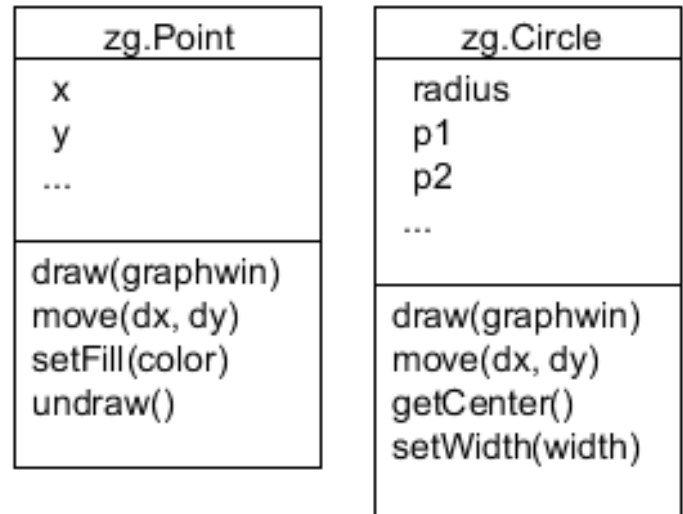


Name: _____

Use this quiz to help make sure you understand the videos/reading. **Answer all questions.** Make additional notes as desired. **Not sure of an answer?** Ask your instructor to explain in class and revise as needed then. **Turn this in via the Session 3 Dropbox on our Moodle site.**

Throughout, where you are asked to “circle your choice”, you can circle or underline it (whichever you prefer).

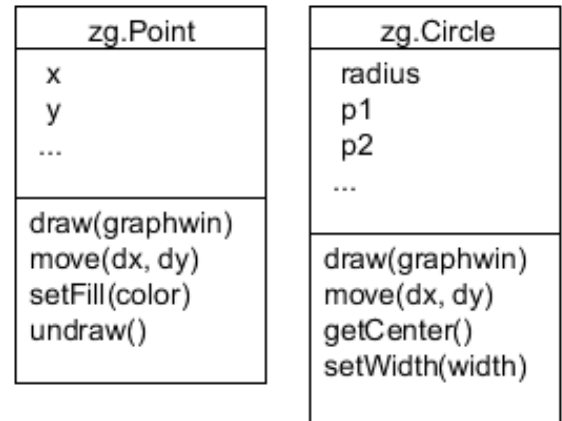
Handout: **Using Objects**
(Objects, Types and Values – and Classes)



- The diagrams to the right are called _____ Class Diagrams, where _____ stands for Unified Modeling Language. (Fill in both blanks with the (same) 3-letter acronym for **Unified Modeling Language**.)
- Consider the 2 UML class diagrams shown above and to the right. What are the names of the two **classes** shown?
- Consider the UML class diagram for the **Point** class shown above. For that class:
 - What are the names of the two **instance variables (fields)** that are shown?
 - What do you think that those fields represent? (You can't tell this authoritatively from the UML class diagram; just make your best guess based on the names of the fields.)
 - How many **methods** are shown?
 - How many parameters does the **draw** method require?
 - How many parameters does the **move** method require?
 - How many parameters does the **undraw** method require?
- Consider the UML class diagram for the **zg.Circle** class shown above. For that class:
 - What are the three **instance variables (fields)** that are shown?
 - How many **methods** are shown?
 - How many parameters does the **getCenter** method require?
 - What do you think that the **getCenter** method does? (You can't tell this authoritatively from the UML class diagram; just make your best guess based on the name of the method.)

- What do you think that the *setWidth* method does? (You can't tell this authoritatively from the UML class diagram; just make your best guess based on the name of the method.)

In the following, continue to use the UML class diagrams for *Point* and *Circle* (repeated to the right for your convenience). Make additional reasonable assumptions as needed.



- Write a statement that constructs a *zg.Point* at (75, 30), putting the constructed object into a variable called *p*.
- Suppose that you have two *zg.Point* objects in variables *p1* and *p2*, respectively. Write a statement that constructs a *zg.Line* from *p1* to *p2*, putting the constructed object into a variable called *line1*.
- Suppose that you have two *zg.Point* objects *s* in variables *p1* and *p2*, respectively. Write statements that set *p1*'s fill color to 'red' and *p2*'s fill color to 'blue'.
- Suppose that you have a *zg.Circle* in variable *circle1*.
 - Write a statement that sets the variable *p* to *circle1*'s center.
 - After the above statement executes, variable *p* is (presumably) a *zg.Point*. Using that fact, write additional statements that set variables *x1* and *y1* to the x and y coordinates of *circle1*'s center.
- True or False: If you had a *zg.Circle* in variable *circle2*, the following statement would set variable *x2* to the x-coordinate of *circle2*'s center:

$$x2 = circle2.getCenter().x$$

True or False (circle your choice)
- Hint to the previous problem: The answer is *True*. Now, explain briefly WHY there are parentheses after *getCenter* in the above statement but NOT after *x*.

Handout: **Counted Loops and Range Expressions**

11. Write the sequence of numbers that each of the following range expressions generates:

- **range(5)** – generates the sequence:
- **range(2)** – generates the sequence:

12. When the code snippet below runs, what gets printed?

```
for k in range(3):  
    print(k, k + 20)  
    print(3 * 'hello')
```

13. Write the sequence of numbers that each of the following range expressions generates:

- **range(1, 5)** – generates the sequence:
- **range(2, 12, 2)** – generates the sequence:
- **range(12, 2, -2)** – generates the sequence:
- **range(2, 4)** – generates the sequence:
- **range(4, 2)** – generates the sequence:

14. Write a loop that prints **'funny'** 40,000 times.

15. Write a loop that prints the cubes of the numbers from 35 to *m* (where *m* is some integer bigger than 35).

Textbook Reading: **Section 1.7 Problem Solving – Algorithm Design** (pages 16 – 22) and **Section 2.3 Problem Solving – First Do It By Hand** (pages 45 – 46).

16. Do **problem 17 on page 47** of your textbook. **Don't spend a long time on this problem;** just enough to feel that you understand what pseudocode is and how to use the “First Do It By Hand” advice. The pseudocode for this problem will be trivial; the “work” is the “do it by hand”.

Textbook Reading: **Section 2.4 Strings** (pages 48 – 51) and **Sections 2.5.1 – User Input** and **2.5.2 – Numerical input** (pages 55-56).

17. Consider the following statement:

```
s = input('Enter your name: ')
```

Write a statement that sets the variable `t` to the string that is the same as `s`, followed by an upper-case version of `s`, followed by a lower-case version of `s`. For example, if the human enters `Rhonda`, so that the value of `s` is `'Rhonda'`, then your statement should make `t` have the value `'RhondaRHONDArhonda'`.

18. Consider the following statement:

```
s = input('Enter your name: ')
```

Suppose that the human user enters `43` in response to the prompt for input. What will the value of `s + s` be in that case?

19. Continuing the previous problem, what if the software developer intended that the value of `s + s` be `86` if the user enters `43` as the input (and similarly for other inputs). What should the programmer change or add?