PWr Spotkanie 6
Aplikacje webowe na platformie .NET
Laboratorium – **Lista 6**

Wstęp.

Jak każdy język obiektowy również C# posiada wiele mechanizmów związanych z dziedziczeniem oraz tworzeniem interfejsów:

- Dziedziczenie zapisywane jest w postaci dwukropka po nazwie nowej klasy, po którym następuje nazwa klasy bazowej.
- W konstruktorze podklasy można wywołać konstruktor nadklasy pisząc za nagłówkiem metody dwukropek, słowo kluczowe **`base`** oraz w nawiasach okrągłych argumenty konstruktora nadklasy.
- Słowo **`base`** służy również do wywoływania metody z nadklasy, szczególnie, gdy tworzymy nową, przesłaniająca wersje metody o tej samej nazwie i liście parametrów.
- Do sprawdzanie czy wartość referencyjna jest określonego typu używa się operatora **`is`** (np. `variable` **`is`** `TypeName`) aby następnie w przypadku rzutować na określony typ poprzez (np. `(TypeName)variable`). W wielu przypadkach lepiej użyć operatora **`as`** (np. `variable` **`as`** `TypeName`) który albo rzutuje wartość referencyjna na podany typ, albo, w przypadku ewentualnego niepoprawnego rzutowania, zwraca wartość `null`. Następnie można użyć operatora .? lub ??.
- Istnieją również klasy abstrakcyjne (w nagłówku klasy przed słowem **`class`** należy umieścić słowo kluczowe **`abstract`**), które nie mogą posiadać swoich bezpośrednich instancji. Klasy takie pomogą posiadać oprócz wszystkich elementów, które posiadają nieabstrakcyjne klasy, również metody i właściwości abstrakcyjne. Takie elementy poprzedzone są również słowem `abstract` i nie mają ciała metod/właściwości.
- Tworzenie interfejsu różni się tym od tworzeniu klasy tym, że występuje słowo kluczowe **`interface`** zamiast **`class`**. Oczywiście interfejs nie jest klasą, więc w swojej definicji posiada tylko metody, które muszą być zaimplantowane w klasach, które obiecują implementacje danego interfejsu.
- W języku C# rozróżnia się jawną i niejawną implementację interfejsu. W tym pierwszym przypadku aby mieć dostęp metody z interfejsu należy referencję najpierw rzutować na interfejs, a implementując metodę w sposób jawny należy w nagłówku metody przed jej nazwą dopisać nazwę interfejsu i kropkę.

List zadań

Przygotować program demonstrujący użycie zaimplementowanych rozwiązań dla różnych przypadków użycia.

1. Napisać klasy powiązane relacją dziedziczenia: `Pojazd`, `Samochód`, `Rower`, `SamochódOsobowy`, `SamochódCiężarowy`. Klasy mają być wykorzystane do pamiętanie pojazdów w serwisie aukcyjnym. W każdej klasie stworzyć minimum 2 pola/właściwości specyficzne dla niej (np. marka, model, nośność itd.) jak również minimum jedną metodę. Wyznaczyć klasę abstrakcyjną. Każda klasa ma mieć swoją implementację metody `ToString()`. Tworzyć tablicę (ewentualnie kolekcję) obiektów klasy abstrakcyjnej. Napisać metody, które z użyciem operatorów **`is`** lub **`as`** policzą sumaryczną nośność pojazdów pod warunkiem, że pojazd posiada taką cechę jak nośność (czyli jest odpowiedniej klasy).
2. Napisać interfejs `IFigure` posiadający metodę:
   `void` `moveTo(`**`double`** `x, `**`double`** `y)`
   Oraz interfejs `IHasInterior` z właściwością do ustawienia i pobrania koloru wnętrza. Stworzyć klasy, z których jedne implementują jeden, a drugie obydwa interfejsy. Dla pierwszego interfejsu użyj implementacji niejawnej, dla drugiego implementacji jawnej. Stworzyć tablicę/kolekcję obiektów klasy **`object`**. Stworzyć metodę, która otrzyma taką tablicę/kolekcję i dla obiektów implementujących interfejs `IHasInterior` wypisze ich kolor, a dla pozostałych tekst „no color”. Kolor może być typu **`string`**.

Uwaga: w zadaniu 1 można użyć języka angielskiego do nazywania klas/metod itd.

Data I: Spotkanie 7 (max 100 punktów)
Data II: Spotkanie 8 (max 80 punktów)
Data III: Spotkanie 9 (max 50 punktów)