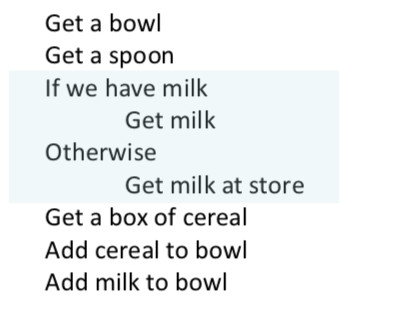
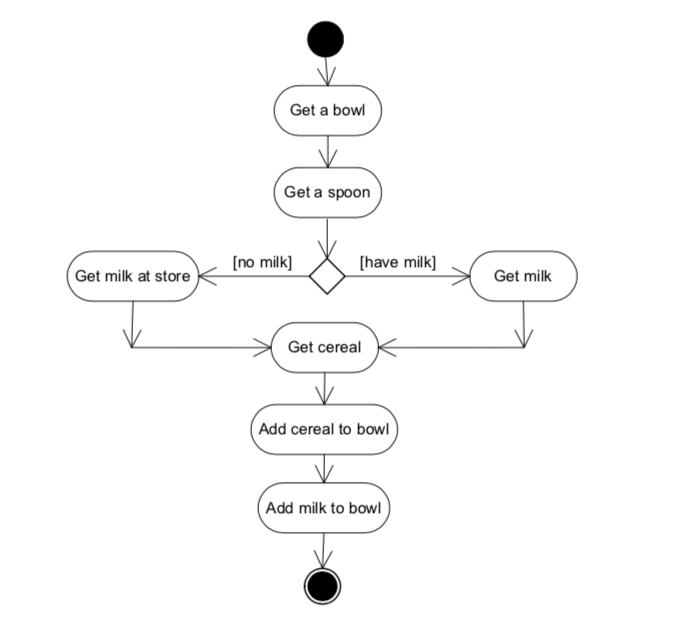
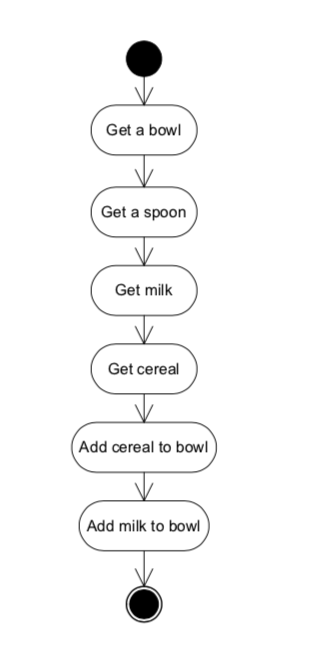
Objectives

* Compare and contrast the common types of programming errors
* Structure simple algorithms to solve computing problems
* Apply the following Java language programming features:
  + Source code comments
  + Basic console input and output

PDF Notes - Introduction to Problem Solving

* 4 important aspect to solving problems with computer programs
  + (1) Thorough understanding of the exact problem
  + (2) Carefully planned strategy for solving the problem
    - Using pseudocode → program design language
    - Top-down stepwise refinement (TDSR) to sketch approach
      * Very high-level step to smaller more specific steps
    - Flowcharting → activity diagrams to provide visual view of a solution procedure
    - Sometimes “thinking out” approach through multiple alternative solutions and choosing the most effective approach
  + (3) Understand the kinds of building blocks that are available for solving the problem
  + (4) Use recommended program design and construction techniques
    - Utility of best practices to design and code programs
    - Helps to improve the readability, reliability, reusability, and understandability of designs and code
* Tips for Formulating Problem Solutions
  + Questions to ask yourself
    - (1) What output information, my must my program produce?
    - (2) What input information, does my program require?
    - (3) How will my program get the input information it needs?
    - (4) What editing, must my program perform ?
    - (5) What information, must be saved?
    - (6) What assumptions can I make?
    - (7) Are my assumptions valid for the problem I’m trying to solve?
    - (8) What would be the impact on the problem solution if my assumptions are not valid?
    - (9) Can i simplify the overall problem to be solved by breaking it into sub-problems?
    - (10) What alternate solutions are possible?
* Algorithms
  + A procedure that can be used to solve a computing problem
  + Specified 2 basic things
    - The set of actions that must be taken
    - The order in which these actions must occur
* Program Control Structures
  + Creating constraints to control steps that precede or succeed other steps
    - Helps specify the order in which certain actions must be performed
  + 3 main program control structures
    - Sequence structures
    - Selection structures (choice and decision structure)
    - Repetition structures (iteration structures)
  + This kind of programming limits the allowable control structures to sequence, selection, and repetition, requires that each control structure have a single point of entry, a single point of exit, and prohibits non-structured transfer of control in a program by statements, such as goto statements
* Modeling Language
  + Unified Modeling Language (UML)
    - Consists of many different types of diagrams used in software development projects
    - I.e. activity diagram
      * Show steps involved in an algorithm and the order which those steps must take place
      * Diamond represents - decision point
      * Square brackets - guard conditions - can be used as T/F
      * “While” is common pseudocode for repetition control structures
        + Also can use “until”, “repeat until”, “do until”



* Variables, Input and Output
  + Commonly using italics to distinguish variables from text

