1. See final project spec on website
2. Go to slido.com #finalproject
Reminder: Guitar Hero Class Meeting
This Wed 12:30pm @ McCosh 50
Motivation

1. To code is to create
2. CS + X
Final Project

- Develop your own **Java program**
- Open-ended spec to allow for **creative project choices!**
- Individual or **partner pair** (partnering is recommended)
- Small **extra credit opportunities** for going above and beyond the course
- **To get full marks**, you do not need to go beyond the course!
Requirements

1. Scope
2. Technical
3. Style
Scope Requirements

- Project size: roughly **2 assignments** worth of work

- Features: Your project should implement **3 features**

  A feature specifies substantive functionality that your project supports

  (more examples later in slides)
Technical Requirements

- User input
  command line args, standard input, etc.

- User output (based on user input)
  standard output, StdDraw, StdAudio, etc.

- Non-trivial data structure
  arrays, 2D arrays, abstract data types (e.g., stacks + queues)

- Custom functions: at least 2 custom functions

- Tests: at least two sets of two tests (one set of tests for each custom function)
Style Requirements

All style expectations for programming assignments...

Plus, a multi-line comment at the top of each .java file.

The comment should describe what the file does.

```java
public class LayerPlayer {

/*
 * Simple client that plays a series of green screen frames overlayed on a
 * background. The player assumes the background (bg) image and foreground
 * ImageLayers have the same dimensions.
 *
 * Usage: java-introcs <background-image> <green-screen-images>
 *
 * Example: java-introcs LayerPlayer 960x540-bg-desert.jpg 960x540-gs-news/*
 * Example: java-introcs LayerPlayer 960x540-bg-Nassau_Hall.jpg 960x540-gs-news/*
 *
 */
```
Small Extra Credit Opportunities

- Using libraries not covered in COS126
- Learning new computer science material
- Developing a substantial, well-written piece of software (2x intended scope)
- Analyzing efficiency of part of your program
- Create an executable .jar file to run your program
Academic Integrity

You may consult external resources when used to debug problems that are

- **small and non-unique** (i.e., not specific to your project)

  **AND**

  - lie **outside the scope** of COS126

All resources (including COS126 materials) must be cited properly.
Small & Non-unique Problems

- Acceptable: how do I use function X from external library Y
- Not acceptable: a tutorial on building project Z
Outside-of-COS126 Problems

- Using an external library
- Implementing a graphical user interface
- Reading data non-.txt input files
- Researching advanced computer science algorithms

For **COS126** problems, use COS126 resources!
Snippets of Code

- You may use and adapt **small snippets of code** from other resources, provided they are cited properly and used to solve small, non-unique, and outside-of-COS126 problems.

- The essence of your project must be **your code**

- Negative example: a project consisting of copy-and-paste snippets from different sources.

- **Code explanations**: You should be able to explain every part of your code (including adapted snippets).
You may look at and adapt small snippets of code (with citation) from any of the following:

- Library documentation
- Example code snippets on how to use the library
- Example programs that are different from your project
- StackOverflow questions about debugging issues with the library
Citing Resources

1. Maintain a bibliography of every resource looked at and include in README

2. Cite resource near the line of code it informed code and explain how you used that resource
When in doubt...

Ask your project preceptor before proceeding!
Timeline

- Mon 11/1: Class meeting (McCosh 50) - Final project introduction & spec walkthrough (happening now!)
- Mon 11/8: Class meeting (virtual) - Ideation workshop with the Center for Digital Humanities
- Sun 11/14: Project proposals due at 11:59pm
- Mon 11/22 & Tue 11/23: Proposal review meetings (10 min meeting with project preceptor)
- Tue 12/7: Status update due at 11:59pm
- Tue 12/14: Final project due at 5:00pm on Dean’s Date

Submitting on time / showing up to required activities is part of your final grade
Project Proposal (due 11/14)

- Submit on Google forms
  (recommended: prepare in Google docs)
- Sample proposal for Sudoku available
- Late proposals will not be accepted for a grade (feedback will still be provided)
- Optional questions help you plan your final project
- If approved, 3 features are binding (any changes must be approved by project preceptor). Be as specific as possible (and keep things as simple as possible)
10-min Proposal Review Meeting

- Monday & Tuesday, November 22 & 23 before break (in person & virtual options)
- Sign up for appointment when submitting proposal
- Attendance is mandatory for all members
- Preceptor you meet with will be your project preceptor (recommended: choose a preceptor of a group member)
- If your proposal is not approved, changes to make your project approved will be suggested
- Any major project changes after this meeting must be approved by your project preceptor
Project Implementation (due 12/14)

Submit 4 items on TigerFile:
1. Project files
2. Video (2 min max). It must show:
   a. Your program’s input & output
   b. Your program’s 3 features
   It should also include other information about your group
   Recommended: Using Loom or YouTube
3. README
   a. Must include viewable video link
   b. Include clear descriptions of how to compile and run your program
4. Optional: Executable .jar file

+ Fill out Google form for project gallery!
Questions?
Project Ideas: Buckets

We are **NOT** grading based on buckets!

