

## CSI31 Lecture 14

**Topics:** *Chapter 5. Objects and Graphics*

5.4 Using Graphical Objects

5.5 Graphing Future Value

## 5.4 Using Graphical Objects

Let's continue to talk about graphics library and OO approach.

We found out, that we can draw several different kinds of objects: graphwin, point, circle, line and so on. Each of these kinds are examples of **classes**.

So we have class of points, circles, lines, ovals, rectangles, graphwin and so on.

When we write an assignment `p1=Point(10,30)`, the following happens:

An *instance* of class **Point** is created (called object), an is assigned to variable **p1**.

In over words, we never «use» classes themselves, we create instances of classes (objects) and work with them.

Every object is an instance of some class, and the class describes the properties the instance has.

When we define a class we

1. define a **constructor(s)** (expression that creates instance of this class)
2. define variable(s)
3. define **method(s)** (function(s) )

## 1. define a constructor

constructor is an expression that creates brand new object.

The general form is: `<class-name>(<param1>, <param2>, ...)`

Name of class (Circle or Point or ...)

Parameters that are required to initialize the object

Examples: class Point

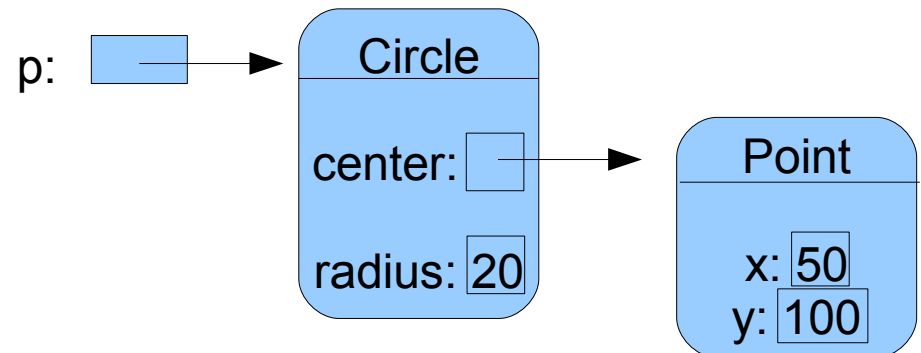
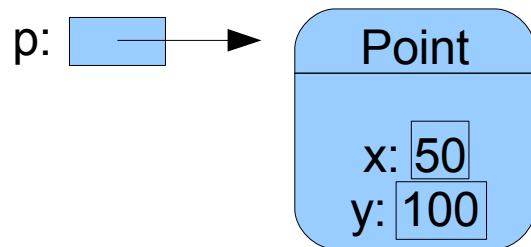
`p = Point(50, 100)`

- the class name is `Point`, and here is a call to the constructor with two parameters (x-coordinate, y-coordinate)

class Circle

`c = Circle(Point(50, 100), 20)`

- the class name is `Circle`, the constructor needs two parameters: center (should be of type Point), and radius.



## 2. define variable(s)

if you recall the Point class, x-coordinate value and y-coordinate value are stored as instance variables inside of the object.

So Point class has at least two variables for x- and y-coordinates.

## 3. define method(s) (function(s) )

To perform an operation on an object we «send» the object a message.

Recall previous lecture example with drawing a point:

```
p1=Point(30,30)
p1.draw(win) ← «message» to the object p1
```

The second line calls a method «draw» of object `p1`. «draw» is a function (method) of class Point, that draws a point. It requires one parameter (where to draw the object).

The general form of a method call:

```
<object>.<method-name>(<param1>, <param2>, ...)
```

Methods can be without parameters (when they are not needed): `p1.getX()`

Methods can change the values of an object's instance variables, hence changing the **state** of an object (called **mutators**):

```
p1.move(10, 30) – moves the point 10 pixels to the right and 30 pixels down.
```

The general form of the move method is `move(dx, dy)`. All the graphical objects have this method (in graphics library)

! It is possible for two different variables to refer to exactly the same object. Changes made to the object through one variable are visible to the other one.

Example: assume I want to draw two ovals of the same size at two different places. So here is my program: see [ovals.py](#)

How to correct a problem?:

1. write a separate code for the right oval, or
2. clone the first one ( method `clone()` ) see [ovals-corrected.py](#)

## 5.5 Graphing Future Value

Recall our example [Future Price](#) from **Lecture 5** (last slide in lecture slides):

It is difficult to make a budget that spans several years, because prices are not stable. If your company needs 200 pencils per year, you cannot simply use this year's price as the cost of pencils two years from now. Because of inflation the cost is likely to be higher than it is today.

Write a program to gauge the expected cost of an item in a specified number of years. The program asks for the cost of the item, the number of years from now that the item will be purchased, and the rate of inflation. The program then outputs the estimated cost of the item after the specified period.

Have the user enter the inflation rate as a percentage, like 5.6 (%). Your program should then convert the percent to a fraction, like 0.056, and should use a loop to estimate the price adjusted for inflation.

### **Algorithm (design):**

print an introduction

input price

input years

input inflation

repeat years times:      $\text{price} = \text{price} + \text{price} * \text{inflation}$

Output the final price

Let's do it in graphics!