CL09 - OOP Practice

Feedback form! (Extra credit!)

- Link on Campuswire!

Hack 110!

- Comp 110 Hackathon!
- 7:00 PM on April 21st to 7:00 AM on April 22nd in Sitterson Lobby
- Two tracks: web development and game development
- Must also attend Friday, April 14th 6 PM workshop
- *Great for beginners!*
- Free food + swag!
- https://forms.gle/DicV5fKfN6FDTedL6 (Link on Sakai)
- Apply by today at 5 pm! (Decisions made TONIGHT.)

Memory Diagram

```python
class Profile:

    handle: str
    followers: int
    is_private: bool

    def __init__(self, handle: str):
        """Constructor to initialize attributes"""
        self.handle = handle
        self.followers = 0
        self.is_private = False

    def tweet(self, msg: str) -> None:
        """Print out handle and message only if account is public"""
        if not self.is_private:
            print(f"@{self.handle} tweets {msg}")

    def toggle_privacy(self) -> None:
        self.is_private = not self.is_private

a: Profile = Profile("alyssa")
b: Profile = Profile("tyler")
a.tweet("Sup")
b.toggle_privacy()
b.tweet("Heyyy")
```

```python
class Profile:

    handle: str
    followers: int
    is_private: bool

    def __init__(self, handle: str):
        """Constructor to initialize attributes"""
        self.handle = handle
        self.followers = 0
        self.is_private = False

    def tweet(self, msg: str) -> None:
        """Print out handle and message only if account is public"""
        if not self.is_private:
            print(f"@{self.handle} tweets {msg}")

    def toggle_privacy(self) -> None:
        self.is_private = not self.is_private

a: Profile = Profile("alyssa")
b: Profile = Profile("tyler")
a.tweet("Sup")
b.toggle_privacy()
b.tweet("Heyyy")
```

Magic Methods Review

```python
from __future__ import annotations

class Point:
    """Model a 2D Point"""

    x: float
    y: float

    def __init__(self, x: float, y: float):
        """Initialize a point with its x,y components"""
        self.x = x
        self.y = y

    def __str__(self):
        """Print prettier version of our point"""
        return f"({self.x},{self.y})"

    def __mul__(self, factor: float) -> Point:
        scaled: Point = Point(self.x * factor, self.y * factor)
        return scaled
```

Code Writing - Operator Overload Magic Method

- Write a magic method so that you can add a Point with a float using +. (In other words, for Point a, I want to be able to write a + 3.0)
- Method name: __add__
- Then, write code to call your method using the + operator.

Code Writing - Magic Method

Let's go back to our Profile class.
Define a __str__ magic method so that calling:

```
print(a)
print(b)
```

Would result in the output:

```
SN: alyssa; Followers: 0; Public
SN: tyler; Followers: 0; Private
```

```python
class Profile:

    handle: str
    followers: int
    is_private: bool

    def __init__(self, handle: str):
        """Constructor to initialize attributes"""
        self.handle = handle
        self.followers = 0
        self.is_private = False

    def tweet(self, msg: str) -> None:
        """Print out handle and message only if account is public"""
        if not self.is_private:
            print(f"@{self.handle} tweets {msg}")

    def toggle_privacy(self) -> None:
        self.is_private = not self.is_private

a: Profile = Profile("alyssa")
b: Profile = Profile("tyler")
a.tweet("Sup")
b.toggle_privacy()
b.tweet("Heyyy")
```