- Standard
- Sprinkle
- Sparkle
1. **Standard**: text-based I/O, graphical or auditory output (StdDraw, StdAudio)

2. **Sprinkle**: simple graphical user interface (GUI) handles mouse + key events using StdDraw, Draw, DrawListener (see [booksite section 3.6](#))

3. **Sparkle**: Complex GUI using Java Swing framework or other libraries
It’s that simple.

Current turn: b
Enter coordinate of piece you want to move (ex. a1, f8):
s6
Enter coordinate of target square (ex. a1, f8):
Implement simplified or partial version of a game

1. **Standard (text-based or graphical output only):**
   - checkers, sudoku, oregon trail

2. **Sprinkle (simple GUI/more complex rules):**
   - Poker, Snake, 2D role-playing games, Sokoban

3. **Sparkle:**
   - Chess, Go, Tetris, Minesweeper, Brickbreaker

Thinking about game features:
- Implementing specific game rules
- Handling user moves
- Representing and updating the board (or game state)
Sudoku Example

Features for “minimum viable product”

1. Represent Sudoku board
   a. Read board from text file
   b. Save as 2D array
   c. Print board

2. Support human moves
   a. Accept move from standard input (e.g., ROW COLUMN NUMBER)
   b. Update board
   c. Print updated board

3. Implement Sudoku rules
   a. Check that each human move is valid (e.g., digits 1-9 must appear exactly once in each row, column, and 3x3 box)
   b. Prompt user for another move if last move was invalid
Project Theme: Data analysis and/or visualization

Analyze a dataset and produce outputs based on your analysis.

Possible datasets on educational outcomes, policing, genetics, environment, etc. (see Datasets section in spec)

- **Standard:** compute statistics, generate simple graphs
- **Sprinkle/sparkle:** More advanced visualizations, apply advanced machine learning/statistical analysis technique

**Next Monday's workshop** will highlight 3 digital humanities datasets for data analysis/visualization projects!

Thinking about features:
- Reading in and representing data
- Producing an output (e.g., graph or analysis)
public class StdStats

  double max(double[] a) \hspace{2cm} \text{largest value}
  double min(double[] a) \hspace{2cm} \text{smallest value}
  double mean(double[] a) \hspace{2cm} \text{average}
  double var(double[] a) \hspace{2cm} \text{sample variance}
  double stddev(double[] a) \hspace{2cm} \text{sample standard deviation}
  double median(double[] a) \hspace{2cm} \text{median}
  void plotPoints(double[] a) \hspace{2cm} \text{plot points at } (i, a[i])
  void plotLines(double[] a) \hspace{2cm} \text{plot lines connecting } (i, a[i])
  void plotBars(double[] a) \hspace{2cm} \text{plot bars to points at } (i, a[i])
Project Theme: Libraries

Create a simple library (e.g., API) for future COS126 students!

- **Standard/sprinkle:** StdRemix, StdTurtle, SimpleCSV
- **Sprinkle/sparkle:** StdDownload, StdPlot, extend existing COS126 library

Thinking about features:
- **What functions will you implement?**
  For simple functions, group them together as one feature; for complex functions, they can be a standalone feature
Project Theme: Algorithms & Applications

Implement or adapt an algorithm from computer science (sprinkle/sparkle).

- Implementation-oriented
  - computer vision: multi-layer neural networks, image features (e.g., histogram of oriented gradients)
  - natural language processing: topic models, word embeddings, etc.
  - misc.: cellular automata

- Application-oriented:
  - complex Snapchat-like image filters
Project Theme: Assignment Extensions

Assessed on extension not original assignment
Project Theme: Study Tools & New Assignments

\[
\begin{align*}
101001_2 &= 32 + 8 + 1 = 41_{10} \\
32 &\quad 16 &\quad 8 &\quad 4 &\quad 2 &\quad 1
\end{align*}
\]
Further guidance on academic integrity

- When your project involves implementing a well-known game, algorithm and/or library, you should not look at existing implementations.

- For library projects: you may look at APIs of similar libraries and examples of using library functions but not implementations.

- For algorithm projects: You may consult pseudo-code / textbook descriptions.
Further guidance on academic integrity

**matplotlib.pyplot**

`matplotlib.pyplot` is a state-based interface to matplotlib. It provides a MATLAB-like way of plotting.

`pyplot` is mainly intended for interactive plots and simple cases of programmatic plot generation:

```python
import numpy as np
import matplotlib.pyplot as plt

x = np.arange(0, 5, 0.1)
y = np.sin(x)
plt.plot(x, y)
```

The object-oriented API is recommended for more complex plots.

### Functions

- `acorr(x, y[, data])`  
  Plot the autocorrelation of `x`.

- `angle_spectrum(x[, Fs, Fc, window, pad_to, ...])`  
  Plot the angle spectrum.

- `annotate(text, xy, *args, **kwargs)`  
  Annotate the point `xy` with text `text`.

- `arrow(x, y, dx, dy, **kwargs)`  
  Add an arrow to the Axes.
Questions?
End

That's it! Thanks for coming.

Good luck on the proposal.

I'm happy to stay after if you have any questions.