

## Objectives

- Group Work: Designing a Social Network
- Prep for Lab 10

Mar 27, 2017

Sprenkle - CSCI111

1

## Review

- Why classes and objects?
- What methods can we implement to compare two objects of the same type?
  - What does each method header look like?
  - What does each return?
  - How can we use each?

Mar 27, 2017

Sprenkle - CSCI111

2

## DESIGNING CLASSES

Mar 27, 2017

Sprenkle - CSCI111

3

## Summary: Designing Classes

- What does the object/class represent?
- How to model/represent the class's *data*?
  - Instance variable
  - Data type
- What *functionality* should objects of the class have?
  - How will others want to use the class?
  - Put into methods for others to call (API)

### General Class Design:

- **nouns** in a problem are **classes/objects**
- **verbs** are **methods**

Mar 27, 2017

Sprenkle - CSCI111

4

## Top-Down Design

Break down larger problems into pieces that you can solve


- Smaller pieces: classes, methods, functions
- Implement smallest pieces and build up
- We've been doing this most of the semester
  - Typically, program was 1) read input, 2) process input, 3) print result
    - Started putting Step 2 into  $\geq 1$  functions
    - Steps 1 and 3 were sometimes a function
  - Now: on larger scale

Mar 27, 2017

Sprengle - CSC1111

5

## Requirements for a Social Network Application

- Reads social network from two files
  - One file contains people
  - One file contains connections between people
- Add connections between people
  - Symmetric relationship 
- Creates a file to show social network as a graph
- Provides a user interface to do these things

Mar 27, 2017

Sprengle - CSC1111

6

## Designing a Social Network Application

- Break down into pieces
- What classes do we need?
  - What data needed to model those classes?
  - What functionality do each of those classes need?
- What does our driver program (user interface) do?
- How should we implement those classes/program?

### Recall: General Class Design:

- nouns in a problem are classes/objects
- verbs are methods

Mar 27, 2017

## Designs

- For each of your classes
  - Data
  - API

Mar 27, 2017

Sprengle - CSC1111

8

## Social Network Classes/Driver Data

- Person
  - Id
  - Name
  - Friends
- Driver (UI)
  - Social network
- Social Network
  - People in network

What are the data types for each class's data?

Mar 27, 2017

Sprenkle - CSC1111

9

## SN Classes/Driver Functionality

- Person
  - Getters (accessors)
  - String rep
  - Setters
- Driver
  - Getting user input to
    - Read people, connections files
    - Store social network to file
    - Add a person
    - Add connections
  - Summary: call appropriate methods on classes to do above
- Social Network
  - Getters
  - String rep
  - Add people to network
  - Add connections
  - Writing to a file

How should we test these?

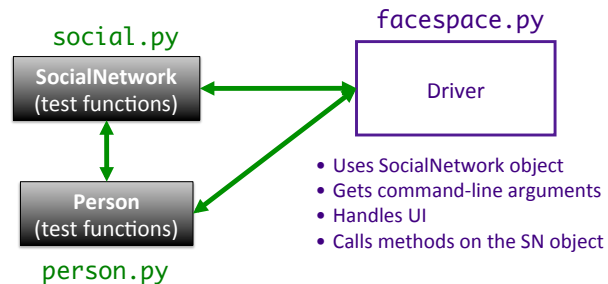
Mar 27, 2017

Sprenkle - CSC1111

10

## Lab 10 Design

- 3 files: person.py, social.py, facespace.py



Mar 27, 2017

Sprenkle - CSC1111

11

## Problem: People Files

- Given a people file that has the format

```

<num_users>
<user_id>
<name>
...
<user_id_n>
<name_n>
    
```

- Write algorithm to create Person objects to represent each person, add to SocialNetwork object

Mar 27, 2017

Sprenkle - CSC1111

12

## Problem: Connection Files

- Given a connection file that has the format

```
<user_id> <user_id>
<user_id> <user_id>
...
<user_id> <user_id>
```

- Each line represents a friend/connection
  - Symmetric relationship
  - Each is a friend of the other
- Update `SocialNetwork` object

Mar 27, 2017

Sprenkle - CSC1111

13

## UI Specification

- Checks if user entered command-line arguments
  - Default files otherwise
- Read people, connections from files
- Repeatedly gets selected options from the user, until user quits
- Repeatedly prompts for new selection if invalid option
- Executes the appropriate code for the selection
- Stops when user quits
- Stores the social network into the file

Write pseudocode

Mar 27, 2017

Sprenkle - CSC1111

14

## UI Pseudocode

Use default files if only one command-line argument

Read people, connections from files

**while** True:

    display menu options

    prompt for selection

**while** invalid option

        print error message

        prompt for selection

    break if selected quit

    otherwise, do selected option

Store social network to designated file

Mar 27, 2017

Sprenkle - CSC1111

15

## Implementation Plan

- Implement `Person` class
  - Test (write test functions, e.g., `testPerson()`)
- Implement `SocialNetwork` class
  - Example runs in lab write up
  - Note: Methods for classes will **not** prompt for input; Use input parameters
  - Test
- Implement driver program

Mar 27, 2017

Sprenkle - CSC1111

16

## Plan for Implementing a Class

- Write the constructor and string representation/print methods first
- Write function to test them
  - See [card.py](#), [deck.py](#), and your [freqobj.py](#) for example test functions
- While more methods to implement ...
  - Write method
  - Test
  - REMINDER: methods should **not** be using `input` function but getting the input as parameters to the method

Mar 27, 2017

Sprenkle - CSCI111

17

## This Week

- Lab 10

Mar 27, 2017

Sprenkle - CSCI111

18