Ćwiczenia 4 - Powtórzenia z programowania obiektowego cd

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. Stwórz klasę `Osoba` o dodaj do niej modyfikator `sealed`. Następnie spróbuj stworzyć klasę potomną `Student`. Jaki komunikat błędu otrzymujesz?

D5. W nowym projekcie wykonaj czynności:

- stwórz klasę `Aaa` i dodaj w niej wrtualną metodę `Metoda()` typu `void` (ma wyświetlić string `"Aaa"`).
- stwórz klasę `Bbb` dziedziczącą po `Aaa`, w nowoutworzonej klasie przesłoń metodę `Metoda()` (ma wyświetlić string `"Bbb"`) i dodaj do metody modyfikator `sealed`.
- stwórz klasę `Ccc` dziedziczącą po `Bbb`, w nowoutworzonej klasie spróbuj przesłonić metodę `Metoda()`. Jaki komunikat błędu otrzymasz?
- stwórz obiekt typu `Ccc` i wywołaj dla niego metodę `Metoda()`.

D6. Wykonaj poniższe czynności:

- Stwórz w nowym projekcie klasę częściową `Wspolrzedne` (zapiszą ją w pliku `WspolrzedneCzesc1.cs`). Następnie zadeklaruj prywatne pola `x` i `y`. Dodaj konstruktr parametry inicjujący oba pola. Następnie w pliku `WspolrzedneCzesc2.cs` stwórz drugą część klasy. Dodaj tam metodę `Pokaz()`, która na konsoli wyświetla współrzędne.

- Stwórz w klasie `Program` obiekt i wywołaj na nim metodę.
- W istniejącym projekcie pomyśl co może robić metoda częściowa?

D7. Wykonaj poniższe czynności:

- pobierz (sklonuj) repozytorium https://github.com/pjastr/Ex701
- przenalizuj 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()
```

D8. Wykonaj poniższe czynności:

- pobierz repozytorium https://github.com/pjastr/Ex702
- do klasy `Osoba` podepnij interfejs `IComparable<T>` aby było możliwe sortowanie (sam określ rodzaj po czym).