Miscellaneous Vocabulary

- Object-Oriented Stuff
- Algorithms
- Software Design
- Testing and Types of Errors
Some terminology:
- *Object-oriented* programming
- *Object-oriented* (programming) language
- *Object-oriented* design
- *Object-oriented* database, etc.

What does the term “object-oriented” mean?
- Based on the idea that all aspects of software – its design, implementation, internal structure – should be based on the real-world objects the software is associated with (i.e., nouns).
- This is in contrast to *structured design*, which focuses on how data flows through a system, and what processes are applied to it (i.e., verbs).
- The OO paradigm is a philosophy that has had, and continues to have, an impact on all aspects of software design and implementation.
- Software can be designed and implemented in a variety of ways, and the OO paradigm is one.
Example - An educational system.

An OO analysis/design would focus on such things as:
- students
- faculty
- courses
- accounts

A structured analysis/design would focus on processes & procedures such as:
- registration
- grade recording
- billing and accounting

You will learn more about the OO paradigm in other classes.
An algorithm is a set of instructions (steps) for solving a problem:
- each step must be clear and precise
- each step must require finite resources
- inputs and outputs must be specified precisely
- the algorithm must be complete
- unnecessary details are left out
- must be finite in size

Analogous to a recipe.

An algorithm may be specified in a number of different formats:
- natural language, e.g., English – ambiguous
- a diagram, such as a “flow chart” – big and cumbersome
- a specific programming language – too many details
- *pseudocode* – a mix of natural and programming languages
Example of an Algorithm

Algorithm for sorting a list of integers:

1) Scan the list to find the smallest value.
2) Move that value to the first position.
3) Among the remaining values, find the next smallest value.
4) Move that value to the second position.
   :
   Continue the above process until the list is sorted.

The above contains several ambiguities…
Algorithm for sorting a list of integers:

**Input:** A list of \( n \) integers, where \( n \geq 1 \).

**Output:** The same list of integers, sorted from smallest to greatest.

1. \( i = 1 \)
2. Repeat the following \( n-1 \) times:
   a. Scan the list to find the smallest value in positions \( i \) through \( n \).
   b. Let \( j \) be the position where the smallest element is found.
   c. Swap the integer in position \( i \) with the integer in position \( j \).
   d. \( i = i + 1 \)

Informally, the above could be referred to as “pseudo-code”
Even with careful programming, most programs will contain errors and must be thoroughly tested.

**Bug** - a mistake in a program.

**Debugging** - fixing mistakes in a program.

**Debugger** – a software tool that helps with debugging.
Generally, there are three types of programming errors:

- syntax
- run-time
- logic
The **syntax** of a programming language is the set of grammatical rules for that programming language.

A **syntax error** is a grammatical mistake in a program.

- misspelling a command, e.g., “pablic” instead of “public”
- missing variable declaration
- missing punctuation
- using things inappropriately, e.g., assigning a string value to an int variable

**Syntax errors:**

- are caught by compiler, hence the phrase *“compiler-time error”*
- relatively easy to fix
- prevent the program from executing
- frequently result in misleading error messages
An error that is detected when you program run (or executed) is called a run-time error*. 
- dividing a number by 0
- accessing memory that was de-allocated
- indexing out of bounds in an array
- reading input from a file that is not open

Run-time errors:
- terminate a programs normal execution
- *occasionally* detected by the compiler
- occur intermittently
- frequently result in misleading, incomplete or non-intelligible messages
- frequently difficult to find and fix

*Note: what is a run-time error in one language might be a logic-error in another
Logic Errors

- Just because a program compiles and runs without producing an error message does *not* mean the program is correct!

- Errors that cause programs to produce incorrect results are *logic errors*.
  - `circleArea = radius * radius;  // pi * radius * radius`
  - `sum = a - b;  // should be sum = a + b;`

- Logic errors:
  - not detected by the compiler
  - produce no run-time error messages
  - cause the program to take incorrect action, or produce incorrect results/output during program execution
  - occur intermittently
  - frequently difficult to find and fix
Run-time and Logic Errors

- Run-time errors and logic-errors can be INCREDIBLY damaging due to their intermittent and after-the-fact nature.

- For that reason, it is imperative that software engineers:
  - test *extensively*, on a variety of different test-cases
  - perform structured walk-throughs

- Testing and walk-thoughts can take a lot of time and money.
1. Mariner Bugs Out (1962)

- **Cost:** $18.5 million
- **Disaster:** The Mariner 1 rocket with a space probe headed for Venus diverted from its intended flight path shortly after launch. Mission Control destroyed the rocket 293 seconds after liftoff.
- **Cause:** A programmer incorrectly transcribed a handwritten formula into computer code, missing a single superscript bar. Without the smoothing function indicated by the bar, the software treated normal variations of velocity as if they were serious, causing faulty corrections that sent the rocket off course.
2. CIA Gives the Soviets Gas (1982)

- **Cost:** Millions of dollars, significant damage to Soviet economy
- **Disaster:** Control software went haywire and produced intense pressure in the Trans-Siberian gas pipeline, resulting in the largest man-made, non-nuclear explosion in Earth’s history.
- **Cause:** CIA operatives allegedly planted a bug in a Canadian computer system purchased by the Soviets to control their gas pipelines. The purchase was part of a strategic Soviet plan to steal or covertly obtain sensitive U.S. technology. When the CIA discovered the purchase, they sabotaged the software so that it would pass Soviet inspection but fail in operation.

- **Cost:** Nearly all of humanity
- **Disaster:** The Soviet early warning system falsely indicated the United States had launched five ballistic missiles. Fortunately the Soviet duty officer had a "funny feeling in my gut" and reasoned if the U.S. was really attacking they would launch more than five missiles, so he reported the apparent attack as a false alarm.
- **Cause:** A bug in the Soviet software failed to filter out false missile detections caused by sunlight reflecting off cloud-tops.

See [http://www.chomsky.info/articles/20140805.htm](http://www.chomsky.info/articles/20140805.htm) for one historical account of this incident.
4. **Medical Machine Kills (1985)**

- **Cost:** Three people dead, three people critically injured
- **Disaster:** Canada's Therac-25 radiation therapy machine malfunctioned and delivered lethal radiation doses to patients.
- **Cause:** Because of a subtle bug called a “race condition,” a technician could accidentally configure Therac-25 so the electron beam would fire in high-power mode without the proper patient shielding.