CSE1322L Assignment 1

Introduction:

US currency is made up of notes and coins. The most common notes are $1, $5, $10, $20, $50 and $100. The most common coins are 1c (pennies), 5c (nickels), 10c (dimes), and 25c (quarters).

In this assignment, you will model US currency with objects, creating objects for each note and coin.

Tasks:

First write a class called Coins that has:

- A private integer called quantityOnHand
- A private float called denomination
- A private float called weight
- A constructor which takes two floats (denomination and weight) and sets the object attributes.
- A method called getTotalWeight() which returns a float representing the total weight of that coin (weight * quantityOnHand)
- A method called getTotalValue() which returns a float representing the total value of this coin (denomination * quantityOnHand)
- A method called increaseQuantity which takes in a quantity (int) and increases the quantityOnHand by that amount.
- A method called decreaseQuantity which takes in a quantity (int) and decreases the quantityOnHand by that amount. Note, you can never have less than 0 of any coin. If you are asked to decreaseQuantity below 0, just set it to 0.
- A method called getQuantityOnHand which takes in no parameters, and returns the value of the quantityOnHand attribute (an int).
- A method called printPretty which takes in a float called amount, and returns a string. The body of the method will be:

  Java:

  ```
  return(“$”+String.format("%4.2f",amount));
  ```

  C#:

  ```
  return(“$”+amount.ToString("F2"));
  ```

- An override of toString (java) or ToString (C#) which returns a string like “$3.25 in $0.25 coins”. The first value should be the total value of the coin, the second value should be the denomination of the coin. Hint printPretty() may come in handy here.

Next write a class called Notes which has:

- A private integer called quantityOnHand
- A private integer called denomination
- A constructor which takes one integer called denomination and sets the object attributes.
- A method called getTotalValue() which returns a int representing the total value of this note (denomination * quantityOnHand)
- A method called increaseQuantity which takes in a quantity (int) and increases the quantityOnHand by that amount.
- A method called decreaseQuantity which takes in a quantity (int) and decreases the quantityOnHand by that amount. Note, you can never have less than 0 of any note. If you are asked to decreaseQuantity below 0, just set it to 0.
- A method called getQuantityOnHand which takes in no parameters, and returns the value of the quantityOnHand attribute (an int).
- A method called printPretty which takes in a float called amount, and returns a string. The body of the method will be:

  Java:

  ```
  return("$"+String.format("%4.2f",amount));
  ```

  C#:

  ```
  return("$"+amount.ToString("F2"));
  ```

- An override of toString (java) or ToString (C#) which returns a string like "$32.00 in $1.00 notes". The first value should be the total value of the note, the second value should be the denomination of this note.

Finally, write a main method. Start it with the following code:

```
Notes twenties=new Notes(20);
Notes tens=new Notes(10);
Notes fives=new Notes(5);
Notes ones=new Notes(1);

Coins quarters=new Coins(0.25f,0.2f);
Coins dimes=new Coins(0.10f,0.08f);
Coins nickels=new Coins(0.05f,0.176f);
Coins pennies=new Coins(0.01f,0.088f);

dimes.increaseQuantity(41);
nickels.increaseQuantity(17);
pennies.increaseQuantity(132);
ones.increaseQuantity(33);
fives.increaseQuantity(12);
tens.increaseQuantity(2);
twenties.increaseQuantity(5);
```

- Next print out how much money you have in $20 notes. Hint, you should be able to use the toString or ToString override from the Notes class.
- Print out how much money you have in $10 notes.
- Print out how much money you have in $5 notes.
- Print out how much money you have in $1 notes.
- Print out how much money you have in quarters ($0.25 coins). Hint, again, use the toString or ToString override in the Coin class..
- Print out how much money you have in dimes
- Print out how much money you have in nickels
- Print out how much money you have in pennies.
- Calculate how much money you have in total. Hint, you'll get the total value of $20 notes, add that to the total value of $10 notes, $5 notes, $1 notes, quarters, dimes, nickels and pennies.
- Calculate how much all the coins weigh. Hint, get the total weight of quarters, add that to the total weight of dimes, nickels and pennies.
- Print out a correct statement in the following format "Total Money is $219.27 total weight is 17.888oz"
- Next, ask the user how much money they want: "How much do you need?"
- Read in their answer (a float).
- Next, you are going to figure out how many of each note and coin you'd have to give the user to equal the amount they requested.
    - For example, if the user needs $22.50, you'd likely think to give them 2 ten dollar notes, 2 one dollar notes, and 2 quarters. However, in this case you don't have any quarters (you know this if you look at your output from above). So instead, you'd give them 2 ten dollar notes, 2 one dollar notes, and 5 dimes.
    - Regardless of the amount requested, your program must output the correct combination of notes and coins to equal that amount. One note/coin per line.
    - Hint: Start with the $20 note. If they ask for an amount greater than $20, you'll keep giving them $20's until the amount left to give is less than 20. Then see if you should give them $10's, then $5's, then $1's, then Quarters, then Dimes, then Nickels, and finally Pennies.

