Lab Exercises: More Practice with Object-Oriented Programming

1. Consider the BankAccount example from last week (see Notes/week09/BankAccount.py on the class Web page). Add a too-good-to-be-true method called `resetBalance`, which takes no input parameters (other than `self`) and resets the account's balance to the original amount that it was created with. For example:

```
>>> a = BankAccount(500)
>>> a.withdraw(350)
Withdrew $350.00
>>> a.withdraw(100)
Withdrew $100.00
>>> a.inquiry()
Balance is currently $50.00
>>> a.resetBalance()
Balance is currently $500.00
>>>
```

2. Now let's make BankAccount objects be password-protected. Modify the `__init__` constructor so that it takes an extra password string as input, as in

```
a = BankAccount(500, "open sesame")
```

The account should process a `withdraw` or `inquiry` request only if it is accompanied by the password with which the account was created, and should otherwise complain:

```
>>> a.withdraw(100, "open sesame")
Withdrew $100.00
>>> a.withdraw(100, "abracadabra")
Sorry, that password is incorrect
>>> a.inquiry("stick em up")
Sorry, that password is incorrect
>>>
```

3. Implement a Product class. A product has a name and a price, for example: Product("Toaster", 29.95). Include a method `__str__` that returns the name and price of the product as a formatted string, a method `priceWithTax(percent)`, which returns the price with sales tax of the given percentage added on, and a method `setNewPrice(amount)`, which changes the product's price to the new amount and prints out a message. Your objects should behave as follows:

```
>>> gizmo = Product("Ronco Tomato Scrambler", 39.95)
>>> print gizmo
Ronco Tomato Scrambler, price: $39.95 plus tax
>>> gizmo.priceWithTax(10)    (10% tax)
43.945
>>> gizmo.setNewPrice(49.95)
Ronco Tomato Scrambler now costs $49.95
>>> gizmo.priceWithTax(10)
54.945
>>>
```

(continued on back)

4. Implement an Auditorium class. An auditorium has a seating capacity that is specified when the auditorium is created. Once this capacity is reached, no more seats can be filled. An Auditorium object should keep track of its seating capacity, and the number of seats that are currently open. Write and test the following methods:

- `seatsAvailable()` returns the current number of empty seats.
- `seatsOccupied()` returns the current number of occupied seats.
- `fillSeats(numRequested)` attempts to fill up to `numRequested` seats. If `numRequested` is greater than the number of currently available seats, all available seats are filled and a message reporting the number of requests that could be accommodated is printed.
- `lookInside()` prints out the current number of filled and unfilled seats.

```
>>> carnegieHall = Auditorium(1500)
>>> carnegieHall.lookInside()
No one is inside
>>> carnegieHall.fillSeats(1000)
Filled 1000 seats
>>> carnegieHall.lookInside()
1000 people inside with 500 seats left
>>> carnegieHall.fillSeats(700)
Sorry, sold out after filling 500 seats
>>> carnegieHall.lookInside()
1500 people inside with 0 seats left
>>> carnegieHall.fillSeats(100)
Sorry, all sold out
>>>
```

5. Implement a Vehicle class that, like the Auditorium class above, keeps track of the number of people inside and the total number of seats available. However, a Vehicle should also remember the names of each person inside. Write and test the following methods:

- `seatsAvailable()` returns the current number of empty seats.
- `seatsOccupied()` returns the current number of occupied seats.
- `addPerson(name)` fills one more seat with a person of the specified name. If there are no more seats available, a message reporting this fact should be printed.
- `removePerson(name)` removes a person from the Vehicle. If there is no person inside with the specified name, an error message is printed. If more than one person with the same name is inside, only one of them is removed.
- `lookInside()` prints out the names of all of the people currently inside the Vehicle.

```
>>> chevy = Vehicle(4)
>>> chevy.addPerson("Thelma")
>>> chevy.addPerson("Louise")
>>> chevy.lookInside()
Thelma
Louise
2 empty seats left
>>>
```