Motivating the OO Way

COMP 401, Fall 2018

Lecture 05

TriangleAreaApp example

- Write a program that reads input as sequences of triangle definitions.
  - Each triangle defined by an identifying name as a single word followed by 6 real numbers
    - *ax ay bx by cx cy*
  - Input will end with the word “end”
- For each triangle:
  - Categorize triangle as one of
    - equilateral, isosceles, scalene
  - Report triangle category for each triangle
  - After end of all input
    - Report average size of triangles by category
    - Report area of smallest triangle

TriangleAreaApp – ta.v01

- Write a program that reads input as sequences of triangle definitions.
  - Each triangle defined by an identifying name as a single word followed by 6 real numbers
    - *ax ay bx by cx cy*
  - Input will end with the word "end"
- For each triangle:
  - Categorize triangle as one of
    - equilateral, isosceles, scalene
  - Report triangle category for each triangle
  - After end of all input
    - Report average size of triangles by category
    - Report area of smallest triangle

TriangleAreaApp – ta.v02

- Write a program that reads input as sequences of triangle definitions.
  - Each triangle defined by an identifying name as a single word followed by 6 real numbers
    - *ax ay bx by cx cy*
  - Input will end with the word "end"
- For each triangle:
  - Categorize triangle as one of
    - equilateral, isosceles, scalene
  - Report triangle category for each triangle
  - After end of all input
    - Report average size of triangles by category
    - Report area of smallest triangle

TriangleAreaApp – ta.v03

- Write a program that reads input as sequences of triangle definitions.
  - Each triangle defined by an identifying name as a single word followed by 6 real numbers
    - *ax ay bx by cx cy*
  - Input will end with the word "end"
- For each triangle:
  - Categorize triangle as one of
    - equilateral, isosceles, scalene
  - Report triangle category for each triangle
  - After end of all input
    - Report average size of triangles by category
    - Report area of smallest triangle

TriangleAreaApp – ta.v04

- Write a program that reads input as sequences of triangle definitions.
  - Each triangle defined by an identifying name as a single word followed by 6 real numbers
    - *ax ay bx by cx cy*
  - Input will end with the word "end"
- For each triangle:
  - Categorize triangle as one of
    - equilateral, isosceles, scalene
  - Report triangle category for each triangle
  - After end of all input
    - Report average size of triangles by category
    - Report area of smallest triangle

Review of non-OO approach

- All functions are static
- Variables are all either declared locally or passed in as parameters
- Static class functions simply act as a library of triangle-related functions used by our application.

Thinking with an object mindset

- Consider the role of ax, ay, bx, by, cx, cy
  - As a collective, they represent a specific triangle.
  - Consider the functions for finding area and classifying
    - Onus on us to provide this information as parameters
    - As well as ensuring that they actually represent a triangle.
- Object-oriented programming flips this relationship.
  - Formalizes the collective meaning of these pieces of information as an abstraction.
  - Abstraction provides means to query properties and invoke “behavior”

Step 1: Name the abstraction

- In Java this means create a class corresponding to the abstraction's name.
- ta.v05

Step 2: Declare its fields

- The fields of your abstraction are pieces of information that collectively define it.
  - Declared like variables
    - Must specify type and adhere to variable naming rules.
    - Declared in class definition
      - NOT within a method, but floating off by themselves.
      - Good idea to keep them together at the top of the class.
- Here you start to make design decisions
  - In our example, triangles defined by 3 coordinates.
    - How else could a triangle be defined?
  - Note that part of this is deciding on types as well.
    - What would be impact of choosing something other than double?
- ta.v06

Step 3: Define a constructor

- Constructor is a special type of method
  - Job is to create and initialize a new instance.
  - Declaration differs from a normal method
    - Name must match class name.
    - Does not have a any sort of return value in its signature.
  - Within the constructor, the keyword *this* refers to the new object to be initialized.
  - Any information needed should be passed in as parameters.
    - Code in the constructor is responsible for making sure that the fields of *this* are appropriately set.
- To call a constructor, use the *new* keyword
  - Result will be a new instance (i.e., object) of the class
- ta.v07

Step 4: Define instance methods

- Functions/procedures that depend on the specific instance
  - What functions in our example calculate values specific to a particular triangle?
    - triangle_category
    - triangle_area
- Declare instance methods without "static" keyword
  - That is what makes it an instance method.
- Instance methods only make sense in the context of a specific instance.
  - Must be called with the "." operator using a reference to an object.
  - *reference*.*method*()
- Within an instance method, the keyword *this* provides a reference to the object itself.
  - To get value of a particular instance field: this.*field*
  - If unambiguous, then "this" can be left off.
    - Must use *this* keyword if a local variable or parameter has the same name as the field. Also known as "shadowing".
- ta.v08

Another improvement

- Notice that within area() and category() we end up calculating side lengths.
  - Would be better if we simply provided methods for retrieving each of the side lengths
    - this.side_ab()
    - this.side_bc()
    - this.side_ca()
  - Implied *this* also works for method names
    - Don't need to use *this* keyword if the method is being called for the "current" object (i.e., on "this" object).
- ta.v09
  - Also, moved static helper function point_distance from TriangleAreaApp to Triangle where it is actually used.

Repeating with Point

- Consider role of ax, ay within Triangle class
  - Collectively they represent a point
  - Same with bx, by and cx, cy
- Opportunity for abstraction again.
- ta.v10
  - Notice name conflict in constructor between parameters passed in and field names.
    - Forces use of this keyword when assigning fields.
    - This is a common idiom for constructors.
      - But otherwise, you generally want to avoid having method parameter names or local variable names that "shadow" field names.
  - Calculating distance to another point is now an instance method associated with a point itself.

Classes and Objects

- Fundamental units of *abstraction*
- Physical Analogy
  - Classes are like factories
    - Contain a blueprint for an object
      - Defines the inner workings (i.e., fields aka members)
      - Defines what it can do, its "behavior" (i.e., instance methods)
    - Factory itself may have some capabilities
      - Class members and class methods
      - Useful for defining named constants and helper methods that are related to the abstraction as a whole but not specific to an instance.
  - Objects are what the factory builds
    - Each object is an instance of a class
    - Name of the class is the "type" of the object.
      - Which means the class name is the type we use for a variable that can reference the object.

Objects as state

- An object is defined by its state
  - Collection of named fields that represent information about the object
    - The current values assigned to those fields reflect the "state" of the object
- Object design reflects purpose
  - What fields to include in an object will depend on how that object is to be used and the kinds of operations that object will be involved in.

Static class fields

- Instance fields are data associated with each instance
  - Every object has it's own set of values
- Class fields are data associated with the class as a whole.
  - Declared like an instance field, but with "static" keyword.
  - Like class methods, you can access them via the class name or directly by code within the class.
- Most common use:
  - Named Constants
    - Best practice: all caps, initialized when declared, declared with "final" keyword to indicate that it won't ever change.
- ta.v11