Metro State University

ICS 240: Introduction to Data Structures

Fall 2024

Programming Assignment 1: Arrays of Objects

Total points: 35

Out: Tuesday September 3rd

Due: Tuesday September 17th

Last day to submit: Wednesday September 18th[[1]](#footnote-1)

In this assignment, you will practice writing Java code using files, Objects, and arrays.

# Requirements

## Part 1: **Item** class

Start by choosing one sort of Item that is of interest to you. (For example: vacation or travel information, real estate property, or restaurants. But don’t feel limited to these ideas!) Do not use any of the following as your Item: Person, Car, Student, Book, BankAccount, Movie, Pet, Animal, Song, or Videogame.[[2]](#footnote-2)

***Name***: Give your **Item** class a meaningful name. Do not name your class **Thing** or **Item**.

***Instance Variables***: Give your **Item** class exactly two private instance variables, a **String** and an **Integer**. Give them meaningful names that describe the characteristics of your Item.

***Some methods***:

* **public Item(** [**String**](https://docs.oracle.com/en/java/javase/22/docs/api/java.base/java/lang/String.html) **s,** [**Integer**](https://docs.oracle.com/en/java/javase/22/docs/api/java.base/java/lang/Integer.html) **n )**

Constructs an **Item** object that that initializes the two instance variables. The order of the input parameters must be **String** then **Integer**.

* Implement getters and setters for all the instance variables of your **Item**.
* **public void toString()**

Returns a **String** representation of your **Item** where the instance variables are on one line and tab-separated. Do not add any text to the output of this method other than the values of the two instance variables.

* **public boolean equals(Item o)**

compares your Item to the parameter **o** for equality. The equality of string attributes should be case insensitive. For example, "MATH", "math" and "Math" are equal. Note: to compare **String** objects in Java use the **boolean equalsIgnoreCase(****[String](https://docs.oracle.com/en/java/javase/22/docs/api/java.base/java/lang/String.html) s)** method from the **String** class. For example, the following code prints true:

**String str1 = "Hello";**

**String str2 = "hello";**

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

## Part 2: Upload data from a text file to an array of **Item** objects

* Create a **.txt** file that includes data about a collection of items. Each line in the file represents one item where the string and integer separated by tab (e.g., **abc 10**). Add at least 20 lines to your text file. Make sure to repeat some of the string values in the file.
* Create another class in your project, called **<Item>ArrayDriver**. Use the name of your item instead of **<Item>**.
* In the main method, declare an array, called **<Item>Arr**, that can hold up to the number of items in your text file.
* In the main method, ask the user to enter file name then read the user input. Create a [**File**](https://docs.oracle.com/en/java/javase/22/docs/api/java.base/java/io/File.html) object using the input file name. Create a [**Scanner**](https://docs.oracle.com/en/java/javase/22/docs/api/java.base/java/util/Scanner.html) object to read from the file.
* Write a loop to read the text file. You need to read one field from the file at a time. Basically, for each line in the text file, you will read the two fields in each row using the following **Scanner** methods: **next()** then **nextInt()**.
* Use the values read from one line from the file to instantiate an **Item** object then add that object to **<Item>Arr**.
* After uploading the data to the array, write a 'for' loop to print on the screen all objects in **<Item>Arr** to make sure that the data to read correctly.

## Part 3: Methods to process **Item** array

Implement the following five methods in **<Item>ArrayDriver**. Test the methods from the main method using **itemArr** as input. Remember: string comparison is case insensitive.

* **public int overallAvg( Item[] a )**

Returns the average (arithmetic mean) of the **Integer** attributes of all items in input array **a**.

* **public int count( Item[] a, Item o )**

Returns the number of **Item**s in the input array **a** that are equal to the input **Item o**. Remember, two **Item**s are equal if they have the same value for both instance variables, where **String** variables are equal if they differ only in case.

* **public int groupAvg(Item[] a, s)**

Returns the average of the **Integer** attributes of **Items** with **String** attribute equal to the input string. For example, **groupAvg(itemArr,"abc")** returns the average of the **Integer** attributes for all objects with **String** attribute equal to **abc**.

* **public Item[] lessThan( Item[] a,** **[Integer](https://docs.oracle.com/en/java/javase/22/docs/api/java.base/java/lang/Integer.html) i )**

Returns an array of **Item** objects that contains all objects with Integer attribute less than the input **Integer**. For example, **lessThan(itemArr, 50)** returns an array of items with **Integer** attribute < 50.

* **public void groupEdit( Item[] a,** [**String**](https://docs.oracle.com/en/java/javase/22/docs/api/java.base/java/lang/String.html) **s,** [**Integer**](https://docs.oracle.com/en/java/javase/22/docs/api/java.base/java/lang/Integer.html) **n )**

For each item in the input array with **String** attribute equal to the input string, the method edits the **Item**’s **Integer** attribute by adding **n** to the **Integer** value. Thus, **groupEdit(itemArr,"abc",10)** adds 10 to the **Integer** values of each **item** in **itemArr** with **String** value **abc**.

# Grading

Your grade in this assignment is based on the following criteria:

* Your submission meets specifications as described above.
* You must use the exact same name (case matched) for all fields and methods as specified in the above description.
* The method input parameters must be in the same order as specified.
* The program is robust with no runtime errors or problems.
* You follow good programming style.

# Submission Instructions

* Using your IDE (Eclipse, Visual Studio, IntelliJ, etc), create a java project and call it **<yourLastName>Assignment1**.
* Create .java files and a text file to implement the classes as described above.
* Archive the three files into one zip file.
* Upload only one .zip file to the D2L folder called Assignment 1.
* It is important that you upload your code in only one zip file. Your assignment will not be graded if you upload individual files to the drop box**.**
* You may submit more than once, but only the last submission (that is, the single .zip file with the most recent timestamp) will be graded.

1. With 10% point deduction [↑](#footnote-ref-1)
2. Note: In the next assignment, you will divide this Item into two subclasses of items. So pick an Item that can be divided easily. For instance, to use a prohibited example, Pet might be subdivided into Dog and Cat. It will be OK if not all Items belong to one of the subclasses. (Maybe you have a bird named “Polly”.) Additionally, you will be asked to add an Integer instance variable to one of the subclasses and a Boolean instance variable to the other. So plan ahead. [↑](#footnote-ref-2)