Coding Assignment 2

CSE 1325

Coding Assignment 2 is a continuation of Coding Assignment 1. Please copy your code from Coding Assignment 1 into a Coding Assignment 2 project. Create your Coding Assignment 2 project and then copy your main and your functions from Coding Assignment 1 into your new Coding Assignment 2 project.

One of the rubric criterions will be for the grader to go back to your Coding Assignment 1 rubric and verify that you have corrected anything you lost points for in that assignment. If you lost points on Coding Assignment 1, make sure you fix those problems in Coding Assignment 2 or you will lose a significant number of points on your new assignment.

Add the following new features to your pencil machine.

Changes in `main()`

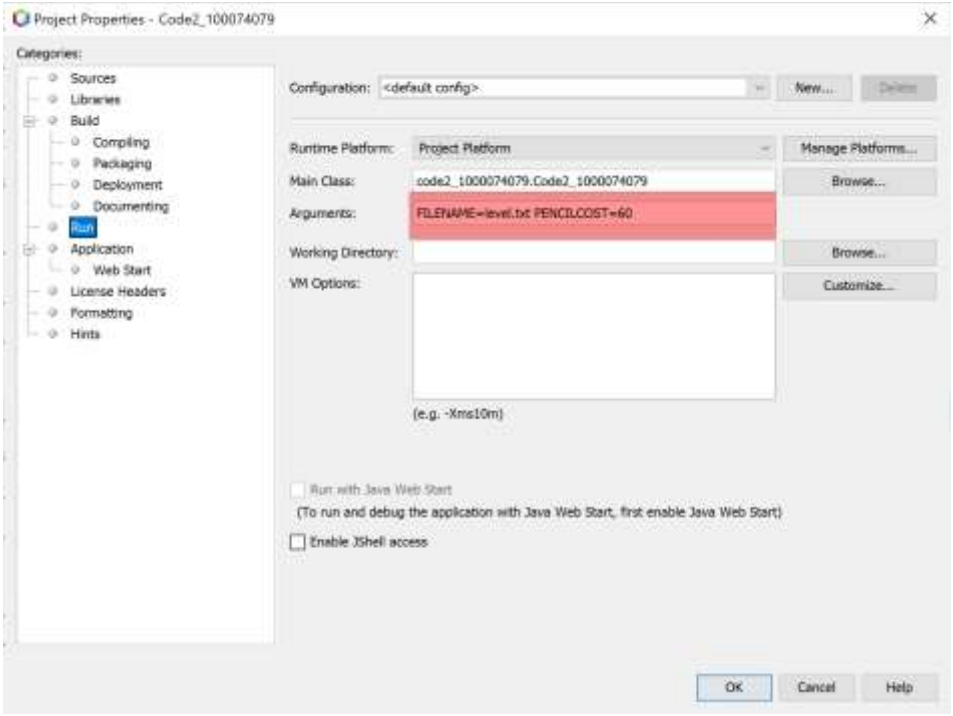
Declare an `ArrayList` of type `String` to hold the pencil colors you will be reading from a file. You should already have an integer array that you are using to hold change and inventory levels.

