**Terms to review:**

identifier

variable

constant

declaration

initialization

assignment

argument

data type

block

scope

nesting

truth table

definite loop

indefinite loop

infinite loop

iteration

accumulator

array

array index

array element

parallel array

initializer list

linear search

range check/match

**function**

**method**

**invoke / call**

**calling method**

**called method**

**method declaration**

**method body**

**return type**

**parameter**

**parameter list**

**argument**

**local variable**

**nested method call**

**return statement**

**implementation hiding**

**Homework & Labs**

*// Please name your projects LB1, LB2, LB3, etc*

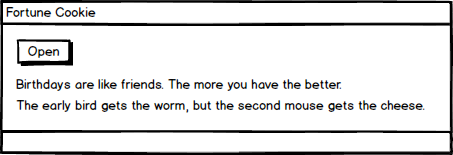
LB1. Complete Naming Conventions Handout

LB2. Complete Data Types Handout

LB3. Write a GUI application that that contains an array of the following fortune cookie sayings and randomly selects two different phrases.

1. Birthdays are like friends. The more you have the better.
2. Your smile is a treasure to all who know you.
3. You never hesitate to tackle the most difficult problems.
4. The most obvious solution is not always the best.
5. If you want the rainbow, you will have to tolerate the rain.
6. The early bird gets the worm, but the second mouse gets the cheese.
7. From listening comes wisdom and from speaking repentance.
8. A smile is your personal welcome mat.

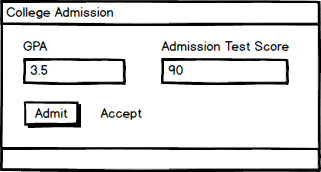
Create a method named **GenerateFortune()** that accepts no parameters and returns two strings.



LB4. Write a GUI program that determines eligibility for college applicants:

* Prompt the user for the high-school GPA and admission test score.
* Display “Accept” if the student’s GPA is 3.0 or higher and their admission test score is 60 or greater.
* Display “Accept” if the student’s GPA is less than 3.0 and their admission test score is 80 or greater.
* Otherwise display “Reject”

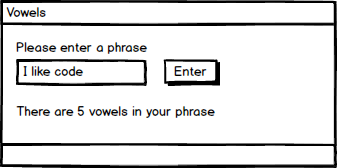
Create a method named **IsStudentAccepted()** that accepts a GPA and an admission test score, and returns a boolean indicating whether or not the student was accepted. **This method should not have any side effects.**



LB5. Write a GUI application that counts the number of vowels in a phrase that is entered by the user. For this exercise, count both uppercase and lowercase vowels, but do not consider y to be a vowel.

Create a method named **CountVowels()** that accepts a phrase and returns the number of vowels in that phrase. **This method should not have any side effects.**

Hint: Treat the string like an array of characters.

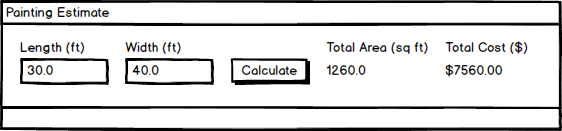


LB6. Write a GUI application that estimates the cost of painting a room. The price of the job is $6 per square foot. Assume that the room has four full walls and 9-foot ceilings.

Create the following methods:

* Create a method named **CalculateTotalArea()** that accepts the length and width of the room in feet and returns the total area of the room in square feet.
* Create a method named **CalculatePaintEstimate()** that accepts the total area the room in square feet and returns the total cost of the job in dollars.

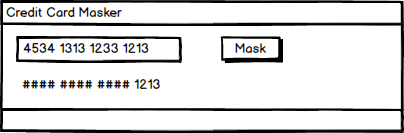
**All of the above methods should not have any side effects.**



LB7. Write a GUI application that masks all but the last 4 digits of a credit card number with a X

1. Leave the last four digits as is.
2. Leave any spaces as is.
3. Replace any number or letter with a #
   1. HINT\*\* Research Char.IsDigit() & Char.IsLetter())
4. **The program should work for credit card numbers that are any length (including numbers that are less than four digits long).**
5. **The program must support credit card numbers with spaces, dashes, or no separator.**

Create a method named **MaskNumber()** that accepts an unmasked string, a replacement character, the number of digits to preserve, then returns the masked string. **This method should not have any side effects.**



LB8. Write a GUI application that estimates the cost of a custom built desk based on a few of factors.

