

# CmpE 275

---

## Section: Introductions

Part 0: Course description, tools

# Welcome To 275



John Gash

[john.gash@sjsu.edu](mailto:john.gash@sjsu.edu)

I'm on campus one-ish days a week:

Monday (275– Clark R/310)

Wednesday (275 - Duncan Hall R/351)

(My office hours start after class and conclude when we are done)

### **How to find me**

1. In person – around campus, after class
2. Email (spotty)
3. Department mailbox (graveyard)

# Prerequisites

- CmpE 273
  - ◆ Special cases can be waived in lieu of 273
    - E.g., if you have professional software engineering experience (not internships) – requires instructor consent
- Proficiency in multiple programming languages
  - ◆ Main: C++, **Java**, **Python**
  - ◆ Secondary: Bash, Ruby, Erlang, ...
- Strong organizational and time management skills
  - ◆ Tasking, time tables and schedules, and integration

# Lectures and data available online (instructure)

- **<https://sjsu.instructure.com>**
  - ♦ Lectures, discussions/help, and greensheet
  - ♦ Information: Assignments, data, examples
- We are mixed lecture and **inverted classroom (flip)** w/ discussions
  - ♦ Labs and discussion balance slides
  - ♦ You are required to research and connect concepts to be successful on the projects
- Communication and Planning
  - ♦ The flipped classroom promotes communication and interaction
  - ♦ Problem solving through teams – form teams quickly!

# Recommended Reading Summarization

(no textbook is required for this course)

**The book spectrum will span many subjects, most applicable:**

The Ruby Programming Language, by Flanagan, Matsumoto, 2008

**Python Essential Reference, 4th Ed., by David Beazley, 2009**

Programming Python, by Mark Lutz, 2011

## **Supporting:**

Software Systems Architecture, by Rozanski, Woods, 2005

Software Architecture in Practice, Second Edition, by Bass, Clements, Kazman, 2003

UML Distilled, Third Edition, by Fowler, 2003

Applying UML and Patterns, 3<sup>rd</sup> Edition, by Larman, 2004

Primary reading:

1.online tutorials

2.online papers

3.online ...

*Additional references included at the end of each lecture*

# Languages and packages

*If you are not familiar with **Java, C++, Python**, it is your responsibility to spend additional time to learn.*

- Java
  - Generics, Threading
- C++11
  - Boost, OpenMP, OpenMPI, Cmake, CUDA (optional, NVIDIA GPU)
- Python
  - Alternatives okay – Erlang, Scala, Lua, ...
- Scripting
  - Bash

**Projects will challenge you –critical thinking and test *your ideas***

- Distributed Concepts
  - ◆ Data and communication
    - Data affinity and distributed processing
    - Asynchronous vs. synchronous
- Technologies and languages
  - ◆ **Java**, Ruby, **Python**, C++
  - ◆ OpenMP, MPI (optional), CUDA (optional)
  - ◆ Google Protobuf, JBoss Netty, NoSQL
  - ◆ Micro-frameworks: Bottle, Sinatra, ...
- Software engineering
  - ◆ Problem solving, domain modeling
  - ◆ Testing and validation
  - ◆ **Scalability and failover**




# Research, Think, Apply



# On track

- Agenda
  - ♦ Course introduction
  - ♦ Projects, grades, philosophy
- Take away points
  - ♦ What am I getting into?
  - ♦ I need to form/join a team!
  - ♦ Basic tools for class and projects



Key points that you will want to ensure that you understand

Reading and references are found at the end of all presentations (except this one) to provide additional reading and materials.

# Grading (from greensheet)

## Distribution

(65 pts with approximate distribution, %)

### Projects

- P1 25 pts
- Deep dive 10 pts

Lightning 10 pts

Mid-term exam 20 pts

# Project 1

Projects are an opportunity to extend discussions and exploration/research ideas through practical experience. They are an integral part of the learning experience.

