Python Tutorial – Part 2: Objects and Classes

Mark A. Austin

University of Maryland

austin@umd.edu
ENCE 688P, Spring Semester 2022

February 20, 2023

Overview

Part 3

Composition of Object Models

Composition of Object Models

Definition

Composition is known as is a part of or is a relationship.

The member object is a part of the containing class and the member object cannot survive or exist outside the enclosing or containing class or doesn't have a meaning after the lifetime of the enclosing object.

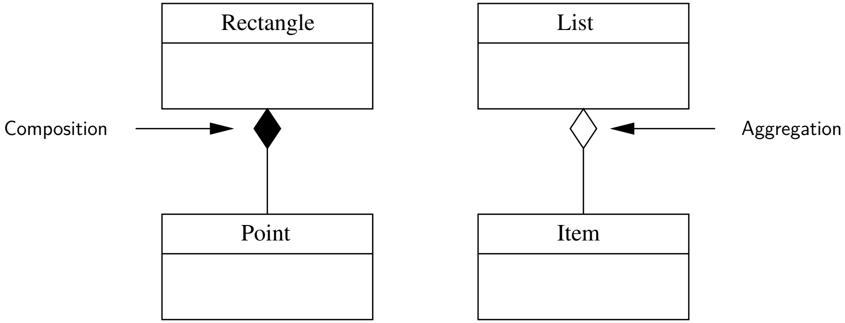
Is it Aggregation or Composition?

- Ask the question: if the part moves, can one deduce that the whole moves with it in normal circumstances?

Example: A car is composition of wheels and an engine. If you drive the car to work, hopefully the wheels go too!

Composition of Object Models

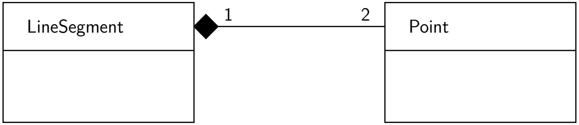
Notation for Aggregation and Composition



Recall: Aggregation is all about grouping of things ...

Example 7. Modeling Line Segments

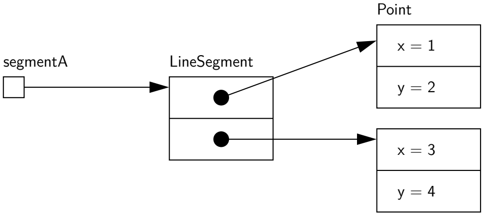
Model Composition



Creating a line segment object with:

```
segmentA = LineSegment( 1, 2, 3, 4 );
```

should give a layout of memory:



Example 7. Modeling Line Segments

Part I: Line Segment Object Model

```python
# ======================================================================
# LineSegment.py: Line segments are defined by end points (x1, y1) and
# (x2, y2). Compute length and angle of the line segment in radians.
#
# Written by: Mark Austin                                October, 2022
# ======================================================================

import math

from Point import Point

class LineSegment:
  __length = 0
  __angle  = 0

  def __init__(self, x1, y1, x2, y2 ):
    self.__pt1 = Point(x1,y1)                 # <-- Object composition ...
    self.__pt2 = Point(x2,y2)                 # <-- Object composition ...
    self.__length = self.__pt1.distance(self.__pt2)
    self.__angle  = self.getAngle()

  # Compute angle (radians) for coordinates in four quadrants ....

  def getAngle(self):
     dX = self.__pt2.get_xCoord() - self.__pt1.get_xCoord();
     dY = self.__pt2.get_yCoord() - self.__pt1.get_yCoord();
```

Example 7. Modeling Line Segments

Part I: Line Segment Object Model (Continued) ...

```

      if dY >  0.0 and dX == 0.0:
         angle = math.pi/2.0
      if dY >= 0.0 and dX > 0.0:
         angle = math.atan( dY/dX )
      if dY >= 0.0 and dX < 0.0:
         angle = math.pi + math.atan( dY/dX )
      if dY < 0.0 and dX < 0.0:
         angle = math.pi + math.atan( dY/dX )
      if dY < 0.0 and dX >= 0.0:
         angle = 2*math.pi + math.atan( dY/dX )

      return angle

   # String represention of line segment ...

   def __str__(self):
      x1 = self.__pt1.get_xCoord();
      y1 = self.__pt1.get_yCoord();
      x2 = self.__pt2.get_xCoord();
      y2 = self.__pt2.get_yCoord();
      return "--- LineSegment: (x1,y1) = (%5.2f, %5.2f), (x2,y2) = (%5.2f, %5.2f),
                  angle = %.2f, length = %.2f" % ( x1, y1, x2, y2, self.__angle, self.__l
```

Example 7. Modeling Line Segments

Part II: Line Segment Test Program

```
# ====================================================
# TestLineSegment.py: Exercise line segment class ...
# ====================================================

from LineSegment import LineSegment

# main method ...

def main():
    print("--- Enter TestLineSegment.main()    ... ");
    print("--- ============================== ... ");

    print("--- Part 1: Create test line segment ... ");

    segmentA = LineSegment( 1.0, 2.0,  3.0,  4.0 )
    print(segmentA)

    print("--- Part 2: Sequence of line segments ... ");

    a = LineSegment( 0.0, 0.0,  3.0,  0.0 )
    print(a)
    b = LineSegment( 0.0, 0.0,  3.0,  3.0 )
    print(b)
    c = LineSegment( 0.0, 0.0,  0.0,  3.0 )
    print(c)
    d = LineSegment( 0.0, 0.0, -3.0,  3.0 )
    print(d)
```

Example 7. Modeling Line Segments

Part II: Line Segment Test Program (Continued) ...

```
    e = LineSegment( 0.0, 0.0, -3.0,  0.0 )
    print(e)

    print("--- =============================== ... ");
    print("--- Finished TestLineSegment.main() ... ");

# call the main method ...

main()
```

Part III: Abbreviated Program Output:

```
--- Part 1: Create test line segment ...
--- LineSegment: (x1,y1) = ( 1.00,  2.00), (x2,y2) = ( 3.00,  4.00), angle = 0.79, length = 2.83
--- Part 2: Sequence of line segments ...
--- LineSegment: (x1,y1) = ( 0.00,  0.00), (x2,y2) = ( 3.00,  0.00), angle = 0.00, length = 3.00
--- LineSegment: (x1,y1) = ( 0.00,  0.00), (x2,y2) = ( 3.00,  3.00), angle = 0.79, length = 4.24
--- LineSegment: (x1,y1) = ( 0.00,  0.00), (x2,y2) = ( 0.00,  3.00), angle = 1.57, length = 3.00
--- LineSegment: (x1,y1) = ( 0.00,  0.00), (x2,y2) = (-3.00,  3.00), angle = 2.36, length = 4.24
--- LineSegment: (x1,y1) = ( 0.00,  0.00), (x2,y2) = (-3.00,  0.00), angle = 3.14, length = 3.00
```

Source Code: See: python-code.d/classes/