Video Lectures

1. Computing Concepts (1A)
   1. CPU (Central Processing Unit) - instructions loaded and executed
      1. Generally very small in PCs/Laptops
      2. Software processing executed via CPUs
      3. Execute instructions via Machine Language Program
         1. Program loaded into memory
         2. Executed by CPU
   2. Main memory - memory component
      1. Stores instructions and data
      2. Also called RAM (Random Access Memory) - quick access
         1. Only stores when program is running or when computer is turned on
   3. Permanent storage
      1. Disk drives / Hard disks (internal / external)
      2. Flash drives, CDs, floppy disks, etc.
   4. Input / Output
      1. Input: keyboard, mouse, game controller, web camera
      2. Output: display, speakers
   5. Operating System
      1. Ties all these items together
      2. E.g. Windows, Apple MacOS, Linux
   6. Programs
      1. Written in high-level language to write more complex programs
      2. Much more English-like that machine code
      3. This is known as the computer’s source code
      4. Most are written in a high-level language via a compiler
      5. Compiler is software itself
      6. Interpreter - translates instructions one at a time
         1. To run on different types of computers, each interpreter would need to be programmed to run on each type of computer
2. Java History, Architecture and Advantages (1B)
   1. History
      1. Launched in 1996 by Sun Microsystems
      2. Embedded in silicon chips
      3. Programming choice for internet programming
   2. Architecture
      1. Uses both compiler and interpretation
      2. Unique in that source code compiles into bytecode and open into a JVM
      3. Can be written once, compiled once and utilized in any JVM on any operating system and hardware
      4. Compiler is very complex program → takes a lot of work in many different platforms
      5. Much easier to create different types of interpreters instead of compilers
         1. “Write once, run everywhere”
   3. Advantages
      1. Portability
         1. Run on many different computers
      2. Reliability
         1. Highly reliable computer programs → contains many features
      3. Distributed
         1. Built-in features that can distribute much more quickly
      4. Secure
         1. Wish distribution and network features in mind, many more secure environment
         2. Due to JVM
      5. Object-oriented
         1. Strongly supports this approach and built in features to make programs quicker and faster to build
3. The Structure of a Java Program (1C)
   1. Set up basic structure
      1. Public class NAMEOFPROGRAM
         1. All parts of program go between opening and closing brace
         2. Always come in pairs
         3. Name of program must begin with a letter, underscore, or dollar sign
   2. Need to begin with main → public static void main (String[ ] args)
      1. Main method must also have matching braces → contain the body of the main method
      2. Programming instruction for main method belong in this section
      3. E.g. System.out.println (“Java is the bomb!!”);
4. Compiling and Running a Java Program (1D)
   1. Create, compile and run a Java program
   2. Create a source file for the program → where the source code is stored
   3. Compile the source code
   4. Run the program → using the Java interpreter
   5. Use a text-editing program to Create a text file
   6. File name matches the class name exactly → JavaDemo.java
   7. Compile → javac NAMEOFPROGRAM.java
      1. If compiled directly then in source file → then compiled as JavaDemo.class
   8. Java NAMEOFPROGRAM → program will execute and will show in the shell window
   9. IDE
      1. Compile and run within an IDE
         1. Type code in the IDE → helps visually inspect
         2. When ready to compile, compile and any error messages will be shows there
         3. Once compiled, choose Run and the output will be show there
5. Compiling and Running a Java Program from an IDE (1G)
   1. Integrated Development Environment (NetBeans, etc.)
      1. Type in Code / Load Code
      2. Compile
         1. Errors should appears if there are any issues
      3. Run
         1. Output comes up in interactions Window
6. Source Code Comments (1H)
   1. Good way to provide comments → make easier to read and understand
   2. Help to retrace what was originally programmed
   3. 3 ways to add comments
      1. C-style comment → /\* until \*/
         1. Treats all text between the two
         2. Lengthy comments
         3. Top of source file and perform some type of computation
      2. C++ - style comment → // and continues until end of the line → good for a single line
      3. Doc-comment → provide external documentation that can be viewed on a webpage
         1. Javadoc tool → complete webpage → load page into any web browser to look at the document
         2. /\*\* → ends with \*/
         3. Can include tags
            1. Starts with @, includes author, version, param, return
         4. Invoked within command line or IDE
         5. Invoking → javadoc PROGRAMNAME.java
7. Doing Basic Program Output -1 (1I)
   1. System.out.println (“Java rocks!!”); → outputs text in parentheses→ Method Name , Argument List
   2. System.out.print (“ “) → does not generate line feed (new line)
      1. E.g. System.out.print (“The value of x is “);
      2. E.g. System.out.println (x)
      3. + concatenates parts of the string
      4. Parentheses causes the compiler to perform numerical arithmetic
8. Doing Basic Program Output -2 (1J)
   1. Escape sequences
      1. E.g. Joe said “Java is cool”
         1. Can’t write system.out.println (“Joe said “Java is cool” “);
         2. Need to write system.out.println ( “Joe said \”Java is cool\” “);
            1. The \ is an escape sequence
      2. \n → line feed
      3. \t → tab
9. Common Types of Programming Errors (1K)
   1. **Syntax**→ errors that occurs because code contains incorrect Java syntax or running a class that does not exist
   2. **Logic** → errors in the code that causes it to produce an incorrect result for the problem being solved → incorrect formula for a computation
   3. **Runtime** → errors that occur when a program is executed and cause the program to terminate abnormally → e.g. diving by a zero
   4. In IDE → provides error location and what to correct