Lab 7

Due: Friday, March 31, at 11:59 PM. Submit your completed "lab7.py" file through Blackboard (multiple submissions are permitted; we will only grade the last/latest submission). Submissions that do not execute due to syntax or other errors will receive a 0.

Previously in lecture, we discussed Python classes, which allow us to define our own data types that combine data (instance variables) and operations (instance methods). In this lab, we will use classes to create a virtual aquarium, and then manipulate it using a simple sorting algorithm known as "bubble sort".

1. Start by defining a new `Fish` class, which will represent a single fish in our aquarium. A `Fish` has four instance variables (its name, its species, its color, and its weight in ounces). Every `Fish` also has the following methods (with the following headers):

    a. `__init__(self, name, species, color, weight)`

    This method creates four new instance variables for the `Fish`, and assigns them the values of the last four method parameters. Remember that an instance variable's name is always preceded by "self.", as in `self.myData`.

    b. `__repr__(self)`

    This method returns a string representing the current `Fish`. This string should include a crude fish symbol (<><) followed by the `Fish`'s name, color, and species in parentheses. For example:

    ```
    <>< (Topper, blue Guppy)
    ```

    c. `__lt__(self, other)`

    This method returns `True` if the current `Fish`'s weight is less than the other `Fish`'s weight instance variable, and `False` otherwise. Remember that Python does not protect instance variables from outside access, so you can directly refer to, for example, `other.weight`.

    d. `kind(self)`

    This method returns the species of the current `Fish`.

2. Next, we need an aquarium to hold our fish. To do this, we need to define a second class, named `Aquarium`. An `Aquarium` has two instance variables: a list of `Fish` instances, and an integer representing the water capacity of the tank in gallons (each `Fish` requires 1 gallon of water). An `Aquarium` also has the following methods:

    a. `__init__(self, capacity)`

    This method creates a new instance variable that is an empty list; this list will eventually be filled with `Fish` objects. It also assigns its second parameter to a new instance variable representing the `Aquarium`'s capacity in gallons.

b. `__repr__(self)`

This method returns a new string that contains the string representations of the `Fish` in the `Aquarium`'s internal list, in order from index 0 to index (len-1). Insert tab('\t') and newline ('\n') characters to ensure that, when this string is displayed, no more than two `Fish` are printed per line (if there are an odd number of fish, the last one will be printed on a line by itself). For example, a call to this method might return a string like `'<>< (Bill, red Gourami)\t<>< (Ted, yellow Goldfish)\n<>< (Mary, pink Mollie)'`.

Note that you can call `str()` on a `Fish` (or any other) object to automatically invoke its `__repr__()` method.

c. `add(self, newFish)`

If the number of `Fish` in the `Aquarium` is less than its capacity, this method adds the new fish to the end of the list of `Fish` and returns `True`. Otherwise, it simply returns `False` (without adding the new fish).

d. `population(self)`

This method returns the number of `Fish` that are stored in the `Aquarium`'s list.

e. `countType(self, type)`

This method returns the number of `Fish` in the `Aquarium` whose species matches `type`. For example, `myTank.countType('Guppy')` would return the number of `Fish` in `myTank` that have 'Guppy' as their species.

3. Add some driver code to your "lab7.py" file to test your two classes:

   a. Read in a tank capacity and the name of a data file from the user.

   b. Create a new `Aquarium` object with the specified capacity

   c. Read the data file. Each line of the data file has the following format:

   species name color weight

   where each field is separated by a single space (note that the order of the fields in this data file format is slightly different from the order of the parameters in the `Fish` constructor). For each line of the data file, create a new `Fish` object and add it to your `Aquarium`. If you are unable to add the current `Fish`, print out a message to that effect.

d. Print out the number of fish that are currently in the aquarium.

e. Print out the aquarium's contents.

f. Get a type of fish from the user, and print the number of fish of that type in the aquarium.

4. Finally, we will add a new method to our `Aquarium` class to sort its fish in ascending order by weight. We will do this using a sorting algorithm called "bubble sort". Bubble sort is very simple, but somewhat inefficient (this is okay; we don't have that many fish to sort).

   Bubble sort works by examining pairs of adjacent elements in a list (the first and second, the second and third, the third and fourth, the fourth and fifth, etc.). If a pair of elements is "out of order" (meaning the second one is less than the first one), we swap them before continuing on. Each pass moves the largest unsorted element to its proper place at the end of the list. In order to completely sort a list of N items with bubble sort, we need N-1 passes:

   ```
   for r in range(number of elements - 1):
      for index in range(number of elements - 1):
         if element at index+1 is less than element at index:
            swap element at index+1 with element at index
   ```

   Add a new method to your `Aquarium` class with the following header:

   ```
   bubblesort(self)
   ```

   This method should sort the fish in the aquarium (by weight) using bubble sort. At the end of each round of the algorithm, print out the current list of fish. Thus, it should be possible to track the progress of the sort as different fish are rearranged.

   Once you have finished implementing bubble sort for your `Aquarium` class, modify your main program code to call that method (which will print out the sorted tank along the way).

