Coding Assignment 4

CSE1325 Fall 2021

1. Create `Code4_xxxxxxxxxx.java`

Copy your working version of `Code3_xxxxxxxxxx.java` to `Code4_xxxxxxxxxx.java`. Create a new version of `CokeMachine.java` to go with your `Code4_xxxxxxxxxx.java`. Make sure `Code4_xxxxxxxxx.java` works just like `Code3_xxxxxxxxxx.java` before continuing with the assignment.

2. Add two new instance methods – `setMachineName()` and `setCokePrice()`

Add a new instance method called `setMachineName()`. Add it as menu item 5, "`Update Machine Name`". Prompt the user "`Enter a new machine name`". Take in `newMachineName` and call `setMachineName()` with one `String` parameter of `newMachineName` and no return value. After calling the method from `Code4_xxxxxxxxxx.java`, print "`Machine name has been updated`".

Add a new instance method called `setCokePrice()`. Add it as menu item 6, "`Update Coke price `". Prompt the user "`Enter a new Coke price`". Take in `newCokePrice` and call `setCokePrice()` with one `int` parameter of `newCokePrice` and no return value. After calling the method from `Code4_xxxxxxxxxx.java`, print "`Coke price has been updated`".

3. Add another `CokeMachine` constructor

Add another constructor to `CokeMachine` that has no parameters and creates a `CokeMachine` using the following values.

`machineName` of "`New Machine`", Coke price of 50, change level of 500 and inventory level of 100

4. Command line parameters

Set up and use two command line parameters

```
INPUTFILENAME=xxxx OUTPUTFILENAME=yyyy
```

If `args.length` is not 2, then print a message of

```
"Missing command line parameters  -  - Usage : INPUTFILENAME= OUTPUTFILENAME="
```

and then use `System.exit()` to end the program.

If `args.length` is 2, fill in variable `inputFileName` with `xxxx` and variable `outputFileName` with `yyyy` where `xxxxx` and `yyyyy` are files of any name. Do not add extensions or anything else to the filenames. DO NOT HARDCODE THE LOCATION OF THE = IN THE COMMAND LINE PARAMETERS OR THE NAMES OF THE PARAMETERS.

5. `FileRead` method

Copy your file reading method from Coding Assignment 2 into `Code4_xxxxxxxxxx.java`. Pass in the input file name and the `ArrayList` of Coke Machines. Open the file and use a `while` to read all lines in the file. If the file does not open, print a message "`xxxx file name does not exist...exiting`" and exit. Inside the `while`, use `split` with | to parse each line from the file. The array resulting from `split()` will have 4 elements. Each element of the array is a value that you will use to construct a `CokeMachine`. You are adding the instantiated `CokeMachine` to the `ArrayList` of `CokeMachine` you passed into the method. We did this in lecture with `Zoo`.

```
Create an array of 4 Strings
while hasNextLine()
   array = read next line from file and call split with |
   YourArrayList.add(new CokeMachine(X[0], X[1], X[2], X[3])
```

This pseudocode is JUST meant to give you the idea – several elements are missing and you need to fill them in (for example, your array is `String` and 3 of the 4 parameters to your `CokeMachine` constructor are `int` so you will need to convert).

The format of each line of the file is

```
machine name|Coke Price|change level|inventory level

Machine Bugs Bunny|50|500|50
Machine Cecil Turtle|45|545|45
Machine Daffy Duck|40|540|1
Machine Elmer Fudd|100|1000|10
Machine Fog Horn|35|350|99
```

5. Machine menu

Create a new menu method to display the list of machines. Pass in the `ArrayList` of `CokeMachine`s. Copy most of the code from your Coke menu method. Instead of a hardcoded menu list, replace that code with a for loop to loop over the `ArrayList` and print each machine's name. See output. Ask the user for the choice, validate the choice, use the `try-catch` to process bad input and return the choice.

REMEMBER – the list of machines can be any length – you should not hardcode anything related to the number of items that you will need in the menu. Use the size of the `ArrayList` instead of a hardcoded value.

Menu option "`Add new machine`" will always be the last menu option.

6. Create a function to write `SetOfCokeMachines` to the output file

Pass in the output filename and `SetOfCokeMachines`. Open the output filename for writing (copy your code from Coding Assignment 2). Add a `try-catch` for `FileNotFoundException`. In the `catch`, print "`Unable to write output file`" and then print the exception object and then `throw` the exception object. You will be required to add "`throws FileNotFoundException`" to your method and to `main()`. We are just catching the exception long enough to produce an error message, but then throwing it back at the JVM so it will blow up and stop the program.

If you do not have a try-catch or a throws clause in this function, you will get

```
error: unreported exception FileNotFoundException; must be caught or declared to be thrown
```

on the new `PrintWriter`. The exercise here is to be able to add a try-catch to catch the error and throw it again; therefore, for this assignment, you are being asked to add **BOTH** a try-catch and a throws clause. Since `PrintWriter` does not throw an exception when a file does not exist, you will not be able to test the code by not having a file (the file will just be created). To test, write to a file that already exists AFTER changing the properties of the file to be read only (in Windows, go to File Explorer->Properties and check the Read-Only box under Attributes). Trying to write to a read only file will cause the exception to be thrown

Use an enhanced `for` loop to print each `CokeMachine`'s information to the output file. Be sure to follow the same format as the input file. Your output file should be able to be used as an input file.

7. `Code4_xxxxxxxxxx.java`

`main()` should take get the input filename and the output filename from the command line parameters.

Create an `ArrayList` of type `CokeMachine` called `SetOfCokeMachines`.

Call the method to open the input file and construct your `ArrayList` of `CokeMachine`s.

Get the machine choice by calling the machine menu.

while the chosen machine is not 0

    if add a new machine option was chosen

        add a new `CokeMachine` to `SetOfCokeMachines` using the default constructor

    else

        Put a copy of the chosen machine in the `CokeMachine` variable you declared at the start of `main()`

        do-while code that runs one one machine from the previous assignment

    Get the machine choice by calling the machine menu.

Once the user chooses to exit the machine menu, write the machines to the output file.

10% BONUS

If the program is run and NO Cokes are sold, then the message "**`0 Coke(s) were sold today!`**" is printed.

```
Pick a Coke Machine


0. Exit
1. Machine Babs Bunny
2. Machine Cecil Turtle
3. Machine Daffy Duck
4. Machine Elmer Fudd
5. Machine FogHorn
6. Machine Sylvester
7. Marvin the Martian
8. RoadRunner
9. Add new machine

Choice? 0
0 Coke(s) were sold today!
```

If the program is run and Cokes are sold, then the message "**`x Coke(s) were sold today!`**" is printed where `x` is the number of all Cokes across all machines.

```
3 Coke(s) were sold today!
```

Determine how to print this message.

This line of code

```
System.out.printf("%d Coke(s) were sold today!\n", xxxxx);
```

should be the last line of the program and must get the number of Cokes sold **WITHOUT** doing ANY of the following

- using a `CokeMachine` object to get the number of Cokes sold
- using a new method or adding a new method
- making any instance variables public
- saving the number of Cokes in another variable

HINT : You will need to add 1 word to one of the instance methods in `CokeMachine`. Consider how we use (for example) `Integer.parseInt()` or `String.valueOf()` or `Thread.sleep()` or `Math.floor()` or `Math.pow()`.