Outline

 Goals
- Understand an example of program development
- Understand the characteristics of a good algorithm
- Understand how algorithmic patterns can help design programs
Program Development

- One program development strategy has these three steps:
  - Analysis: Understand the problem
  - Design: Organize the solution
  - Implementation: Get the solution running

*Program development* is the progression from analysis to design to implementation.

- We'll see deliverables from all three phases

  *deliverable*: An item that indicates progress was made
Deliverables

- **Analysis deliverable**
  - A list of variable names and some test cases

- **Design deliverable**
  - An algorithm that outlines a solution

- **Implementation deliverable**
  - An executable program
Analysis

- Synonyms for analysis
  - inquiry, examination, study

- Activities
  - Read and understand the problem statement.
  - Name the pieces of information necessary to solve the problem
    - these data names are part of the solution
Problem

Need a program to compute the course grade

Name the data needed to compute an answer

1. ____________________
2. ____________________
3. ____________________
4. ____________________
5. ____________________
6. ____________________

Name data to store the answer

1. ________________
There is more to data than just names

- The data are stored in variables
- Each variable that stores numeric data
  - has a name
  - has a value
  - can be used in a variety of operations such as
    - storing different values into the variable
    - multiply *, divide /, add +, and subtract
    - retrieve the value to be used in expressions
    - display the value
To input or output?

- It help in the early programs if data is distinguished either as input or as output
  - *Input* Information the user must supply to solve the problem
  - *Output* Information the computer must display after the processing has occurred
Test cases help us understand the problem

- A test case includes specific input data, and the output it should generate

**Problem**: Find wholesale price given the markup and retail prices

<table>
<thead>
<tr>
<th>Test Case 1</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>markup</td>
<td>1.0</td>
<td>Input (1.0 represents 100%)</td>
</tr>
<tr>
<td>retailPrice</td>
<td><strong>100.00</strong></td>
<td>Input</td>
</tr>
<tr>
<td>wholesalePrice</td>
<td>50.00</td>
<td>Output</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test Case 2</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>markup</td>
<td>0.5</td>
<td>Input (0.5 represents 50%)</td>
</tr>
<tr>
<td>retailPrice</td>
<td><strong>100.00</strong></td>
<td>Input</td>
</tr>
<tr>
<td>wholesalePrice</td>
<td>66.67</td>
<td>Output</td>
</tr>
</tbody>
</table>
Summary of Analysis

- Things to do during analysis
  - Read and understand the problem
  - Pick variable names that represent the answer
    - This is the output
  - Decide what data the user must enter to get the answer—the input *(the unknowns)*
  - Develop some test cases
  - Create a document that summarizes the analysis
## Other Problems as Analysis Deliverables

<table>
<thead>
<tr>
<th>Mini Problem Description</th>
<th>Variable Names</th>
<th>Input or Output?</th>
<th>Sample Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute the average of three test scores</td>
<td>test1, test2, test3, testAverage</td>
<td>Input, Input, Input, Output</td>
<td>70.0, 80.0, 93.0, 81.0</td>
</tr>
<tr>
<td>Compute a monthly loan payment</td>
<td>amount, annuaIntRate, months, payment</td>
<td>Input, Input, Input, Output</td>
<td>12500.00, 0.08, 48, 303.14</td>
</tr>
</tbody>
</table>
Design

- Synonyms of design: model, think, plan, devise, pattern, propose, outline
- We'll use these design tools:
  - algorithms
  - algorithmic patterns
An algorithm is a set of activities that solves a problem.

An algorithm must:
- list activities that must be performed
- list the activities in the proper order
Algorithm to Bake a Cake

- A recipe (a.k.a. an algorithm)
  - Preheat Oven
  - Grease Pan
  - Mix ingredients
  - Place ingredients into pan
  - place pan in oven
  - remove pan after 35 minutes

- Switch some activities around

  ... what happens if ...
Algorithmic Patterns

- **Pattern**: Anything shaped or designed to serve as a model or guide in making something else

- **Algorithmic Pattern**:
  - Serves as a guide to help develop programs
  - It represents a common type of action that occurs over and over again in programs
  - A solution that can be used in different contexts

- **The Input/Process/Output (IPO) Pattern can be used**
  - This IPO pattern will be used in our first project
# IPO Algorithmic Pattern

<table>
<thead>
<tr>
<th>Pattern:</th>
<th>Input/Process/Output (IPO)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem:</td>
<td>The program requires input data to generate the desired information</td>
</tr>
</tbody>
</table>
| Outline:          | 1. obtain input data from user  
|                   | 2. process input data  
|                   | 3. output the results |
Patterns ala Alexander

"Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice."

From A Pattern Language, Christopher Alexander, Oxford University Press, 1977
Example of Algorithm Design

- The design deliverable will be an algorithm
- The IPO patterns provides a guide to design a more specific algorithm (that is a bit sketchy)

<table>
<thead>
<tr>
<th><strong>IPO Model</strong></th>
<th><strong>IPO applied to Chapter 1 Case Study</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>I)Input</td>
<td>Obtain the items that count for 127A's</td>
</tr>
<tr>
<td>P)rocess</td>
<td>Compute the course grade</td>
</tr>
<tr>
<td>O)utput</td>
<td>Display the course grade</td>
</tr>
</tbody>
</table>
We often need to refine one or more activities (algorithm steps).

- For example, Compute the course grade might now be refined with the mathematical addition + and multiplication * symbols.
Implementation

- Synonyms for Implementation
  - accomplishment, making good, execution
- Implementation deliverable: computer program

<table>
<thead>
<tr>
<th>Activities during Implementation phase</th>
<th>Deliverable</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Translate algorithm into programming language</td>
<td>Source code .java file</td>
</tr>
<tr>
<td>2. Compile source code into byte code</td>
<td>Byte code .class file</td>
</tr>
<tr>
<td>3. Execute the byte code</td>
<td>Running program</td>
</tr>
<tr>
<td>4. Verify program does what it is supposed to</td>
<td>A correct program</td>
</tr>
</tbody>
</table>
Example Translations

- Here are some pseudocode algorithm step translated into the Java programming language
  
  - Display the value of courseGrade
    ```java
    System.out.println("Course grade: "+ courseGrade);
    ```
  
  - Obtain `test1` *Tell user what you want, then "read" it in*
    ```java
    Scanner keyboard = new Scanner(System.in);
    System.out.print("Enter test one: ");
    int test1 = keyboard.nextDouble();
    ```

- Code Demo: Implement the Java Program
Once the algorithm is translated into a programming language abstraction:
- use the compiler to generate byte code
  - you will have errors so you'll need to fix errors, compile, and repeat until there are no compile time errors
- use the Java virtual machine to execute program
- test the program
  - run the program several times with different sets of input data based on sample problems from analysis
  - compare program output to expected results
- fix your program if it doesn't work
Testing occurs at any point in program development:

- **Analysis:** think of some test cases
- **Design:** step through the algorithm
- **Implementation:** run program with several sets of input data

- A running program isn't always right
  - however, testing provides confidence that it works correctly