CS 421 --- Objects Activity

Manager	Keeps team on track	
Recorder	Records decisions	
Reporter	Reports to class	
Reflector	Assesses team performance	

Objectives

In this activity, you will:

- implement objects using two separate techniques
- implement inheritance

Part 1 --- A Counter

Here is a counter, similar to the one in the state lecture. A Jupyter notebook containing this code can be found in the `release` repository in the `examples-objects` branch.

```python
def mkInc(init=0):
    ct = init
    def inc(delta=1):
        nonlocal ct
        ct = ct + delta
        return ct
    return inc

c1 = mkInc()
```

Problem 1) We mentioned last time (or should have, anyway) that objects and closures are very similar. In what way does `mkInc` resemble an object? What is the constructor? What are methods? Is there private and public data?

Problem 2) What is missing from this story about objects?

Part 2 --- Multiple Methods

Here is a trick to introduce multiple methods.

```python
def mkInc(init=0):
    ct = init
    def inc(delta=1):
        nonlocal ct
        ct = ct + delta
        return ct
    def reset(init=0):
        nonlocal ct
        ct = init
        return ct
    return (inc,reset)

(c2,r2) = mkInc()
```

Problem 3) We now have multiple methods! Would you be happy programming with an object system like this? Why or why not?

Problem 4) Add a method `dec` to this that decrements a counter.

Part 3 --- Dictionaries

Dictionaries greatly improve our quality of life.

```python
def mkInc(init=0):
    ct = init
    def inc(delta=1):
        nonlocal ct
        ct = ct + delta
        return ct
    def reset(init=0):
        nonlocal ct
        ct = init
        return ct
    return { "inc": inc, "reset": reset }
```

Problem 5) How would you add `dec` to this version of objects?

Problem 6) Using a dictionary allows us to simulate inheritance. Here is some starter code: try to write a ``class'' `mkFastInc` that doubles the increment each time the `inc` method is called. It should call the superclass methods whenever possible.

```python
def mkFastInc(init=0):
    superInc = mkInc(init)
    --- your code here!
```

Problem 7) Suppose you wanted to be able to access the state of the superclass directly. What options do you have to do that?

Objects Activity --- Team's Assessment (SII)

Manager or Reflector: Consider the objectives of this activity and your team's experience with it, and then answer the following questions after consulting with your team.

1. What was a **strength** of this activity? List one aspect that helped it achieve its purpose.

2. What is one things we could do to **improve** this activity to make it more effective?

3. What **insights** did you have about the activity, either the content or at the meta level?

Objects Activity--- Reflector's Report

Manager	Keeps team on track	
Recorder	Records decisions	
Reporter	Reports to Class	
Reflector	Assesses team performance	

1. What was a strength of your team's performance for this activity?

2. What could you do next time to increase your team's performance?

3. What insights did you have about the activity or your team's interaction today?