

After CSC207 you'll be able to do ...

Why CSC207

In CSC207, you'll adopt the skills to design and develop (in teams) large-scale software architectures which are complex, yet stable, efficient and easy to maintain.

In short, your code will be **elegant**.

What’s in CSC207?

What’s to be learned in CSC207

- Java: not just another programming language.
- Version control, build management, unit testing, scrum: manage your project and collaborate with others like a pro.
- Object-oriented design: how to “put things together” in a complex software product. The stuff you do before you start writing code.
- Design patterns: “Put things together” in the most right and clever way.
- Regular expression: only those who have learned it shall use it.
- Floating point: the numbers in the computer are not what you think they are.

How to do well in CSC207

First of all...

Be Interested.

Logistics

- Course website: http://axiom.utm.utoronto.ca/~207/17f/index.shtml
- Lecture Git repos:
  - **(Larry) git clone ssh://UTORID@cslinux.utm.utoronto.ca/student/cslec/207/207_L0102.git**
  - (Arnold) git clone ssh://UTORID@cslinux.utm.utoronto.ca/student/cslec/207/207_L0101.git
  - (Sadia) git clone ssh://UTORID@cslinux.utm.utoronto.ca/student/cslec/207/207_L0103.git
  - Replace "**UTORID**" with your UTORID
- Course announcements are via emails (from Arnold most of the time). Make sure to read them carefully.
  - His first email sent on Sep 4 has important info on how get started!

Important Tip

You don’t have READ-ONLY access to the lecture repos, so you cannot push to the remote source.

You probably don’t want to commit changes locally either, because next you pull from the repo, it may cause a conflict.

For your assignment repos, you have READ and WRITE access so you can do whatever you want (keep it organized though).

Logistics

- TUT sections are for the labs (held on Mondays)
  - Labs are worth marks!
- PRA is only relevant for the week of midterm test.
- Office hours:
  - Larry: Tuesday 2:30 - 3:30 PM
  - Sadia: Wednesday 2 PM - 3 PM
  - Arnold: TBA
- Discussion board (MyBB): https://mcs.utm.utoronto.ca/~mybb/utmcs/forumdisplay.php?fid=298

Assignments and Tests

- Three programming assignments
- Midterm Test
- Final Exam

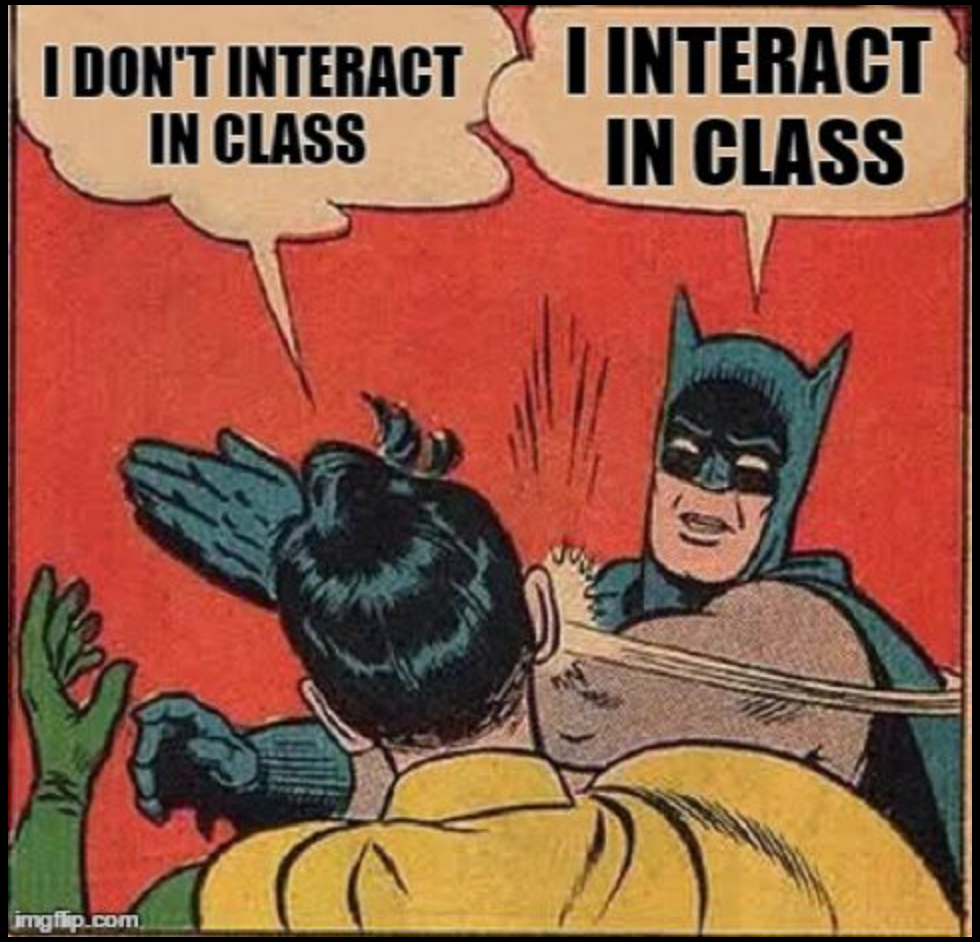
Read the syllabus for detailed marking schemes, as well as other policies.

