**Terms to review:**

identifier

variable

constant

data type

value data type

reference data type

class

object/instance

constructor

field (aka. member variable or instance variable)

method

encapsulation

composition

aggregation

**polymorphism**

**inheritance**

**base class**

**subclass**

**overloaded method**

**overridden method**

**virtual method**

**abstract method**

**abstract class**

**interface**

**strategy pattern**

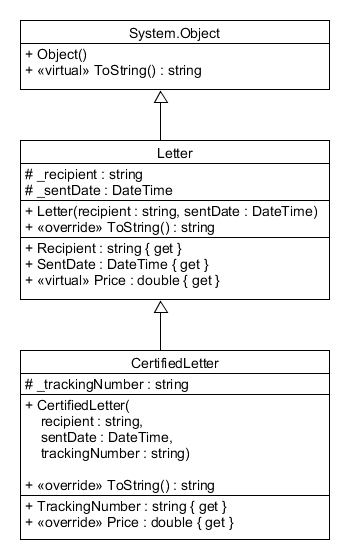
**IComparable**

**Homework & Labs**

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

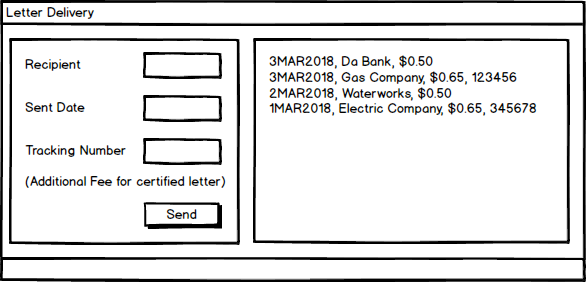
LBI. Complete Naming Conventions Handout

LBII. Complete Data Types Handout

**LB1 Letters**

Create a GUI application for a letter delivery service.

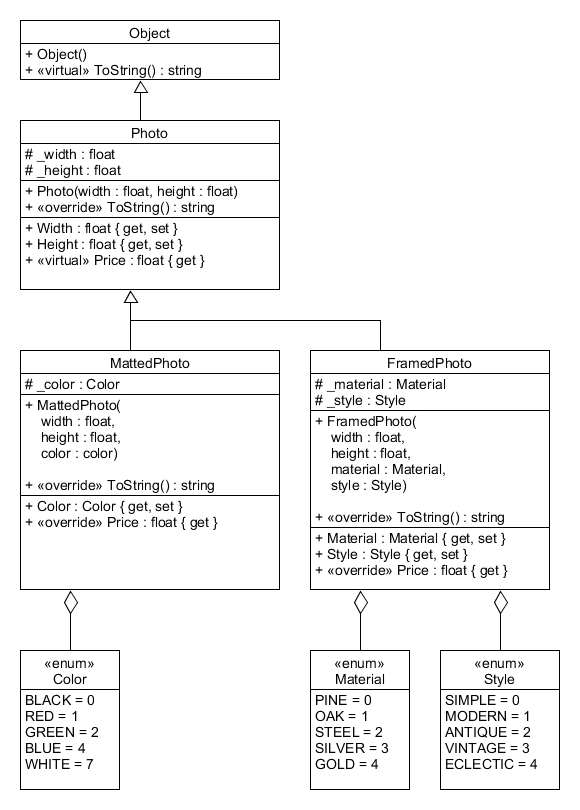
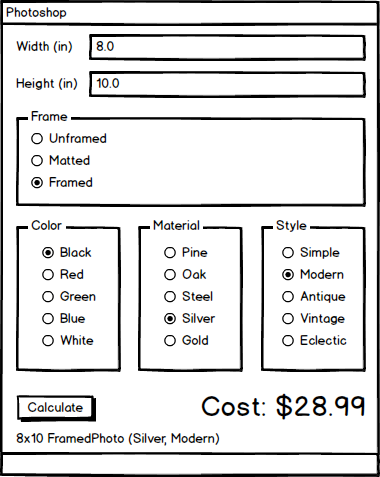
* Letters cost 50 cents to send.
* Certified letter cost an additional 15 cents to send, but include a tracking number.
* The user can send up to 20 letters.
* Every time a letter is sent it will be added to the **top** of the transaction log.
* Store the list of sent letters in a single array.
* **If the letter does not include a tracking number then instantiate a Letter object.**
* **If the letter includes a tracking number then instantiate a CertifiedLetter object.**



**LB2 Photos**

Write a GUI application for a photography business.

