Ćwiczenia 4 - Polimorfizm, wzorzec prototyp, singleton

D1. W nowym projekcie wykonaj poniższe czynności:

- Stwórz klasę `Osoba` z polami `imie` i `nazwisko`. Następnie stwórz klasę potomną `Student` z polami `rokStududiów`, `numerGrupy`, `numerAlbumu`. Dodaj w obu klasach konstruktory domyślne i parametryczne (można wykorzystać kod z jednych z poprzednich ćwiczeń).
- W obu klasach stwórz metody `WypiszInfo()` wyświetlające wszystkie pola z klasy na konsoli (mają to być tylko instrukcje `Console.WriteLine()` z parametrami).
- Dodaj `new`, aby pozbyć się ostrzeżenia (warning) w Visual Studio. Czemu takie rozwiązanie nie jest najlepsze?
- Wykonaj rzutowanie w górę (`Osoba student1 = new Student()`) i wywołaj dla tego obiektu metodę `WypiszInfo()`.
- Usuń `new` (dodane w trzecim podpunkcie). Następnie do metody `WypiszInfo()` w klasie `Osoba` dodaj `virtual`, a w klasie `Student override`. Jaka jest różnica?
- Zmodyfikuj metodę `WypiszInfo()` w klasie `Student` używając `base.WypiszInfo();` (o ile nie zrobiono tego wcześniej).
- Narysuj diagram klas UML projektu.

D2. Zmodyfikuj kod z polecenia 1, by zamiast metody `WypiszInfo()` przesłonić metodę `ToString()` (zwróć uwagę na inny typ zwracany).

D3. Stwórz nowy projekt. A w nim wykonaj następujące czynności:

- Dodaj abstrakcyjną klasę `Figura`, a w niej pola `a`,`b`,`c` (z modyfikatorem `protected`) i abstrakcyjną metodą `ObliczPole()`.
- Stwórz dwie klasy potomne `Kwadrat` i `Trojkat` dziedziczące z klasy `Figura`.
- Stwórz obiekty z klasy `Figura`, `Kwadrat`, `Trojkat` (w razie potrzeby stwórz konstruktory). Czy wszystko jest możliwe?
- Dodaj w klasach `Kwadrat` i `Trojkat` przesłoniętą metodę `ObliczPole()` i wywołaj ją dla stworzonych obiektów.
- Stwórz listę obiektów z klas pochodnych do klasy `Figura`. Następnie za pomocą instrukcji `foreach` oblicz dla nich pola.
- Narysuj diagram UML projektu.

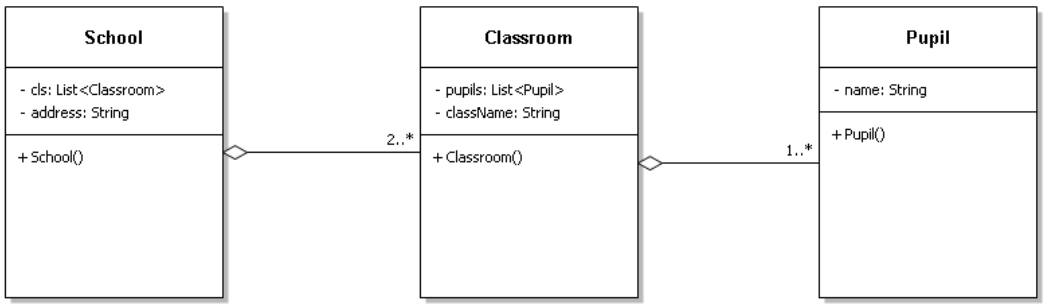
D4. Wykonaj poniższe czynności:

- pobierz (sklonuj) repozytorium https://github.com/pjastr/Ex701
- przeanalizuj plik `Program.cs` i zbudowaną łamigłówkę dziedziczenia klas.
- W metodzie `Main` stwórz dokładnie 6 obiektów, dla każdego z nich wywołaj po 2 metody tak, aby na konsoli wyjście wyglądało następująco:

```
A VirtualFun()
D VirtualFun2()
A NormalFun()
A VirtualFun()
B VirtualFun2()
A NormalFun()
B NormalFun()
C VirtualFun()
D VirtualFun()
D VirtualFun2()
```

```
B VirtualFun2()
C VirtualFun()
F VirtualFun()
F VirtualFun2()
```

D5. Na podstawie diagramu UML stwórz projekt. Następnie zaimplementuj wzorzec prototypu/głębokiej kopii. Stwórz co najmniej jeden przypadek testowy.



D6. Przeanalizuj przykłady dostępne na stronie https://refactoring.guru/design-patterns/singleton/csharp/example#example-1 Na podstawie tego przygotuj projekt z dwoma wersjami wzorca Singleton (z obsługą wątków i bez).

D7. Zmodyfikuj wzorzec singletonu tak, aby dopuszczał maksymalnie dwie instancje obiektów danego typu.

(*) czy jest możliwe aby były dokładnie dwie instancje albo zero?