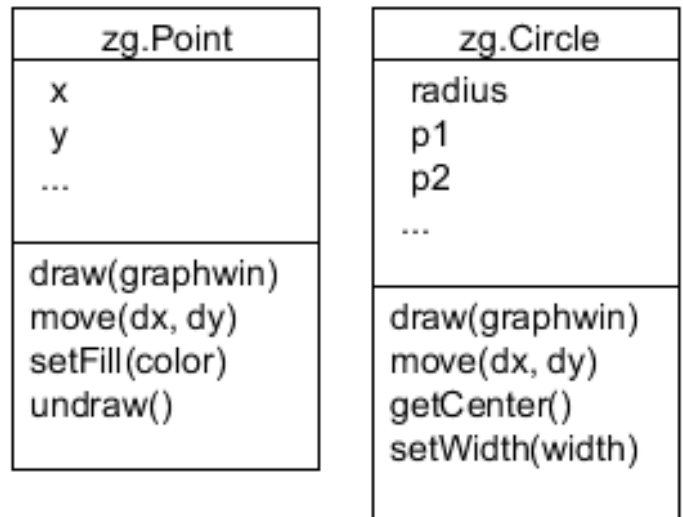


Name: _____

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.

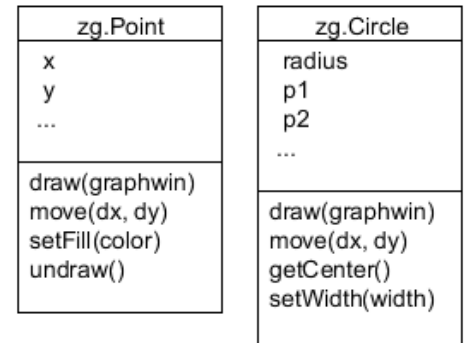
Handout: **Using Objects**
(Objects, Types and Values – and Classes)



- 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 **zg.Point** class shown above. For the zg.Point class:
 - What are the names of the two **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 the zg.Circle class:
 - What are the three **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 `zg.Point` and `zg.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.Points` 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.Points` 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**.

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)  
    print(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).