As a team (4 or 5 persons) ← change from greensheet

Project design, implementation, presentations (project 2 only)  
discussions

Each team member

- Contributes to all aspects of the project – design, coding, testing, ...
- You are reviewed by your team members – it's part of your grade

Typically, a project will include multiple deliverables, a presentation, source code, test data, and/or a 10-15 page research/investigation paper. Deliverables will vary by assignment.

# Project 2 (Deep Dive)

Deep learning opportunity to look at an area for your 295 project, for work/internship, or the joy of learning

As a team (2-3 persons)

Research, state of the art, and implementation

Lightning talk

# Delivering your project

Team Caffeine is submitting their project 1. They have provided the following files and source directory:

- project1-cafeine/**
- **Installation notes**
- **caffeine-project-report.doc (.pdf)**
- **project-1/** *(do not send class or jar files)*

The archive submitted (emailed) is **caffeine-project1.zip** (.jar, .rar) of the directory:

**zip -r project1-cafeine.zip project1-cafeine**

Upload to Instructure.com - **DO NOT email me your assignment**, I will lose it.

# Disclosure – 275 is a demanding course

For the semester, you will likely spend as much time in 275 as your other courses combined (really).

- This is **not** a learn-by-osmosis class
- High expectations
  - ♦ New technologies
  - ♦ Projects
  - ♦ Design and engineering concepts
  - ♦ Research investigations
  - ♦ Quality of work (design, code, reports)

All this takes time, time, and more time

# Extracting the most from class time (interactive labs)

- Lectures (**Flipped class**)
  - ♦ Attend as you may/can
  - ♦ Why should you come to class?
    - We are not Presentation heavy in 275. Class time is hands on – working on problems that complement the lectures and/or supports the projects.
    - Help from peers and me
  - ♦ Lectures may run late
    - This is especially true as project dates approach
    - If we run late, I will make it up to you by having shortened (or optional) lectures

# Communication and interaction in class and working on projects

- Communication and sharing of ideas (not code) is important to your success in 275.
  - ♦ Networking with fellow class members
  - ♦ Communication and cooperation within teams
  - ♦ In class projects and discussions



# What do you need (and I need) to know about your project

- **There is no single, correct solution to a project**
  - ♦ Descriptions can be vague => your view points count => many solutions
  - ♦ Support your design/decisions – provide reasons/justifications
- Projects are a team effort (no slacking)
  - ♦ Each team is required to submit a summary of what each person worked on
  - ♦ Grades may be reduced for non-participation (do not let your teammates down)
- Deliverables
  - ♦ Code, test cases, test data, installation instructions
  - ♦ Documentation (Design, approach, experiences, limitations, follow up areas, contributions)
  - ♦ Email me your project and documentation
    - in one zip/tar/rar
    - email should include (cc) all team members
  - ♦ Due date: \*-ish
    - Try to be within 3–4 days of the due date (project 2 exclusions apply) – definitely before the next lecture
    - As soon as we discuss the project solutions (next lecture), no projects will be accepted

# Manage your time carefully when working on the projects

- Projects take **60–90 hours (or more)** to complete
  - ♦ **Really, I'm not joking.**
  - ♦ Do not wait until the last week
    - Let's do the math:

*If we estimate a project will take 60 hours then*

- *That's 2.5 days (24 hours a day)*
- *Or 5 days at 12 hours a day*
- *Or 15 days at 4hr/day*

In other words, if you work 15 hours a week, it will take you 4 weeks to complete



# Project 1



- Project 1 will be available online next lecture
- Preparing
  - ◆ Form teams by the third lecture (we don't meet next week)
    - use Instructure to advertise your need or openings in your teams
  - ◆ Install technologies (next slide)

# Technologies for project 1 and related labs

(The list may change)