    - If at the end you don't have enough money (i.e. you still owe them more than 0.001), you should tell them that, and let them know how much you still owe them. (see examples below).
    - Remember as you give them each note/coin, you should decrease the quantity of that note/coin that you have on hand.
- Finally, print out how much money you have left, and how much the coins weigh.

Be aware, floating point numbers are not a good way to store currency in general. This is because computers cannot accurately store numbers like 0.01. They store a representation of the number. This is why above, it tells you to check if you still owe more than 0.001 rather than checking if you owe more than 0.0 because it's likely a fraction of a penny will still be owed at the end. You'll find that sometimes when you run your code it'll be off by a penny, which is ok for this assignment. If you are curious, check out this wikipedia page:
https://en.wikipedia.org/wiki/Floating-point_arithmetic#Accuracy_problems

Example Runs: [User input in red]

```
$100.00 in $20.00 notes.
$20.00 in $10.00 notes.
$60.00 in $5.00 notes.
$33.00 in $1.00 notes.
$0.00 in $0.25 coins.
$4.10 in $0.10 coins.
$0.85 in $0.05 coins.
$1.32 in $0.01 coins.
Total Money is $219.27 total weight is 17.888oz
How much do you need?
42.47
Give them a $20 note
Give them a $20 note
Give them a $1 note
Give them a $1 note
Give them a dime
Give them a dime
Give them a dime
Give them a dime
```

```
Give them a nickel
Give them a penny
Give them a penny
I have $176.80 left, it's total weight is 17.216oz
```

Run #2:

```
$100.00 in $20.00 notes.
$20.00 in $10.00 notes.
$60.00 in $5.00 notes.
$33.00 in $1.00 notes.
$0.00 in $0.25 coins.
$4.10 in $0.10 coins.
$0.85 in $0.05 coins.
$1.32 in $0.01 coins.
Total Money is $219.27 total weight is 17.888oz
How much do you need?
19.27
Give them a $10 note
Give them a $5 note
Give them a $1 note
Give them a $1 note
Give them a $1 note
Give them a $1 note
Give them a dime
Give them a dime
Give them a nickel
Give them a penny
Give them a penny
I have $200.00 left, it's total weight is 17.376oz
```

Run #3:

```
$100.00 in $20.00 notes.
$20.00 in $10.00 notes.
$60.00 in $5.00 notes.
$33.00 in $1.00 notes.
$0.00 in $0.25 coins.
$4.10 in $0.10 coins.
```

$0.85 in $0.05 coins.
$1.32 in $0.01 coins.
Total Money is $219.27 total weight is 17.888oz
How much do you need?
220.10
Give them a $20 note
Give them a $20 note
Give them a $20 note
Give them a $20 note
Give them a $20 note
Give them a $10 note
Give them a $10 note
Give them a $5 note
Give them a $5 note
Give them a $5 note
Give them a $5 note
Give them a $5 note
Give them a $5 note
Give them a $5 note
Give them a $5 note
Give them a $5 note
Give them a $5 note
Give them a $5 note
Give them a $5 note
Give them a $1 note
Give them a $1 note
Give them a $1 note
Give them a $1 note
Give them a $1 note
Give them a $1 note
Give them a $1 note
Give them a $1 note
Give them a $1 note
Give them a $1 note
Give them a $1 note
Give them a $1 note
Give them a $1 note
Give them a $1 note
Give them a $1 note
Give them a $1 note
Give them a $1 note
Give them a $1 note

Give them a $1 note
Give them a $1 note
Give them a $1 note
Give them a $1 note
Give them a $1 note
Give them a $1 note
Give them a $1 note
Give them a $1 note
Give them a $1 note
Give them a $1 note
Give them a $1 note
Give them a $1 note
Give them a $1 note
Give them a $1 note
Give them a dime
Give them a dime
Give them a dime
Give them a dime
Give them a dime
Give them a dime
Give them a dime
Give them a dime
Give them a dime
Give them a dime
Give them a dime
Give them a dime
Give them a dime
Give them a dime
Give them a dime
Give them a dime
Give them a dime
Give them a dime
Give them a dime
Give them a dime
Give them a dime
Give them a dime
Give them a dime
Give them a dime
Give them a dime
Give them a dime
Give them a dime