The user can select from four kinds of wood:

* Enter "m" for mahogany
* Enter "o" for oak
* Enter "p" for pine
* Enter anything else for other

Pine desks have a base cost of $100.

Oak desks have a base cost of $140.

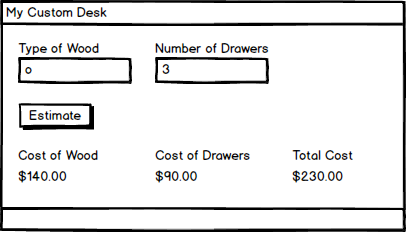
All other woods have a base cost of $180.

There is a $30 surcharge added for each drawer.

Create the following methods:

* **GetWood()** accepts no parameters, and returns the type of wood as a string ("mahogany", "oak", "pine", "other").
* **GetDrawers()** accepts no parameters, and returns the number of drawers as an int.
* **CalculateWoodCost()** accepts type of wood as a string, and returns the cost of the wood as a double.
* **CalculateDrawerCost()** accepts the number of drawers as an int, and returns the cost of the drawers as a double.
* **CalculateTotalCost()** accepts type of wood as a string and the number of drawers as an int, and returns the total cost of the desk as a double.

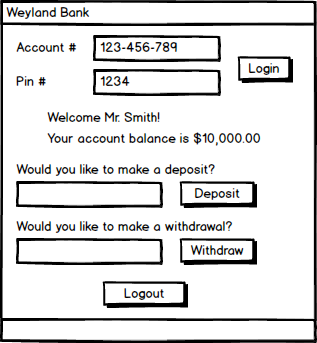
**All of the above methods should not have any side effects.**



Lb9. Create a prototype for an online bank.

* The program requires an account number and pin number to login.
* Once logged in, the user can deposit or withdraw money.
* When the user is done, they can log out of the system.
* There can only be one user logged in at a time.

**Screen Mockups**



**Methods**

**Login()** accepts an account number and pin number as strings. If valid, logs the user in, displays a welcome message, and displays their balance. If invalid, logs out the previous user and displays an error message.

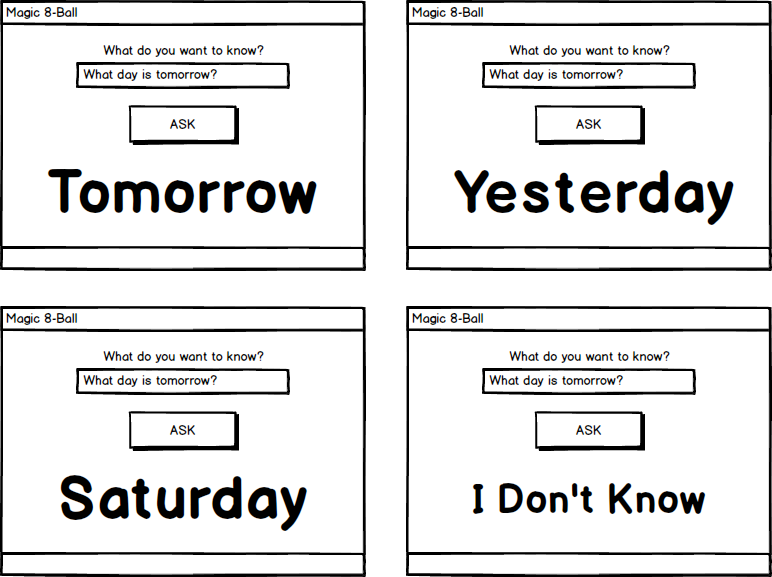
**Logout()** logs out the active user.

**MakeDeposit()** accepts a deposit amount as a decimal and displays the user's updated balance.

**MakeWithdrawal()** accepts a withdrawal amount as a decimal and displays the user's updated balance.

Lb10. Make a digital version of a Magic 8-Ball.

**Screen Mockups**



**Methods**

**ShowAnswer()** displays a random answer to the user's question.

(Always displays "I Don't Know" if the user does not enter a question.)