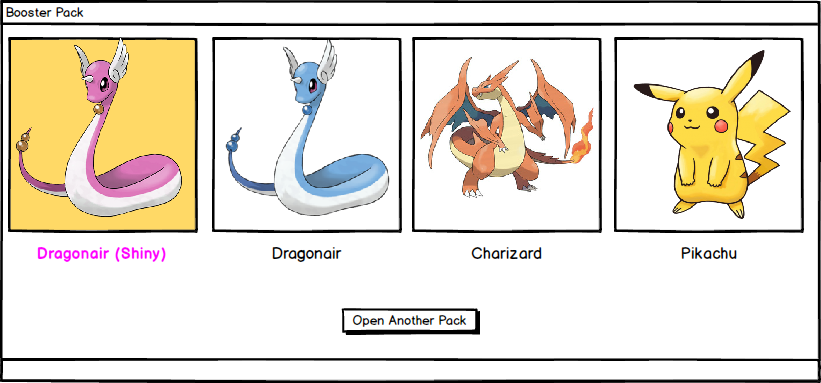
* 8" by 10" photos are $3.99
* 10" by 12" photos are $5.99
* All other sizes are $9.99
* Customers can add either a Matte or a Frame.
* Matted photos are an additional $10.
* Framed photos are an additional $25.
* Override the ToString() methods on the Photo, MattedPhoto, and FramedPhoto classes so that they display all of the information in the fields.

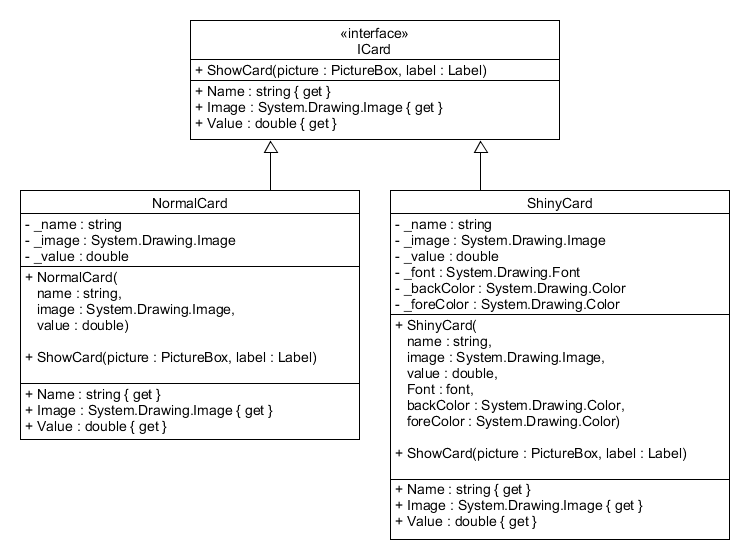


**LB3 BoosterPack**

Write a GUI application to randomly generate a pack of pokemon cards.

* Each pack has 10 cards
* There is a 20% chance of getting a shiny pokemon
* Shiny pokemon cards must be displayed with different font styles, foreground colors, and background colors (from the regular version)
* Sort the cards in the booster pack by value (using IComparable).

