Welcome to

CS 351
Design of Large Programs

Instructor: **Joel Castellanos**

e-mail: joel@unm.edu

Web: http://cs.unm.edu/~joel/

Office: Electrical and Computer Engineering building (ECE). Room 233

1/18/2017

Computer Science Course Sequence

1 CS-105: Intro programming JavaScript and HTML5, currently optional.

1 CS-152: Intro programming Java. More rigorous than 105.

2 CS-251: Intermediate programming Java.

2 3 CS-261: Mathematical Foundations of Computer Science.

3 CS-241: Data Organization using C.

CS 357: Declarative Programming.

CS-351: Design of Large Programs.

CS-361: Algorithms I.

Who Should Be Here?

- Before attempting CS-351, you need to ***already know how to program***.
- For example, you should already be able to use Java to create a Graphical User Interface with working buttons, and sliders. You should be able to draw figures on a JPanel.
- If you have not had CS-152, CS-251, CS-241 AND CS-261, then you should take those first.
- If you do not know how to program, start with CS-152.
- If you know basic programming, but not OOP and have not created GUIs in Java, C++ or some other language, then take CS-251.

3

Course Description

- CS-351 is about designing ***big software***, where big refers to projects with a scope too large to be handled by:
  - Any one person (communication with collaborators)
  - At any one time (communication with "ghosts" of past and "apparitions" of future).
- Any semester long project is "small" by industry standards – but I will do my best to push you to your limits of complexity.
- CS-351 primarily deals with software design, time management, and strategies for completing complex coding.
- Java 1.8.
- IntelliJ (IDE).
- GitHub version control software and repo.

4

Quiz: Write a method in Java or C++ that...

1) Takes no arguments.
2) Does not return a value.
3) Uses ***a nested loop*** to display to the console a 15 x 15 multiplication table.
4) The multiples of 1 (1 through 15) must print on the first line with at least one space between each number.
5) The multiples of 2 (2, 4, 6, ... 30) must print on the second line with at least one space between each number.
6) The same must be true for each of the other multiples.

5

Global Game Jam

UNM hosting NM site for Global Game Jam

Day 1 FRIDAY (1/20): 6:00 PM - Theme reveal

Day 3 SUNDAY (1/22): 5:00 PM - Game Jam Ends

Global Game Jam - a 48 hour game development event. Whether you're a game developer, coder, artist, musician, or have really good ideas, you can be part of this cool event.

REGISTRATION
http://globalgamejam.org/2017/jam-sites/unm-mesa-del-sol

More information:
http://globalgamejam.org/faq

6

Course Resources

Blackboard Learn: https://learn.unm.edu/

- Assignement Drop-box
- Assignement Discussions (blogs)
- Grades

Class website: http://cs.unm.edu/~joel/cs351/

- Syllabus
- Projects
- Lecture Notes
- Readings Assignments
- Source Code

7

Grading

60%: Individual and Group Programming Projects.

20%: Class Participation (quizzes, code reviews, and discussions).

10%: Midterm exam (take home, short essay).

10%: Final exam (take home, short essay).

8

Class Discussion of Reading

Be prepared to discuss:

- What major points were made and/or methods used?
- What was well explained and/or particularly interesting?
- What was confusing and/or uninteresting?
- Did the authors make any unstated assumptions? If so, what are they, what are some reasons they might be false, how likely are those reasons and how would that effect the results?
- Was the writing good? How could it be improved?

9

Grading Discussion of Reading

- I will pick random students from the roster and call on you to answer. Then, I will open each question to general discussion.
- If you did not have a good answer to a question you are asked, than MAKE SURE you do have something intelligent to say in one of the general discussion on that same day.
- In general, there are not specific answers I expect you to have.
- What I do expect, is for you to make it clear to me that you have read the article and have thought intelligently about it.
- If English is not your first language, or for some other reason, you feel worried about expressing yourself orally, then speak with me after class today.

10

Quiz (do on paper, PRINT your name)

- In Java, what is a static initializer and what would be a good reason for using one?
- In Java, what is an instance field initializer? How is an instance field initializer different from a static initializer? Why would it be better to use an instance field initializer rather than placing the code inside the object constructor?

11

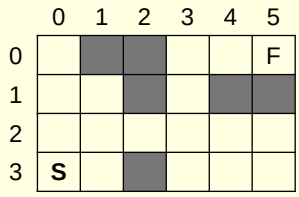
What is a Large Program?

- This course is the Design of Large Programs.
- What does that mean?

12

Quiz: A*

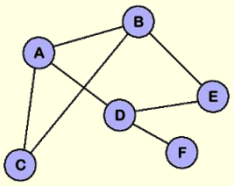
- Given the map below that allows 4 directions of movement with coast of either 1 (empty) or infinity (wall).
- Show the contents of A*'s priority queue after 10 elements have been added.
- Each element of the queue contains three values: x, y and weight. At step 0, the queue contains (0, 3, 8).

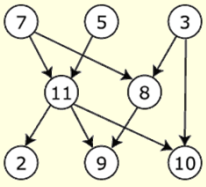


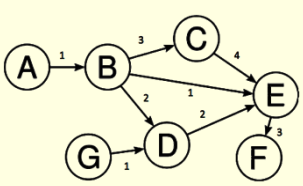
13

Graph (Abstract Data Type)

- In computer science, a graph is an abstract data type that is meant to implement the ***undirected graph*** and ***directed graph*** concepts from mathematics.
- A graph data structure consists of a finite (and possibly mutable) set of vertices (nodes), connected by a set of edges (arcs) for an undirected graph or directed edges (arrows or directed arcs) for a directed graph.







