Outline

- Programs
- Course goals
- About Java
- Models
- OO Concepts: Object, Behavior, Class
- Predefined classes
- Using objects in a program

Programs

- ***Program*** - a description of steps to perform, written with ***code***
- Program is written in a ***programming language***
- A computer runs a program by ***executing*** the code

Algorithm

Algorithm:	a finite set of instructions that specify a sequence of operations to be carried out in order to solve a specific problem or class of problems. [Zwass]

Properties of algorithms:

Finiteness:	Algorithm must complete after a finite number of instructions have been executed.
Absence of Ambiguity:	Each step must be clearly defined, having only one interpretation.
Definition of Sequence:	Each step must have a unique defined preceding and succeeding step. The first step (start step) and last step (halt step) must be clearly noted.
Feasibility:	All instructions must be able to be performed. Illegal operations (division by 0) are not allowed.
Input:	0 or more data values.
Output:	1 or more results.

3-Iteration Newton-Raphson Algorithm

```
Algorithm:   Newton-Raphson (3 iteration version)

1.  Input: a real number X.
2.  If (X < 0) Then
       Output: X " cannot be negative."
       STOP.
    Endif
3.  Set a variable, S, to 1.0.
4.  Set a counter variable, I, to 3.
5.  While (I > 0) Do
       a. Calculate the value (S + X / S) / 2.0.
       b. Set S to the value calculated in step 5a.
       c. Subtract 1 from I.
    Endwhile
6.  Output: "The square root of " X " is about " S
7.  STOP.
```

Does this possess the properties of an algorithm listed on the previous slide?

The Programming Process

Development Process

Problem-Solving Phase

1. Analysis & Specification
2. Design of a General Solution
3. Verification.

Implementation Phase

1. Production of a Specific Solution
2. Test

These are not really just a simple sequence of actions.

Discovered errors will force a re-examination of the algorithm, or perhaps the underlying analysis or even the problem specification.

Pólya's Four-Step Process

Maintenance Phase

1. Use (Install, Execute)
2. Maintain (Modify)

Maintenance often costs more than development and marketing together. Especially if the design is ill-considered or the testing is inadequate.

Pólya's Four-Step Process

There is no recipe for solving general problems. However these ideas are often useful:

1. Understand the problem
    a. Know the boundaries of the problem
    b. Know the constraints on the solution
    c. Know what actions are allowed

2. Devise a plan
    a. Organize thoughts to develop a detailed algorithm
    b. Use tools such as: outlining, flowcharting, and pseudocode.

3. Implement the plan
    a. Carry out the steps in the algorithm
    b. Translate the problem into a language understandable by the device to be used.

4. Test the plan
    a. Did the solution yield appropriate results?
    b. Can the solution be improved?

George Pólya (1887-1985)

Understand the Problem

What are the data?

What is the required result?

What is the starting point (initial conditions)?

Is it at all possible to get the result from the data?

Useful approach: solve the problem by hand.

What steps did you take? Can you write them down?

Is the problem divided into major parts? Can they be identified?

Have any problem assumptions been made? What are they?

Is the solution general?

Devise a Plan

Have you seen a similar problem before?

Do you know of a related problem with a useful solution?

Can you use part of the related problem?

Can the solution of the related problem be modified and used?

Look at the data... repetitions hint at loops in the solution.

Identify the patterns in the data.

If you can't solve the problem, can you solve part of the problem?

Implement the Plan

Check each step of the plan.

Have you considered all the special cases?

Can you arrive at a reasonable result given...

reasonable data?

unreasonable data?

Make certain your solution works for "boundary conditions".

Empty data? Data set too large?

Data sets that results in illegal operations (division by zero)?

Test the Plan

Did you compute any intermediate results that were not used later?

Can they be eliminated?

Can the result be derived differently?

Can the solution be made simpler or more general?

Can the solution method be used for other problems?

Our Tasks

1. To learn how to program
2. To learn the programming language Java
3. To learn how to execute Java programs

About Java

- Language developed by Sun Microsystems
- Intended for small devices (pervasive computing: smart refrigerators, toasters,…)
- Evolved into language for web applications (applets)
- Supposed to allow creating program on one computer that will run on another (portability)
- Not everything it is hyped to be, but is a useful language

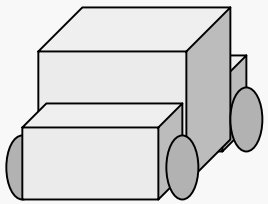
About Programming in Java

- Java is an *object-oriented* language
  - Different from C or COBOL
- Proper use requires different way of thinking about programming
  - Think in terms of building objects that will work together
  - Objects often *model* real-world entities or concepts

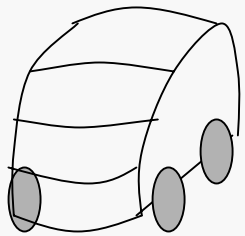
Models

- A model is a simplified representation of something
- Examples:
  - A model car - smaller, exterior looks like actual car, but many details missing
  - A street map - shows relative location of streets at a small scale
- Model serves some purpose

Details and Models



Toy Car



Wind tunnel model

Production

Simple — Complex

Less detail — More detail

Models and Purpose

- Model excludes details of real entity
- Which details are important is determined by purpose
  - Toy car
    - Purpose: play
    - Details: has body and wheels
  - Wind tunnel model
    - Purpose: determine drag of car shape
    - Details: must have exact shape of car

Objects

- General view: An ***entity*** or concept
- Java view: a component of a Java program
- Examples:
  - Physical: car, tire, customer, repair person
  - Conceptual: Order, location, date, complex number

Behaviors

- Behavior of an object are tasks it can perform
- Repair person can move from one location to another, repair objects, etc.
- Behavior includes how object created
- Rules for how object can be used

Classes

- A ***Class*** is a conceptual grouping of similar objects with common behavior
  - Examples: Cars, Repair people, Purchase orders, Buildings
- ***Object*** of class is an *instance* of class
  - Examples: My Saturn SL2, Tom Henderson, PO #A96325, McBryde Hall
- In Java (and C++) programmer writes classes that can be used to create objects

Predefined Classes

- Java comes with libraries of classes
- Programmers can build collections of classes that can be shared
- It is possible to use these classes in many different programs

Referring to an Object

- Reference – a phrase that refers to an object
- Ex: “the seat in the middle of the front row”
- Java example:
  - **System.out**
    - Name of an instance (or object) of the predefined **PrintStream** class
    - This Object represents the console of the computer – where text output is displayed

Messages to Objects

- Message – a request for behavior (or action) from a particular object
  - Send the message to that particular object
- Message to data projector from remote:
  - Receiving object: “projector on ceiling”
  - Message “turn on lamp”

–Java-esque: Projector.lamp(“on”);

Messages in Java

- To send message need to identify
  - Reference to receiving object
  - Name of desired behavior (message “name”)
  - Message contents
- Ex: print “hello!” to console

```
System.out.println(“hello!”)
           ^^^^^^^^^^^^^^^^^^ message
```

Add semicolon to make this a Java statement – like an English sentence

A Java Program

```
import java.io.*;
class Program1 {
        public static void main(String[] arg) {
                System.out.println(“hello!”);
        }
}
```