Give them a dime
Give them a dime
Give them a dime
Give them a dime
Give them a dime
Give them a dime
Give them a dime
Give them a dime
Give them a dime
Give them a dime
Give them a dime
Give them a dime
Give them a dime
Give them a nickel
Give them a nickel
Give them a nickel
Give them a nickel
Give them a nickel
Give them a nickel
Give them a nickel
Give them a nickel
Give them a nickel
Give them a nickel
Give them a nickel
Give them a nickel
Give them a nickel
Give them a nickel
Give them a nickel
Give them a nickel
Give them a nickel
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny

Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny

Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny

```
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
Give them a penny
I don't have enough money.  I still owe you $0.83
I have $0.00 left, it's total weight is 0oz
```

Submitting your answer:

Please follow the posted submission guidelines here:
https://ccse.kennesaw.edu/fye/submissionguidelines.php

Ensure you submit before the deadline listed on the lab schedule for CSE1322L here:
https://ccse.kennesaw.edu/fye/courseschedules.php

Rubric:

Coins Class (35 points total):

- 3 attributes (6 points total)
    - 1 point for attribute defined private, one for correct type used.
- Constructor (6 points total)
    - Takes in 2 parameters of correct type (2 points)
    - Correctly sets object variables (4 points)
- Method getTotalWeight (3 points total)
    - Takes no parameters (1 point)
    - Returns a float (1 point)
    - Correctly calculates total wight (1 point)
- Method getTotalValue (3 points total)
    - Takes no parameters (1 point)
    - Returns a float (1 point)
    - Correctly calculates total value(1 point)
- Method increaseQuantity (3 points total)
    - Takes one parameter of type int (1 point)
    - Correctly increases the quantityOnHand by the parameter (2 points)
- Method decreaseQuantity (3 points total)
    - Takes one parameter of type int (1 point)
    - Correctly decreases the quantityOnHand by the parameter (2 points)
- Method getQuantityOnHand (3 points total)
    - Takes in no parameter (1 point)
    - Returns an integer (1 point)
    - Correctly returns the quantityOnHand.
- Override of toString/ToString (5 points total)
    - Correct override method header (2 points)
    - Correctly returns the appropriate string (2 points)
    - Uses prettyPrint() correctly (1 point)
- printPretty() (3 points total)
    - Takes in a float (1 point)

    - Returns a string (1 point)
    - Copied correct line to format to a currency format.

Notes Class (28 points total):

- 2 attributes (4 points total)
    - 1 point for attribute defined private, one for correct type used.
- Constructor (4 points total)
    - Takes in 1 parameters of correct type (1 points)
    - Correctly sets object variables (2 points)
- Method getTotalValue (3 points total)
    - Takes no parameters (1 point)
    - Returns an int (1 point)
    - Correctly calculates total value(1 point)
- Method increaseQuantity (3 points total)
    - Takes one parameter of type int (1 point)
    - Correctly increases the quantityOnHand by the parameter (2 points)
- Method decreaseQuantity (3 points total)
    - Takes one parameter of type int (1 point)
    - Correctly decreases the quantityOnHand by the parameter (2 points)
- Method getQuantityOnHand (3 points total)
    - Takes in no parameter (1 point)
    - Returns an integer (1 point)
    - Correctly returns the quantityOnHand.
- Override of toString/ToString (5 points total)
    - Correct override method header (2 points)
    - Correctly returns the appropriate string (2 points)
    - Uses prettyPrint() correctly (1 point)
- printPretty() (3 points total)
    - Takes in a float (1 point)
    - Returns a string (1 point)
    - Copied correct line to format to a currency format.

Driver class/program (37 points total):

- Correctly copied/pasted object instantiation & increaseQuantity code (2 points total)
- Correctly uses toString/ToString in each object to print out value of each note/coin (There should be 8 lines - 1 point each)
- Correctly calculates the total value of money they have (5 points)

- Correctly calculates the total weight of the coins (4 points)
- Correctly prints out the total value of money as well as total weight of coins (1 point)
- Correctly calculates how many $20's to give. Prints out the right number of "Give them a $20 note" lines, and correctly keeps track of how many $20's are left. (2 points)
- Correctly calculates how many $10's to give…(2 points)
- Correctly calculates how many $5's to give…(2 points)
- Correctly calculates how many $1's to give…(2 points)
- Correctly calculates how many quarters to give…(2 points)
- Correctly calculates how many dimes to give…(2 points)
- Correctly calculates how many nickels to give…(2 points)
- Correctly calculates how many pennies to give…(2 points)
- Correctly calculates the final amount of money left, and it's weight and prints it (1 point).