# CS 211
# Object Oriented Programming

# Introduction

# Intro, Housekeeping

Syllabus
https://cs.gmu.edu/media/syllabi/Spring2018/CS_211SnyderM.html

Schedule
https://cs.gmu.edu/~marks/211/schedule.html

All documents
https://cs.gmu.edu/~marks/211

CS211 Textbook (the old "Lab Manual")
https://cs.gmu.edu/~marks/211/textbook/

Free Practice Problems

- http://practiceit.cs.washington.edu/
- http://www.codingbat.com

# Sign up/login for things:

Blackboard – mymason.gmu.edu

- submit work, view grades

Piazza – piazza.com

- announcements, public/private correspondence

Pytania - pytania.cs.gmu.edu

- class participation questions (free; join your section!)

**Zyante - https://learn.zybooks.com/**

- Join code: **GMUCS211SnyderSpring2018**

# Keys to Success

**Most important: practice, practice, practice!**

don't allow yourself to get behind the curve.

**Read**

the reference material prior to lecture

**Ask questions**

→ Silence denotes implicit understanding.

**Utilize available resources**

→ Discussion forums, office hours, lab time, etc.

# Discussion Forums on Piazza

## Public posts

- general questions, clarifications
- NO PROJECT CODE!

## Private posts

- specific questions about your situation, concrete questions about your code.
- Only discuss our class on piazza. Be respectful to fellow students and TAs.
- These posts get no attention:

  *"here's my code; plz tell me what's wrong."*

# Thoughts

- Only administrative issues should go through email (GMU accounts only)
- Be responsive in class—nod for yes, speak up with questions, and so on
- There are many students here, but interact with me and I'll try to learn your name ☺
- Each credit-hour in class should match two or three hours outside of class. (12 credits = 36 hour weeks, 15=>45, …)
- Don’t distract others (loud food, web browsing, etc)

# How to get an A

- Read all assigned reading before class.
- Attend all lectures and lab sections.
- Try assignments very early, in time to ask questions.  Only turn in 100% working code.
- Go to office hours regularly with your questions! Also use the forums on piazza.
- Study early, often, and well for tests.

# How to get a B

- Do pretty much all the reading.
- Only miss one or two lectures/labs, and catch up with someone on what you missed.
- Start assignments earlier than the night before, and get programs working.
- Occasionally use office hours, forum as needed.
- Study hard just before tests.

# How to get a C

- Only miss one or two labs and lectures.  Goof off or get distracted occasionally in them, though.
  → *this includes doing project coding in class!*

- Do some of the reading, but not before class.

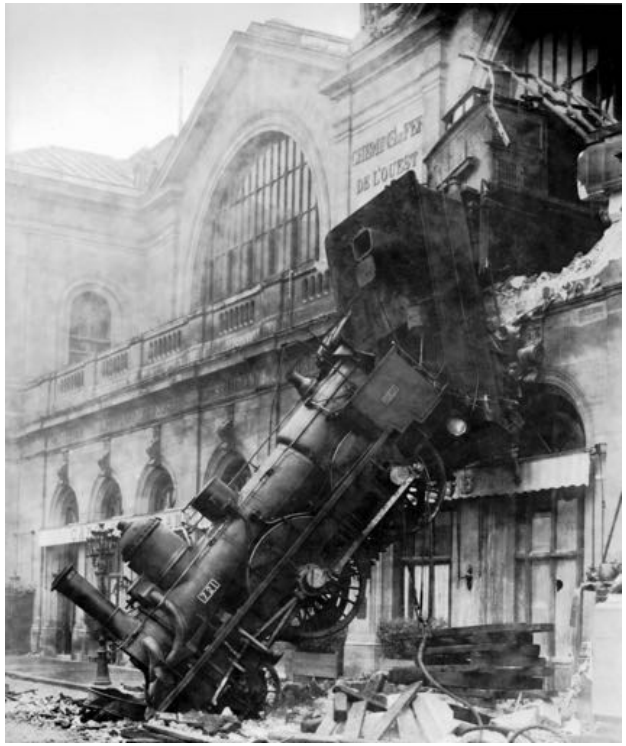
- Try hard on the assignments, but usually with self-imposed deadlines, just the night before or in one ‘power session’.  No time to ask for help or clarification.

- “I don’t have time for office hours”.

# How to get a D

- Miss a lot of lecture/lab sessions.
- Skip most of the reading to save time, except when the work is too confusing.
- Start assignments at the last moment.
- Realize about 2/3 into the semester that you're failing, and genuinely try really hard to catch up.
- Never come to office hours (except maybe when it's already too late!).

# How to Fail



- Decide the first couple weeks were easy, and stop coming to lecture.

- Miss many lab sections.
- Skip all the reading because it's not graded.

- Start assignments the night before they're due, and not be able to ask questions.
- Cram for tests the night before, if at all.

# Course Assumptions

- You have credit for CS 112: you know how to program procedurally.
- You do not know Java yet, but want to learn it well.

- You need a solid programming foundation for later courses, or for your major, or for other personal reasons.

- You are an adult who understands that great grades are earned through hard, consistent work. You will earn whatever grade you get, be it an F or an A.

- You have interests beyond this course, classes besides this class, a life outside of CS 211. Time management will decide whether you spend your time well or not.

# Course Overview

- Java Introduction
- Classes and Objects
- Inheritance
- Interfaces
- Exceptions / Handling
- Unit Testing
- Searching, Sorting
- Data Types
- Recursion
- Generics/ Collections

# On to Java!

# Java


A *programming language* that supports object-oriented programming and other styles.

Source files are:

→ written in Unicode (just a text file)

→ compiled to bytecode (a machine-independent repr.)

→ interpreted via the Java Virtual Machine (JVM)

Java was introduced in 1995 and its popularity has grown quickly since.

# Syntax and Semantics

- The *syntax rules* of a language define how we can put together symbols, reserved words, and identifiers to make a valid program
- The *semantics* of a language define what statements mean
- A program that is syntactically correct is not necessarily logically (semantically) correct
- A program will always do what we tell it to do, not necessarily what we meant to tell it to do

# Java Program Structure

In the Java programming language:

- A program is made up of *classes*
- A class contains *methods*
- A method contains program *statements*

A Java program always has a **`main`** method

# Java Program Structure

```
public class HelloWorld {
    public static void main (String[] args) {
        //our instructions go here
        System.out.println("Hello, World!");
    }
}
```

# Errors

A program can have **three types of errors:**

*compile-time errors:* The compiler finds syntax/type errors

- If compile-time errors exist, an executable version of the program can't be created (this isn't a valid Java program)

*run-time errors:* A problem occurs during program execution, such as trying to divide by zero; program terminates abnormally

*logical errors:* A program runs, but gives incorrect behavior

- program's meaning doesn't match our intention

# Basic Program Development