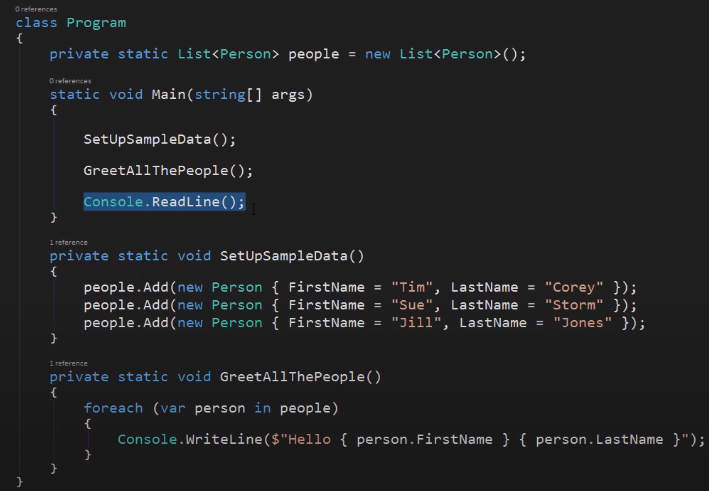
Robiąc projekt aplikacji do zarządzania plikami pootwieraj sobie:

* dokumentację z exceptions google ---> najlepsze rozwiązania dotyczące wyjątków,
* książkę z treeView, mdi - Microsoft Visual C# .NET - Jon Jagger, John Sharp ---> Część IV,
* C# księga przykładów rozdział 1.2,
* SOLID z csharpcorner.com,
* klasa abstraykcyjna (planowana rozbudowa, 1 dziedziczenie) czy interfejs (kontrakt, tylko metody abstrakcyjne i właściwości, kilka implementacji)
* Never trust the user z IamTimCorey ---> C# best practises, i rób dokładnie i powoli wg tego.

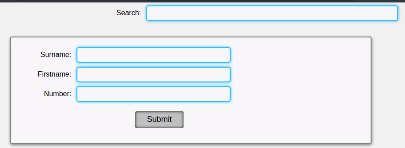
Zanim stworzysz jakąś klasę czy metodę, zastanów się najpierw aby była zgodna z powyższymi zasadami tzn. miała **dokładnie obsłużone możliwe wyjątki**, oraz **dobrze zaprojektowany widok** np. użycie kontrolek spliter do podziału ekranu pomiędzy treeView oraz szczegółowego panelu, zmiana położenia, czy rozmiaru ikon, menu kontekstowe itp, **posiadała dobrze zaprojektowane GUI**, które m.i. oddziela warstwę logiczną od warstwy prezentacyjnej, a na tekstowej konsoli wyświetla działania użytkownika oraz błędy, **spełnia wszystkie 5 założeń SOLID**, a także jest **dostosowane do wymagań użytkownika**.

Ogólne zasady:

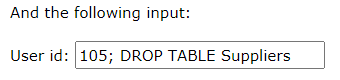
**1) Każda funkcja robi jedną rzecz:**

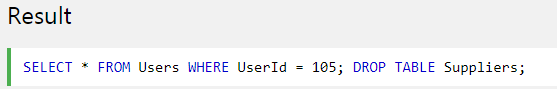


**2) never trust the user input:** (sql or javascript injection)

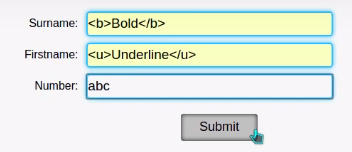
**** (książka telefoniczna)

Gdy użytkownik wpisze  wówczas aplikacją wyświetli nam wszystkie rekordy, ponieważ zapytanie  zwróci prawdę dla każdego rekordu.





zapobiegnięcie poprzez dodanie znacznika "@": 



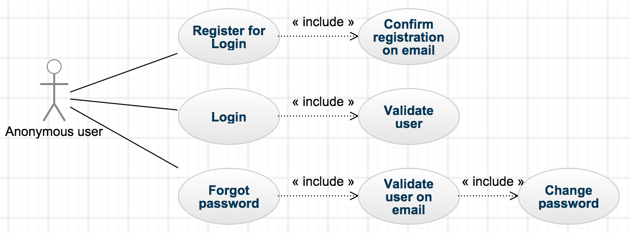
**3) Plan before you build**

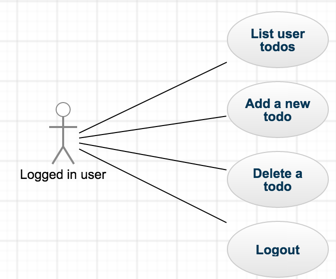
Zapisz sobie w notatniku ważne funkcjonalności, które mają być zawarte w programie i przed każdym kodowaniem zastanów co jeszcze mogłoby się pojawić lub co zmodyfikować. Odhaczaj już wykonane rzeczy. Zaplanuj również jakiej technologii i narzędzi użyjesz.