14

Adjacency Matrix for 5 word Dictionary

	green	greed	reed	red	bed
green	0	1	0	0	0
greed	1	0	1	0	0
reed	0	1	0	1	0
red	0	0	1	0	1
bed	0	0	0	1	0

15

Suggestions in Building the Graph (1 of 4)

Create a Node class:

- String name,
- An array of 3 ArrayList<Node> edgeList: one with pointers to all single-change words that are one letter shorter than this word, one for those of the same length and one for those one letter longer.
- Later, after you have build the graph and are ready to start finding the shortest path, you can add fields for visited, costSoFar, and whatever else you might need.

16

Suggestions in Building the Graph (2 of 4)

Create your own dictionary with only 25 or 30 words having 3, 4 and 5 letters. This will allow you to print out your entire graph structure and check it by hand for correctness.

For example, if your dictionary is: bed, greed, green, red, and reed. Then your graph could be printed as:

bed: red
red: bed, reed
reed: red, greed
greed: reed, green
green: greed

This printing order of both nodes and their edges is a natural outcome of the input dictionary being in alphabetical order and keeping words of a length in separate array lists.

17

Suggestions in Building the Graph (3 of 4)

1) Build the graph one word at a time as you read the dictionary.

2) A simple and good structure to hold the word nodes is an array of ArrayLists: one ArrayList for all words of each length found in the dictionary. As a word is read, append it to the appropriate ArrayList and it never needs to move!

3) Just before you add the new node to its ArrayList, walk linearly through each word in the three ArrayLists for words of length ± 1 of the new word. If the examined word is one change from the new word, then 1) add the new word to that found word's edge list and add the found word to the new word's edge list.

After reading the last word, the graph is done!

18

Suggestions in Building the Graph (4 of 4)

The structure outlined on the last slide is simple and efficient:

When finding a path, from word1 to word2, you need to find word1 in the graph:

- Jump to the ArrayList for words of the length of word1.
- By construction, the words in each ArrayList will be in alphabetical order so do a simple and quick binary search.

19

Hamming Distance

- The ***Hamming distance*** between two strings of equal length is the number of positions at which the corresponding symbols are different. It measures the minimum number of substitutions required to change one string into the other.

`"karolin"` `"kathrin"`	`"karolin"` `"kerstin"`
Hamming distance = 3	Hamming distance = 3

20

Levenshtein Distance

The ***Levenshtein distance*** is a string metric for measuring the difference between two sequences. Informally, the Levenshtein distance between two words is the minimum number of single-character edits (i.e. insertions, deletions or substitutions) required to change one word into the other.

Upper and lower bounds:

- It is always at least the difference of the sizes of the two strings.
- It is at most the length of the longer string.
- It is zero if and only if the strings are equal.
- If the strings are the same size, the Hamming distance is an upper bound on the Levenshtein distance.
- The Levenshtein distance between two strings is no greater than the sum of their Levenshtein distances from a third string.

21

Levenshtein Distance ≤ Hamming Distance

An example where the Levenshtein distance between two strings of the same length is strictly less than the Hamming distance:

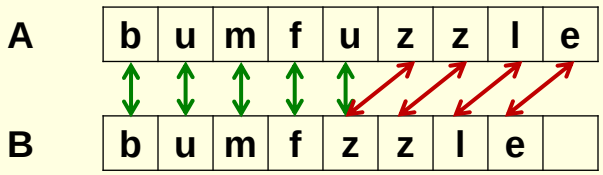
"flaw" and "lawn".

Here the Levenshtein distance equals 2 (delete "f" from the front; insert "n" at the end).

The Hamming distance is 4.

22

Efficient Check: is A one delete from B



- Avoid creating temporary storage (such as breaking each word into every possible substring).
- Avoid comparing a pair of letters more than once.
- Use `.charAt(int idx)` on the existing String.
- Let **`offset`** be the current shift in index from B to A.
- Initialize: **`offset=0`**.
- If, a difference is found and **`offset=0`**, then **`offset=1`**.
- If, a difference is found and **`offset=1`**, then break.

23

Quiz

1) In Java, what are ***generics***?
2) In Java, when are ***static initializers*** called?

24

Project Structure (within .jar)

- Your basic project structure should always be:
  - *projectRoot*`/README.txt`
  - *projectRoot*`/src` for source code in correct package structure.
  - *projectRoot*`/`*toplevelPackage* with .class files in correct package structure.
  - *projectRoot*`/data` for data files and resources.
  - *projectRoot*`/doc` for JavaDoc.
- Do not use the package `com.company`
- Do not use deep and/or mostly empty packages.

25

This code compiles and runs correctly. How can it be improved?

```
for (int i=0; i<getClient().getUser().size(); i++)
{
  card = new
    PolicyCard(getClient().getUser().get(i),
           getClient().getUser().getRegion());

  if (card.votesRequired()==0)
  {
    noVotes.add(getClient().getUser().get(i));
  }
  else
  {
    votes.add(getClient().getUser().get(i));
  }
}
```

26

Why is this stupid?

```
private void initializeProductList()
{ //Add each enum in EnumFood to productList.
  productList = new ArrayList<>();
  productList.add(EnumFood.TROPICAL_FRUIT);
  productList.add(EnumFood.TEMPERATE_FRUIT);
  productList.add(EnumFood.NUT);
  productList.add(EnumFood.GRAIN);
  productList.add(EnumFood.OIL);
  productList.add(EnumFood.VEGGIES);
  productList.add(EnumFood.SPICE);
  productList.add(EnumFood.FISH);
  productList.add(EnumFood.MEAT);
  productList.add(EnumFood.POULTRY);
  productList.add(EnumFood.DAIRY);
}
```

27