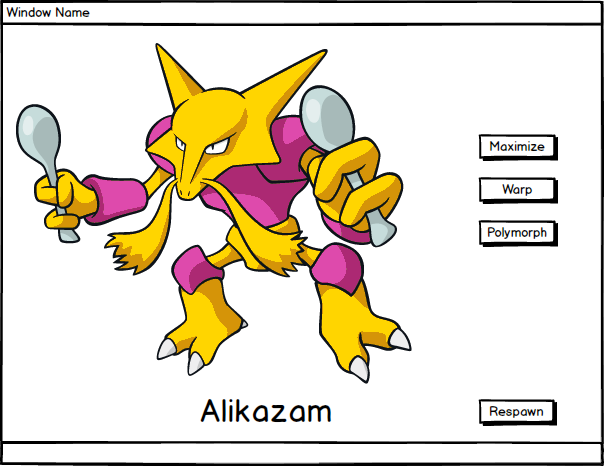




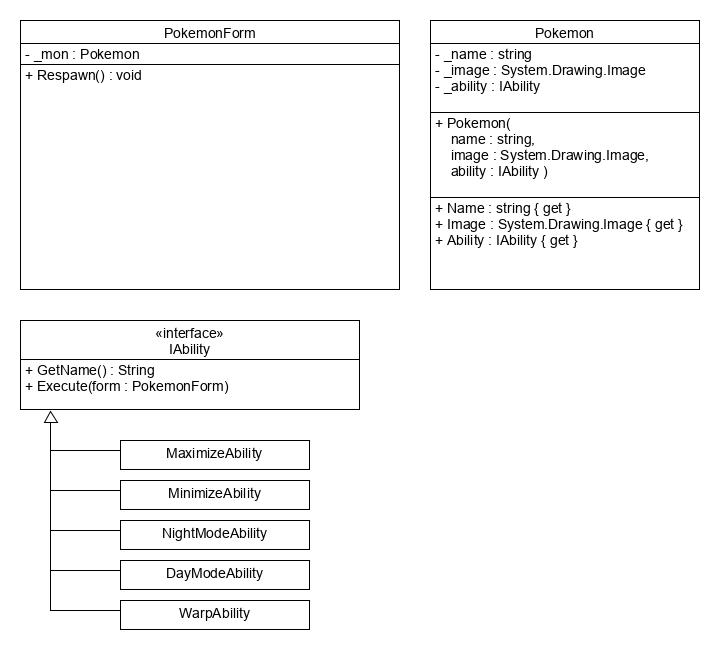
**LB4 Polymorph**

Write a GUI application that spawns a pokemon with a random ability.

* There will be a button to activate the ability. *(Only one ability button should be added in the designer. Not as shown in diagram.)*
* Each ability must have a different behavior. Examples:
  + Maximize the window
  + Minimize the window
  + Warp the window to a random screen position
  + Change the font style, background color, and foreground colors.
  + Make the window bounce
  + Make the window shake
  + Transform the pokemon into random farm animal
* The button must display the name of the ability.
* There must be at least 5 different possible abilities. (See UML for suggested abilities.)
* The user can click the Respawn button to spawn a new random pokemon.



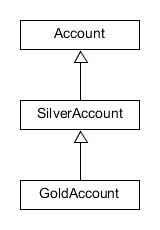
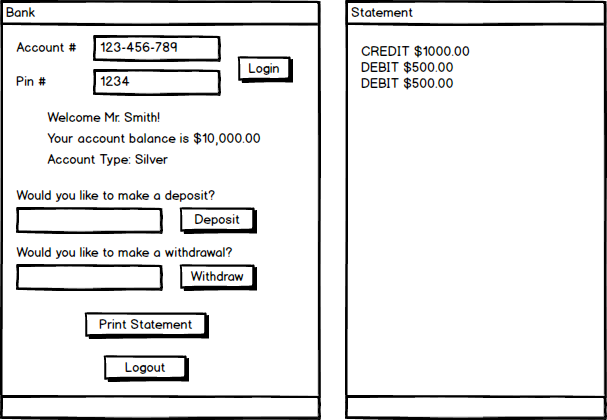
*See UML diagrams on next page*



**LB5 BankLoginEnhanced**

Write a GUI application for a bank.