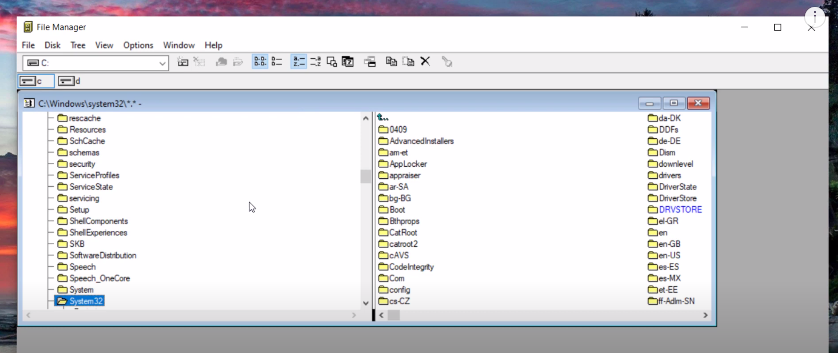
Zaprojektuj również wstępnie schemat bazy danych np. model logiczny, biznesowy, tabele, relacje (diagram encji), typy asocjacji np. 1 do 1 czy 1 do n.

**4) Avoid global variables**

Dla bardziej rozbudowanych aplikacji możesz stworzyć diagram:

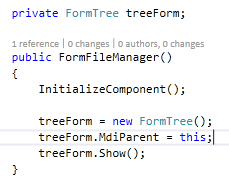




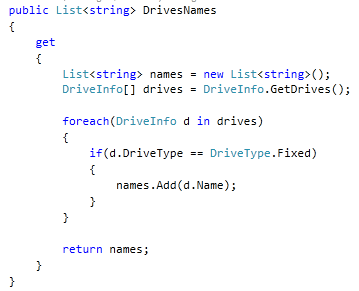


Użycie MDI dla formularza z wyborem folderów (ustawienie isParentContainer = true dla formularza głównego), użycie kontrolki treeView (po lewej stronie) oraz listView (po prawej stronie), menu. Poniżej menu dodaj toolBox z comboBoxem, w którym możesz wybierać dyski, a na samym dole okna dodaj StatusStrip, który będzie wyświetlał informacje dotyczące wybranego dysku np. rozmiar, wolne miejsce, liczba plików dla wskazanego węzła.

Wyświetlenie formularza potomnego zawierającego drzewko folderów.



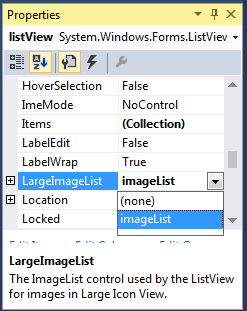
Pobranie nazwy dysków na twoim komputerze, których typ jest równy fixed (odrzucenie CD-ROMów)



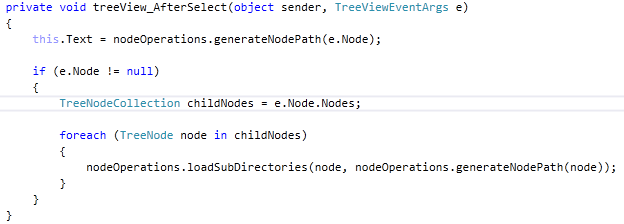
Pobranie nazw ścieżek wszystkich podfolderów dla danej ścieżki



Połączenie kontrolki imageList z właściwościami kontrolki listView takimi jak: LargeImageList oraz SmallImageList.



Po zaznaczeniu któregoś z węzłów kontrolki treeView generowana jest ściażka oraz wygenerowane są dla zaznaczonego węzła kolejne podfoldery jego podfolderów.



**Dokończ:**

1) przekopiuj błąd podczas otwierania pliku i popraw to

2) dorób zmiany layoutów w menu View na details, smal icon, large icon itp

3) funkcję search w której użyjesz rekurencję do wyszukiwania plików

4) wrzuć na forum DriveInformation i IDriveInformation oraz ich uzycie w formisie czy jest ok i czy można to zrobić lepiej

5) sort files by size, name, date\* 6) rozwinięcie węzłów tree z menu

CZY Z KLASĄ DIRECTORY ORAZ IDIRECTORYINFORMATION jest wszystko ok, i czy można to zrobić lepiej? FileInformations, FileOperations - czy jest sens się rozdrabniać i dzielić operacje na różne rodzaje np moving, modyfing, running itp