Tips: How to do well in CSC207

- Read announcements carefully.
- Read the readings.
- Know how to find useful info in documentations, know how to Google.
- Attend lectures, be interactive, ask questions.
- The lectures can’t teach everything. You learn more from docs and practices
- Go to labs, practice is the key!
- Office hours are very useful.
- Work hard on assignments, collaborate intelligently, manage your time.
- Leeching off others is almost the only way to fail the course for sure!
- Give us feedbacks

Anonymous Feedback Form for Larry

https://goo.gl/forms/kGACdkCcnCopRi2p2


I DON'T INTERACT IN CLASS
I INTERACT IN CLASS
imgflip.com


GODIVA

Kahoot!

Sometimes, we do some in class pop quiz challenges in class.

To participate, you'll need:

- be in the class
- have access to a browser (on a phone or laptop)
  - will visit: https://kahoot.it/
- or, you can install the Kahoot app for iOS or Android
- **small prizes** will be offered to each game's 1st-place winner
  - has nothing to do with your course grade

Intro to Java

Overview

- Version 1.0 released in 1995 by Sun Microsystems (now owned by Oracle)
- Designer: James Gosling

Write Once, Run Anywhere (WORA)

- All computers first install a Java Virtual Machine (JVM)
- Java programs run within the JVM and never need to worry about hosting computer's hardware.

The execution of a program

- The **compiler**-based approach (C/C++)
  - Before running, the **human code** is translated into **machine code** instructions which can be understood by the hosting computer's hardware. (Hardware dependent)
  - When running, the computer hardware executes the machine code instructions.
- The **interpreter**-based approach (Python)
  - Nothing done before running.
  - When running, the human code is translated and executed on the go, one statement after another.
- The **hybrid** approach (Java)
  - Before running, the human code is translated into **bytecode**, a.k.a. the machine code for the Java Virtual Machine. (Hardware independent)
  - When running, JVM executes the bytecode instructions.

Other things about Java

- It’s still popular: https://www.tiobe.com/tiobe-index/
- It’s faster than Python: http://benchmarksgame.alioth.debian.org/u64q/python.html
- Android apps are written in Java (Kotlin is only the rise though).

Learning Java is gonna be quick

- You already know Python; that means you know most of the stuff already.
- Just need to learn about the differences and you'll be ready to go.
- 8 months (CSC108 + CSC148) vs 8 days

Difference #1: Static Typing vs Dynamic Typing

- In Python, you can change the type of a variable dynamically.
- In Java, every variable must be first declared with a particular type, and you're not suppose to change the type afterwards.

```
# Python
x = 3
# x is an integer 3

x = “Hello World!”
# x is now a string
```

```
// Java
int x = 3;
// x is declared as an integer

x = “I’m a string”;
// ERROR! NOT OK!
```

What are the pros and cons of each?

Difference #2: Braces vs Indentation

- In Python, scopes are indicated using indentation.
- In Java, we use braces {}.

Java

```
if ( a > b )
{
    a = b;
    b = c;
}
```

Python

```
if  a > b :
    a = b
    b = c
```

Difference #3: Java is wordier

HelloWorld.py:

```
print("Hello, world!")
```

Hello World.java

```
public class HelloWorld
{
    public static void main (String[] args)
    {
        System.out.println("Hellold!");
    }
}
```

We’re ready to start coding.

Demos

- HelloWorld.java in the “207prep” repo
- DataTypes.java
- ControlsFunctions.java (will do in Week 2)
- OO Programming: Balloon.java vs Balloon.py (Will do in Week 2)

Git Quick Starter

```
# download the repo
git clone ssh://UTORID@cslinux.utm.utoronto.ca/student/cslec/207/207_L0102.git


git pull                 # pull the latest change
git log                  # check the history of commits
git checkout COMMIT_ID   # revert to an older commit
git checkout master      # go back to the latest version

git status               # check status

git add filename         # add the change you made to a file
git commit -m “message”  # commit the change to your local repo
git push                 # push to the remote source
```

Readings

- Java Language Basics
  - http://docs.oracle.com/javase/tutorial/java/nutsandbolts/index.html
- Git
  - https://www.atlassian.com/git/tutorials/
  - https://git-scm.com/docs/gittutorial

More References

- The general tutorial:
  - http://docs.oracle.com/javase/tutorial/
- The complete API documentation (has all info you need)
  - http://docs.oracle.com/javase/8/docs/api/
- More code examples by Arnold
  - http://www.cs.toronto.edu/~arnold/cit/summer2007/code/NonOOFeatures/index.html

If I can't login to cslinux (or git clone), email andrew.wang@utoronto.ca