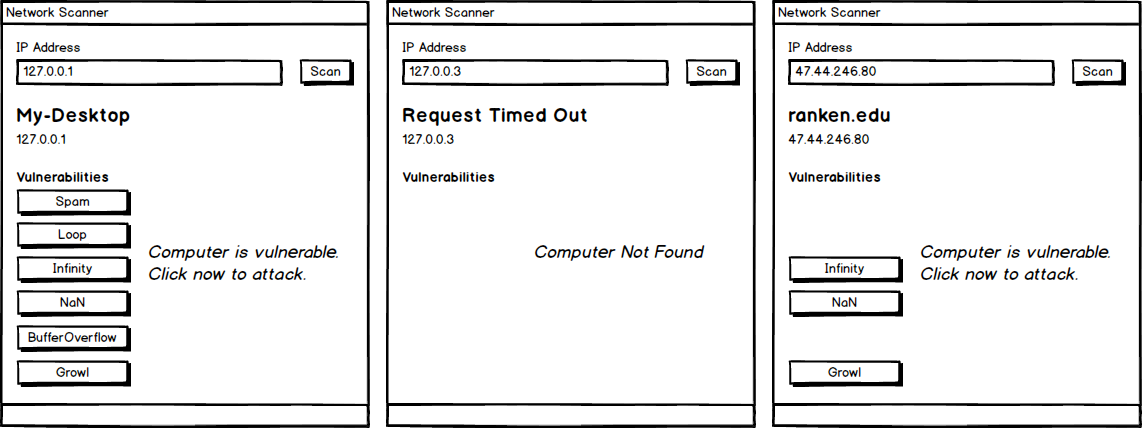
* Users must enter their username and pin number to login.
* The program should have at least 5 accounts.
* There can only be one user logged in at a time.
* Once logged in, the user can deposit money, withdraw money, or print their last 10 transactions.
* When the user is done, they can log out of the system.
* Deposits do not have an associated transaction fee.
* Withdrawals have a 1% transaction fee.
* Users cannot make a withdrawal that would leave them with less than their minimum balance.
* There are three different account types, each has different perks.
* **Bronze** accounts have a minimum balance of $0 and can only make view their current balance,deposit money and withdraw money.
* **Silver** accounts have a minimum balance of $1000 and can also print out their past 10 transactions (show them in a separate window).
* **Gold** accounts have a minimum balance of $5000, can print a **fancy statement,** and do not pay any transaction fees.



**LB6 NetworkScanAttack**

Write a GUI application that can scan a network for vulnerable computers and exploit those weaknesses.

* ***This program will not scan any real computers or execute actual exploits.***
* There are 5 computers on the hypothetical network.
* Each computer has 2-5 open vulnerabilities.
* Each vulnerability requires a different exploit.
* Clicking the button for the vulnerability will execute the required exploit in a new window.
* Exploits:
  + Show 3 spam messages in separate windows
  + Infinite loop
  + Divide by zero using integer division
  + Try to parse the string "ABCXYZ" as an int
  + Try to access an array index past the end of an array
  + Growl like Pikachu, Charizard, Squirtle, or Bulbasaur

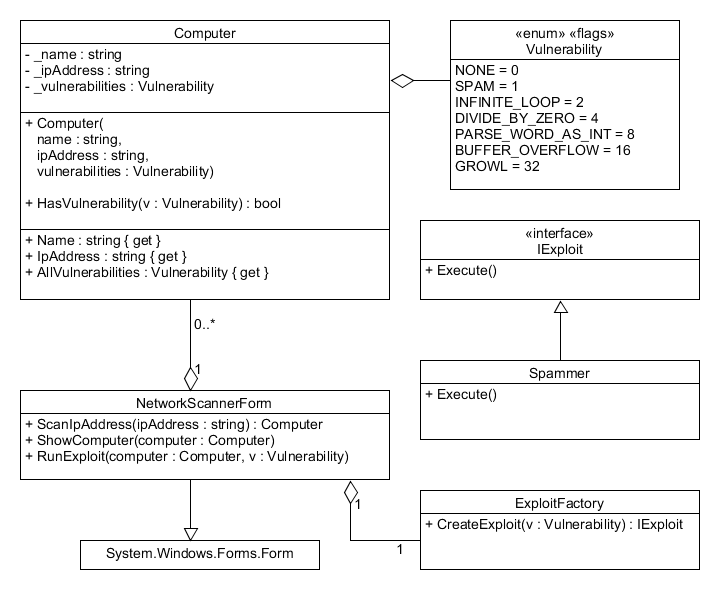


*See UML diagrams on next page*

*See* [*https://youtu.be/kztYsmUJNU4*](https://youtu.be/kztYsmUJNU4) *for how to use the [Flags] attribute with enums*

*More info here:*

[*https://docs.microsoft.com/en-us/dotnet/api/system.flagsattribute?view=netframework-4.7.1*](https://docs.microsoft.com/en-us/dotnet/api/system.flagsattribute?view=netframework-4.7.1)



**LB7 Build-A-Lab**

Create your own lab.

* Submit your design to the bin in class.
* Submit your implementation to bitbucket.
* Program must have tab order configured.
* Program must follow naming conventions for all controls, variables, constants, methods, classes, and enumerations.
* **Program must have at least one interface.**
* **Program must have at least two classes.**
* **Program must demonstrate polymorphism.**