Command line arguments

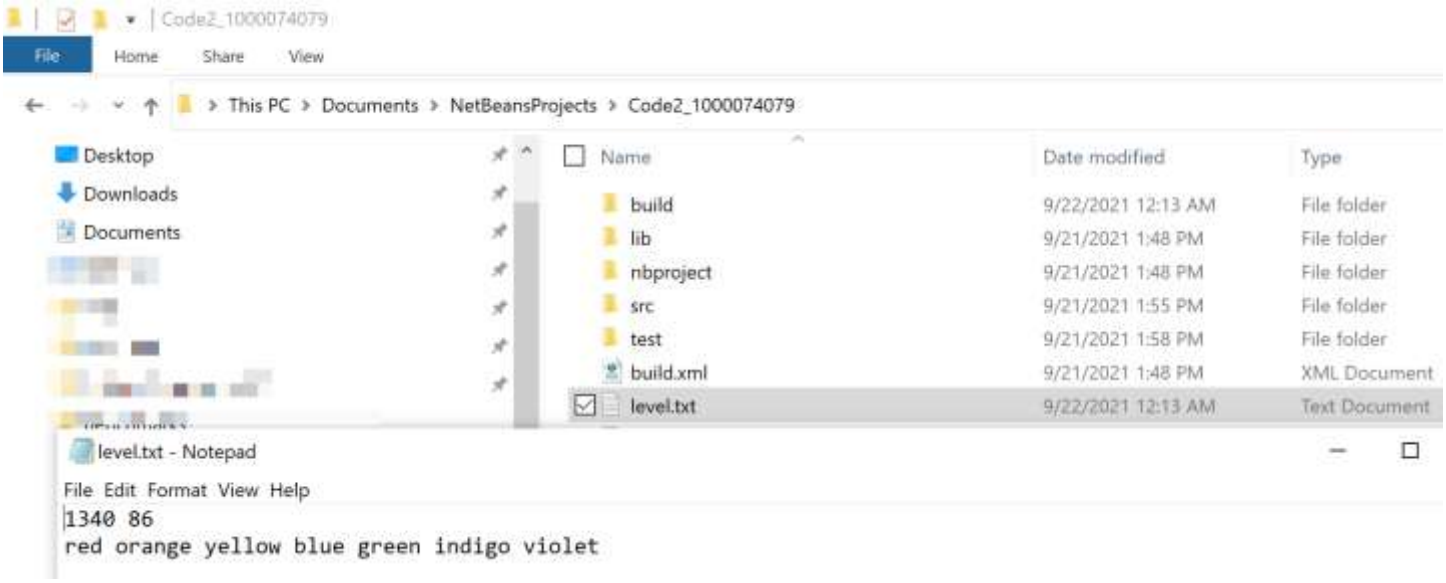
Set your IDE to take in the following command line arguments

```
FILENAME=XXXXX PENCILCOST=YY
```

where `XXXXX` is the name of the file you have created to contain the change level and the inventory level and `YY` is the integer cost of a pencil. In this example, the filename is `level.txt` and the pencil cost is set to `60`.



You need to create the file in your project's folder. The first value in the file is the change level and the second value is the inventory level. You MUST use the same file format as shown here (change level value and then a space and then the inventory level). When your code is graded, this file will be used (you will not be submitting your file).



In `main()`, add code to use `args[0]` and `args[1]`.

`args[0]` will have `FILENAME=XXXX` in it. You will need to parse this string (I suggest using `substring` and `indexOf`) to extract the filename after the = in the string.

`args[1]` will have `PENCILCOST=YY` in it. You will need to parse this string (I suggest using `substring` and `indexOf`) to extract the pencil cost after the = in the string. You will also need to transform this string into an integer (suggest use `Integer.parseInt`).

For example, if the command line arguments are set to

`FILENAME=MyLevelFile.txt PENCILCOST=43`

then you should use and parse `args[0]` and `args[1]` and end up with a filename of `MyLevelFile.txt` and a pencil cost of `43`. You should now have a `String` to hold the filename so you can open and manipulate the file and you should replace your final int pencil price with the parse value from `args[1]`.

Add File Handling

Now that you have parsed the file name from the command line arguments, call a function named `ReadFile`. Pass in the integer array used to hold change level and inventory level and pass in the filename and the `ArrayList` that will hold the colors read from the file. Inside the function, open the file and read the first line into a string and the second line into a string.

Use a combination of `split()` and `parseInt()` on the first line and set change level and inventory level based on the values in the file.

Use `split()` to break the second line up into an array. Use an **enhanced `for` each loop** to write the array into the passed in `ArrayList`.

Close the file before finishing the function.

