CS: 114 **(OPPDA)** **Fall, 2025**

Lab 2 (50 points) **Due 3/3/2025**

Submit your solutions to canvas. For programming assignments do not send the entire project. All that is needed are the files ending in .java. Each class will have to be defined in its own separate .java file. All driver code should be put in one .java file. Please make sure your name is included at the top of each .java file.

There will be no re-submission of work. There will be no exceptions. So, before you submit your work, please ensure you have everything the way you want it.

Goal: Practice using the Java Map data structure. Also, practice the enhanced for-loop for a Java `Map.` This lab is very similar to Project3 problem 3. Only this lab uses maps as opposed to sets. Please refer to the lecture6 slides and examples.

Problem 1

Using the `PersonP3` class that you created in Project3 the first step is to create a map like this:

```
Map<Integer, PersonL2> map = new HashMap<>();
```

Your `PersonL2` class should contain a private data member declared like this:

```
List<String> transActionList;
```

Along with two other data private members, `id` and `name`:

```
int          id;
String       name
List<String> transActionList;
```

Your `PersonL2` constructor should use `new` to create a `List` object of type `ArrayList` with an initial capacity of 100. It is not necessary to create setters or getters for `transActionList.` You should implement a public `void` method that adds a transaction `String` to `transActionList.` You should also have a `public void` method that accepts a `FileWriter` object as a parameter and uses it to display each transaction `String` from `transActionList` on a separate line in the output file. You might name the method `outputTransactionsToFile`. For testing purposes, you might also have a method that writes all the transactions to the console window each on a separate line.

Use the file browser to read in a file that contains transaction records. Each record in the file contains `id` as an integer, `name` as a `String`, and `transaction` as a `String`. Once you read in these three items construct a `PersonL2` object like so:

```
PersonL2 p = new PersonP3(id, name);
```

The constructor would create an empty `ArrayList` of `String` to hold the transactions with an initial capacity of 100.

Next, use `get` with argument of `id` to search for a `PersonL2` reference variable `pF`.

```
if pF is not null then
{
   Add the transaction to pF
}
else
{
   add the transaction to p
   put p in the map using id as the key, and p as the value.
}
```

There are no loops required in the above if-else statement.
Use an enhanced for-loop to display each `PersonL2` in the map along with each person's transaction list.

Test your code and make sure your code works with all of the below maps:

```
Map<Integer, PersonP3> map = new HashMap<>(); // Unordered

// Order based on insertion order
Map<Integer, PersonP3> map = new LinkedHashMap<>();

// Ordered Map sorted order of key
Map<Integer, PersonP3> map = new TreeMap<>();
```

Output from your program should look like this:

```
The selected file is
\\rowanads.rowan.edu\home\rabbitz\Documents\Bank1.txt
Reading the bank transaction file
100 Jim DEP
200 Ron WIT
300 Harry BAL
100 Jim WIT
100 Jim VIS
200 Ron DEP
300 Harry DEP
400 Kim DEP
400 Kim DEP
400 Kim VIS
```

```
100 JIM WIT
200 RON WIT


Display of customer transaction log
id = 400 name = Kim
There were 3 transactions.
DEP
DEP
VIS
id = 100 name = Jim
There were 4 transactions.
DEP
WIT
VIS
WIT
id = 200 name = Ron
There were 3 transactions.
WIT
DEP
WIT
id = 300 name = Harry
There were 2 transactions.
BAL
DEP
```

Your program should write the customer transaction log to an output file.