CS 421 --- Objects Activity

<table>
<thead>
<tr>
<th>Manager</th>
<th>Keeps team on track</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recorder</td>
<td>Records decisions</td>
</tr>
<tr>
<td>Reporter</td>
<td>Reports to class</td>
</tr>
<tr>
<td>Reflector</td>
<td>Assesses team performance</td>
</tr>
</tbody>
</table>

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
1 def mkInc(init=0):
2     ct = init
3     def inc(delta=1):
4         nonlocal ct
5         ct = ct + delta
6         return ct
7     return inc
8
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
1 def mkInc(init=0):
2     ct = init
3     def inc(delta=1):
4         nonlocal ct
5         ct = ct + delta
6         return ct
7     def reset(init=0):
8         nonlocal ct
9         ct = init
10        return ct
11     return (inc,reset)
12 (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
1 def mkInc(init=0):
2     ct = init
3     def inc(delta=1):
4         nonlocal ct
5         ct = ct + delta
6         return ct
7     def reset(init=0):
8         nonlocal ct
9         ct = init
10        return ct
11     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
1 def mkFastInc(init=0):
2     superInc = mkInc(init)
3     --- 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?
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 thing 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

<table>
<thead>
<tr>
<th>Role</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manager</td>
<td>Keeps team on track</td>
</tr>
<tr>
<td>Recorder</td>
<td>Records decisions</td>
</tr>
<tr>
<td>Reporter</td>
<td>Reports to Class</td>
</tr>
<tr>
<td>Reflector</td>
<td>Assesses team performance</td>
</tr>
</tbody>
</table>

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?