A Gentle Introduction to Object-Oriented Programming

2 Dec 2005
CMPT14x
Dr. Sean Ho
Trinity Western University

Reminders:

- ***journals in folder***

- ***Paper marking rubric is online***

Review of last time

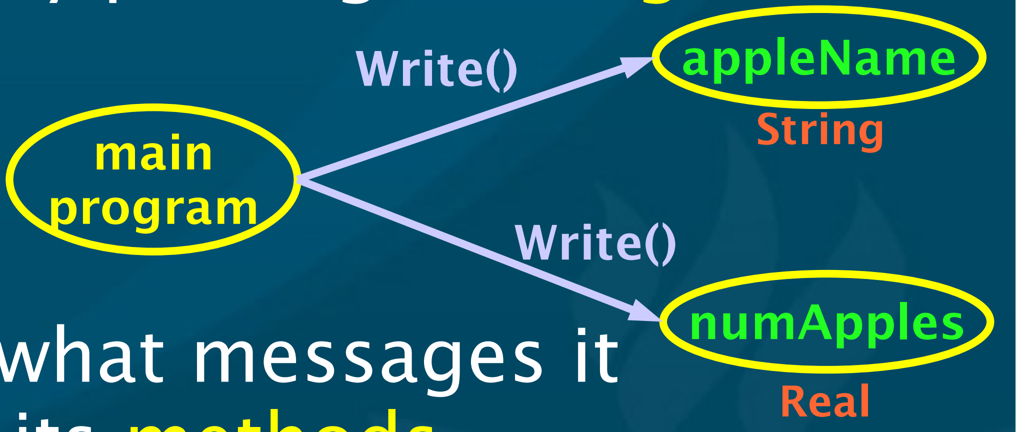
- Trees:
  - Definition of terms:
    - Parent, children, root, leaves, degree, depth, level, forest
  - Depth-first vs. breadth-first search
  - Binary trees: pre/in/post-order traversal
  - Binary search trees (BST):
    - Type definition
    - Search, Insert, Delete
    - Algorithmic efficiency of BST Search

Procedural vs. object-oriented

- Statements in M2 code place the emphasis on the action being performed rather than the data:
  - WriteString ("Ambrosia");
    - **"I will write; here's the string I will write"**
- Procedural code is generally action-oriented;
  - Object-oriented code is data-oriented:
    - appleName := "Ambrosia";
    - appleName.Write();
      - **"appleName, write yourself!"**
      - **(appleName is a string object that knows how to write itself; we don't have to know how it does that)**
- OO languages: C++, Java, SmallTalk, Python

Everything is an object

- In object-oriented programming, all data are objects:
  - Variables, modules, libraries, even exceptions
- We make things happen by passing messages between objects
  - appleName.Write();
  - numApples.Write();



- The object itself defines what messages it accepts: these are called its methods
  - e.g., Strings may have Write() and Length(), but Reals might only have Write()

Methods and attributes

- Everything you can do with an object is encapsulated in its object definition
  - Methods make up the interface to the object
- Objects can also have attributes (variables)
- Our Fractions ADT example:
  - Methods: Numerator(), Denominator(), Add(), Multiply(), Invert(), etc.
    - Everything you need to interact with a Fraction
  - Attributes: frac : ARRAY[1..2] OF INTEGER;
    - Could also have two attributes: num, denom: INT

Classes and instances

- We define (declare) object classes (types)
  - Attributes
  - Methods (interface)
    - Constructor and destructor

Thinking OO

- In procedural form, we might multiply a Fraction by a constant c by:
  - frac2 := Fractions.Multiply (frac1, c);
- In OO form, we ask the frac1 object to multiply itself by c and return the result:
  - frac2 := frac1.Multiply (c);
- But OO is more than just different syntax:
  - OO design process is different:
  - Design objects and interfaces between objects

Designing an OO program

- Simple cash-register example:
  - Type in list of items (name and price)
  - Print subtotal and total including GST
- One possible OO design:

TODO items

- Lab 10 due next week: §11.10 #(25 / 30)
- Paper due next Wed
- Final exam: Wed 14Dec 2-4pm here