- Install, configure, and know these technologies
  - ♦ Java, Python, C++ (gcc v4.9 is recommended), Python
    - for Mac OS X install xcode with command line option
  - ♦ JBoss Netty
  - ♦ Google Protobuf
  - ♦ RabbitMQ
  - ♦ JUnit, slf4j, Ant, Virtualenv
  - ♦ Cmake
  - ♦ Boost (C++)
  - ♦ OpenMP
  - ♦ Bash (or your favorite shell)
  - ♦ Editor/IDE (Eclipse, Netbeans, ?)
    - w/ support for Java, C++, and Python

**Optional: Vagrant (a VM)**  
Linux distros: CentOS,  
Ubuntu, ...

# Additional reading

- Future lectures will include additional sources of information

You do not have to buy a book. This may give you a sense of euphoria about this course.

You are wrong. It should scare the \$#%P out of you!  
You are responsible for finding the information!

# Aim for success

- Manage schedules
  - ◆ Set integration dates where the team gathers to integrate code
    - ◆ Assess progress and assign actions/tasks
    - ◆ If you are having trouble meeting internal deadlines, you will have problems meeting project deadlines
    - ◆ Teaming conflicts should be addressed early rather than at the due date or end of the semester
- ◆ Technically consistent
  - ◆ Test cases provides confidence that your code is working
  - ◆ Everyone in a team should be using the same tools (e.g., everyone uses Eclipse)
  - ◆ **Use online (free) version control sites to coordinate**

# Originality and citations

- Plagiarism is never worth the risk.
  - ♦ I check!
- Know how to use citations.

**If you are stressed or out of time? Talk with me before the due date!**

# How to (or not) have a grand time during 275

## Not so good

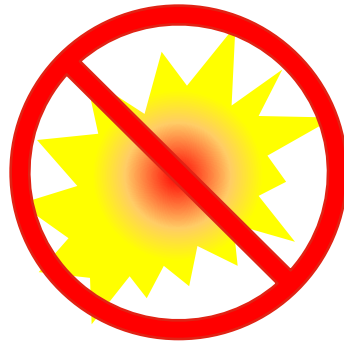
- Isolation yourself from teammates, peers
- Not communicating
- Disorganized, not focused
- Procrastination, sleeping on the job
- Expecting me to design your project

## Good

- Work with your peers (outside of team)
- Regular team discussions
- **Organized** teamwork: track tasks, design
- Creative thinking – open minded
- Work hard, eat well, sleep after Dec 15



“Learn. Experience. Question.” ...



Don't go splat

# References

- Image sources
  - ♦ Time management image
    - <http://learningcommons.ubc.ca/student-toolkits-2/managing-your-time/files/2013/06/LWLW-2-COLOR2.jpg>

# Backup Slides

# The question I will ask you and you should ask me

## Why?

---

- Why did your team choose this design and architecture?
- Why is this [technology, concept, ...] important?
- Why should or can't...?
- Why?

# Stay on track – what's in your scope?

- Organization: Schedules and tasks
  - ♦ Create tasks
  - ♦ Assign dates
  - ♦ Meet your team deliverables
- Lifecycle
  - ♦ Project investigation
    - Assignments are real-world => not fully specified, vague
    - Requires you to think, ask questions!
    - Will I help? Yes. Will I tell you how to code it? No.
  - ♦ Understanding concepts and technologies
  - ♦ Implementation
  - ♦ Testing, validation, stable, and consistent
  - ♦ Documentation
    - How to test, How to install
    - Design, Observations, Conclusions, Recommendations, Evaluations, References

# Balance and Values

Work/Life Balance. Life is not predictable and occasionally we need to rebalance family, class, and work obligations. There are times when no amount of planning allows one to satisfy all requirements and conflicts arise; if you find yourself in such a situation, please talk with me to see if there are options or adjustments that will allow you to be successful.

Philosophy. As mentioned previously, this class is built upon interactive research and discussion. This format requires individuals to perform investigations and research prior to each discussion topic. Sessions or lectures are for the discussion and examination of the topic at hand. Everyone is required to fully participate in all discussions.