Sample Program Output

(user input is in **bold**; program output is in *italics*)

```
How many gallons of water can the tank hold? 10
Enter the name of the file containing the list of fish: tank1.txt
Could not add Krusty to the tank!
Could not add Bob to the tank!

Current tank population: 10 fish

<>< (Alvin, orange Tetra)     <>< (Simon, gray Guppy)
<>< (Theodore, blue Oscar)    <>< (Bart, yellow Cichlid)
<>< (Lisa, green Barb)        <>< (Homer, yellow Goldfish)
<>< (Marge, red Gourami)      <>< (Maggie, pink Mollie)
```

```
<>< (Ned, gray Cichlid)       <>< (Maude, orange Barb)

Enter a type of fish to count: Cichlid
You have 2 fish of that type.

Organizing the fish...

Round 1
<>< (Alvin, orange Tetra)     <>< (Theodore, blue Oscar)
<>< (Bart, yellow Cichlid)    <>< (Simon, gray Guppy)
<>< (Lisa, green Barb)        <>< (Marge, red Gourami)
<>< (Maggie, pink Mollie)     <>< (Ned, gray Cichlid)
<>< (Maude, orange Barb)      <>< (Homer, yellow Goldfish)


Round 2
<>< (Theodore, blue Oscar)    <>< (Alvin, orange Tetra)
<>< (Bart, yellow Cichlid)    <>< (Simon, gray Guppy)
<>< (Marge, red Gourami)      <>< (Maggie, pink Mollie)
<>< (Ned, gray Cichlid)       <>< (Maude, orange Barb)
<>< (Lisa, green Barb)        <>< (Homer, yellow Goldfish)


Round 3
<>< (Theodore, blue Oscar)    <>< (Alvin, orange Tetra)
<>< (Bart, yellow Cichlid)    <>< (Marge, red Gourami)
<>< (Maggie, pink Mollie)     <>< (Simon, gray Guppy)
<>< (Ned, gray Cichlid)       <>< (Maude, orange Barb)
<>< (Lisa, green Barb)        <>< (Homer, yellow Goldfish)


Round 4
<>< (Theodore, blue Oscar)    <>< (Alvin, orange Tetra)
<>< (Marge, red Gourami)      <>< (Maggie, pink Mollie)
<>< (Bart, yellow Cichlid)    <>< (Simon, gray Guppy)
<>< (Ned, gray Cichlid)       <>< (Maude, orange Barb)
<>< (Lisa, green Barb)        <>< (Homer, yellow Goldfish)


Round 5
<>< (Theodore, blue Oscar)    <>< (Marge, red Gourami)
<>< (Maggie, pink Mollie)     <>< (Alvin, orange Tetra)
<>< (Bart, yellow Cichlid)    <>< (Simon, gray Guppy)
<>< (Ned, gray Cichlid)       <>< (Maude, orange Barb)
<>< (Lisa, green Barb)        <>< (Homer, yellow Goldfish)


Round 6
<>< (Theodore, blue Oscar)    <>< (Maggie, pink Mollie)
<>< (Marge, red Gourami)   <>< (Alvin, orange Tetra)
<>< (Bart, yellow Cichlid) <>< (Simon, gray Guppy)
<>< (Ned, gray Cichlid)    <>< (Maude, orange Barb)
<>< (Lisa, green Barb)     <>< (Homer, yellow Goldfish)


Round 7
<>< (Theodore, blue Oscar) <>< (Maggie, pink Mollie)
<>< (Marge, red Gourami)   <>< (Alvin, orange Tetra)
<>< (Bart, yellow Cichlid) <>< (Simon, gray Guppy)
<>< (Ned, gray Cichlid)    <>< (Maude, orange Barb)
<>< (Lisa, green Barb)     <>< (Homer, yellow Goldfish)


Round 8
<>< (Theodore, blue Oscar) <>< (Maggie, pink Mollie)
<>< (Marge, red Gourami)   <>< (Alvin, orange Tetra)
<>< (Bart, yellow Cichlid) <>< (Simon, gray Guppy)
<>< (Ned, gray Cichlid)    <>< (Maude, orange Barb)
<>< (Lisa, green Barb)     <>< (Homer, yellow Goldfish)


Round 9
<>< (Theodore, blue Oscar) <>< (Maggie, pink Mollie)
<>< (Marge, red Gourami)   <>< (Alvin, orange Tetra)
<>< (Bart, yellow Cichlid) <>< (Simon, gray Guppy)
<>< (Ned, gray Cichlid)    <>< (Maude, orange Barb)
<>< (Lisa, green Barb)     <>< (Homer, yellow Goldfish)
```

Grading Breakdown

This lab is worth a total of 10 points, broken down as follows:

Point Value	Grading Criterion
2 points	Program correctly adds fish from a user-specified data file into the aquarium, up to its capacity
1 point	Program prints an error message for each fish that exceeds the aquarium's capacity
1 point	Program correctly prints the number of fish in the aquarium
2 points	Program correctly prints the contents of the aquarium
1 point	Program correctly counts the number of a user-specified type of fish in the aquarium
2 points	Program correctly uses bubble sort to organize the fish in the aquarium
1 point	Program prints out the result of each round of the bubble sort algorithm