Before the program ends (which is triggered by picking menu option 0), open the same file for writing and write the new change level and inventory level to the file. You should be able to rerun the program and see that the change level and inventory level start where the previous run left off. Use an **enhanced `for` each loop** to write the `ArrayList` to the file.

Add Exception Handling

In your pencil menu function, add exception handling. Add a `try` block around the `nextInt()` that captures the choice and add a `catch` block that sets choice to an invalid value that will cause the invalid input message to be display and the menu to be redisplayed. Be sure to remove the bad input from `stdin` as discussed in class.

In your read file function, add exception handling around opening the file. If the file cannot be opened, use `System.exit` to end the program after printing a message about not being able to open the file.

Pick a random color

When a pencil is sold, a color should be mentioned in the output to the screen

```
Here's your violet pencils and your change of $0.51

Here's your blue pencils.  Thank you for exact payment.
```

Create a function that picks a random color from the `ArrayList` that was created when the input file was run. This function should return a randomly chosen color as a `String`. The function should be able to pick a random value that is within the size of the `ArrayList` without hardcoding that size. Your program will be tested with a set of colors different from the example given.

Test Cases

Test	Test Case Description	Expected Result
1	0 is entered for menu choice	program completes without any further printing or prompts
2	1 is entered for menu choice 3 pencils are requested for purchase Exact payment is given	Pencil purchase price is properly displayed as 3 times the pencil price. User is thanked for exact payment. Menu is redisplayed.
3	1 is entered for menu choice 3 pencils are requested for purchase Over payment not exceeding available change is given	Pencil purchase price is properly displayed as 3 times the pencil price. User is given the correct change. Menu is redisplayed.
4	1 is entered for menu choice 3 pencils are requested for purchase Over payment exceeding available change is given	Pencil purchase price is properly displayed as 3 times the pencil price. User is informed that the pencil machine does not have enough change and cannot accept the payment. Menu is redisplayed.
5	1 is entered for menu choice 3 pencils are requested for purchase Under payment of greater than 0 is given	Pencil purchase price is properly displayed as 3 times the pencil price. User is informed the payment was insufficient and pencils will not be dispensed. Menu is redisplayed.
6	1 is entered for menu choice 3 pencils are requested for purchase Under payment of 0 is given	Pencil purchase price is properly displayed as 3 times the pencil price. User is informed the payment was insufficient and pencils will not be dispensed. Menu is redisplayed.
7	1 is entered for menu choice 3 pencils are requested for purchase Under payment of less than 0 is given	Pencil purchase price is properly displayed as 3 times the pencil price. User is informed the payment was insufficient and pencils will not be dispensed. Menu is redisplayed.
8	2 is entered for menu choice	Inventory level is properly displayed. Option should be run before and after test case 2 to show proper inventory level decreases
9	3 is entered for menu choice	Change level is properly displayed. Option should be run before and after test case 2 to show proper change level increases.
10	9 is entered for menu choice	Invalid menu option message is displayed and menu is redisplayed.
11	1 is entered for menu choice -3 pencils are requested for purchase	User is told “Cannot sell that quantity of pencils. Please reenter quantity”
12	1 is entered for menu choice 0 pencils are requested for purchase	User is told “Cannot sell that quantity of pencils. Please reenter quantity”
13	1 is entered for menu choice 200 pencils are requested for purchase (inventory should start at 100)	User is told “Cannot sell that quantity of pencils. Please reenter quantity”
14	2 is entered for menu choice to retrieve current inventory level. 1 is entered for menu choice All pencils in inventory are purchased. 1 is entered for menu choice	After purchasing all available pencils, the second run of menu choice 1 should display a message stating that the pencil dispenser is out of pencils.
15	A letter is entered at the menu prompt	Exception is handled and invalid menu option is displayed and menu is redisplayed.
16	1 is entered for menu choice A letter is entered for the number of pencils to purchase	Exception is thrown and program ends
17	1 is entered for menu choice 3 pencils are requested for purchase A letter is entered for the payment.	Exception is thrown and program ends