METRO STATE UNIVERSITY

ICS 141 - 02: Problem solving with programming
Spring 2023
Part 1

Assignment 5: Implementing a collection class of `Things`

Out: Wednesday, March, 29, 2023
Due: April, March, 12, 2023 @ 11:59 PM

Total points: 25

In this assignment, you will implement a collection class to manage a group of things of your choice.

Requirements

Part A: Implement your `Thing` class

In this assignment, you will implement a `Thing` class. Your `Thing` class to meet the following requirements:

- You thing must have exactly 3 instance variables such that: `o` one instance variable must be integer (i.e, `int`) and this variable will be used to find aggregate information (e.g., total or max) about `Things` that are stored in a collection class as will be explained later.
    - one instance variable must be `String` and this variable will be used as a categorization attribute that will be used to divide your collection into groups. You are going to use this attribute to count. For example, for a collection of cars, then the cars can be categorized based on `color` or `brand`.
    - The third instance variable can be either `int` or `String` type based on the semantics of your `Thing`.
- All instance variables must be **private.**
- Implement getters and setters for all the instance variables of your `Thing`.
- Implement `toString` method that returns a `String` representation of your `Thing` class where all the instance variables are in one line and separated by tabs.
- Implement the `equals` method for your `Thing` where two things are considered equal when they have the same values in the three instance variables. Note that, the equality of String attributes should be case insensitive. For example, “MATH”, “math” and “Math” match each other. In order to compare strings in Java use the String's `equalsIgnoreCase` method. For example, the following code prints `true`:

```
String str1 = "Hello";
```

```
String str2 = "hello";

System.out.println(str1.equalsIgnoreCase(str2));
```

Part B: Implement a `Collection` class

- Implement a collection class of `Thing`s using an array.
    1. You may NOT use any of the collection classes in Java library like `ArrayList` or `Vector` or any other java collection class.
    2. You **must use** simple java Arrays as we discussed in class during the BookCollection demo.
- The name of your collection class should include the name of your Thing. For example, if your `Thing` is called `Country`, then your collection class should be called `CountryCollection`.

- Your collection class should include three instance variables as follows:
    1. `name`: a String that represents a given name to a collection.
    2. `numItems`: an int that represents the current number of items in the collection.
    3. `items`: `Thing[]` which is an array that stores references to `Thing` objects.

- Implement a constructor for your collection class that takes two input parameters that represent (1) the collection name, and the maximum number of elements that can be stored in your collection.

- Implement the following methods in your collection class:

    1. `add`: a method that takes one input parameter of type `Thing`. The method adds the input thing to the collection.
    2. `add:` overload the add method by implementing another `add` method that takes three input parameters and the method uses these inputs to instantiate an object of type `Thing` and adds it to the collection.
    3. `size:` a method that returns the number of objects in the collection.
    4. `toString:` a method that returns a String representation of the collection that includes all elements that are stored in the collection. The output string must be nicely formatted in a tabular format. For example, a list of students is to be displayed as follows:

```
name        grade       major
---------------------------
John        20          CS
Eric        25          CIT
Hanna       30    Math Reem
      27    CS
```

5. `total:` this method uses the `int` instance variable of the `Thing` class. Basically, the method calculates and returns the sum of the `int` instance variable of all objects stored in the collection. For example, in the student collection, this method finds the sum of `age` of all students in the list.
6. `greatest:` this method returns the object with the maximum value in the `int` attribute.
7. `countCategory:` this method takes one input parameter of type String that represent one category of things in your collection (e.g., the input is 'red' in a collection of Cars with 'color' attribute). The method counts and returns the number of objects in the collection that falls in this category (e.g., the number of 'red' cars). Note that String comparison must be case insensitive.
8. `contains:` this method takes one input parameter of type Thing. The method searches for the input thing in the collection and returns true or false based on whether the thing is found or no. Note, that you have to use the `equals` method to compare the equality of classes.
9. `countOccurances:` this method takes one input parameter of type Thing and returns as output how many copies of the input thing exist in the collection. Note that things must be compares using the `equals` method.

Part C: Implement your Driver class

Implement a `Driver` class that works as follows:

- creates a collection class that can hold up to 10 things.
- uses the `add` methods to insert 5 items in the collection. You may use any arbitrary values in in the objects to be inserted.
- The driver then calls the methods 3 through 9 in order. Note that you do not have to use a `Scanner` to read any inputs from the user. On the other hand, use hard coded values for the different methods' inputs.

Grading

Your grade in this assignment is based on the following:

- Your submission meets specifications as described above.
- **Add appropriate comments to your code.**
- Variable names should convey meaning.
- At the top of each Java file, include your name, a brief description of the program and what it does and the due date.
- All code blocks must be indented consistently and correctly. Blocks are delimited by opening and closing curly braces. Opening and closing curly braces must be aligned consistently.
- You must use the exact same name (including upper case and lower-case letter) for all methods as specified in the above description.
- The output of our program must be nicely formatted.

- You must follow the method requirements in terms of the number and data type of input parameters and the output data type.
- The program is robust with no runtime errors or problems.
- The programs should display your name.

Part 2 Instructions

- Zip up Part1 and name it i.e. DillonArraysOfObjectCollectionPart1, but Do Not submit it. Continue to Part2.
- Part2 is going to build on Part 1.
- After completion of Part2 and testing all your methods in the ThingDriverCollection, follow the steps in Part2 to submit your project.
- Part1 and Part 2 will be in separate .zip folders, but the final submission will in one project folder.
- If you have questions, please email me way before the deadline. **Do Not** email 1 or 2 fays before the due date. Email me in advance so I can have ample time to respond.