main 1 branch 0 tags

Go to file Add file Code

kmolenda Update README.md 2ec3fe5 on 19 Nov 2020 6 commits

UnitTests	Initial setup	17 months ago
.gitignore	Initial commit	17 months ago
README.md	Update README.md	17 months ago

README.md

Lab-03. *Well formed class* - `Pudelko`

- Krzysztof Molenda, ver. 0.2/2020

Cel, zakres

- *well-formed type*, implementacje kluczowych interfejsów (`IEquatable` , `IFormatable` , `IEnumerable`), przeciążenia operatorów (`==` , `!=` , jawnej i niejawnej konwersji, indekser), metody rozszerzające, definiowanie porządku sortowania (delegat `Comparison`), testy jednostkowe
- orientacyjny czas realizacji - przy sprawnym programowaniu ok. 60 minut
- orientacyjna ilość kodu - ok. 200 linii dla implementacji klasy, ok. 200 dla implementacji testów jednostkowych

---

Sformułowanie problemu

Współpracujesz w rozwoju systemu wspomagającego obsługę Firmy Kurierskiej i optymalizującego załadunek. Twoim zadaniem jest opracować klasę `Pudelko` .

Pudełko to prostopadłościan o zadanych długościach krawędzi (umownie: długość, wysokość, szerokość). Wymiary te mogą być podawane w milimetrach, centymetrach bądź metrach - jako wartości rzeczywiste. Cyfry poza zakresem dla określonej jednostki są odcinane (np. dla `2.54637 m` przyjmujemy `2.546 m` czyli `254.6 cm` , czyli `2546 mm`)!

Przyjmujemy, że maksymalny pojedynczy wymiar pudełka nie przekracza 10 metrów (ze względu na ograniczone możliwości załadunkowe).

Twoim zadaniem jest zaimplementowanie klasy `Pudelko` spełniającej podane poniżej warunki:

- Obiekty klasy `Pudelko` są niezmiennicze.
- Domyślnym pudełkiem jest prostopadłościan o krawędziach `a` , `b` , `c` o wymiarach odpowiednio 10 cm × 10 cm × 10 cm.
- Klasa jest zapieczętowana (nie można z niej dziedziczyć).

> Sugestia: w aplikacji, zamiast używać długiej nazwy `Pudelko` , możesz zastosować alias (np. `P` wpisując `using P = MyLib.Pudelko;`)

> ⚠ Wykorzystaj testy jednostkowe (MS Test v2) dołączone do zadania (testujące tylko część funkcjonalności). Dopisz własne metody testujące, dodaj własne przypadki testowe, modyfikuj kod testów - jeśli uznasz to za stosowne.

Wytyczne

1. Ograniczenia wymiarów
   - wymiary pudełka muszą być dodatnie
   - wymiary pudełka nie mogą przekroczyć 10 m
2. Tworzenie obiektu `Pudelko`
   - Domyślnym pudełkiem jest prostopadłościan o wymiarach 10 cm × 10 cm × 10 cm.

- Tworzenie obiektu odbywa się na podstawie podania listy wartości długości boków (`a`, `b`, `c`) oraz jednostki miary. Wszystkie parametry konstruktora są opcjonalne.
  - Typ reprezentujący jednostki miary to `enum` o nazwie `UnitOfMeasure` definiujący składowe: `milimeter`, `centimeter`, `meter`.
  - Jeśli podano mniej niż 3 wartości liczbowe, pozostałe przyjmuje się jako o wartości 10 cm, ale dla ustalonej jednostki miary.
  - Jeśli nie podano jednostki miary, to przyjmujemy, że wymiary podawane są w metrach.
- W przypadku próby utworzenia pudełka z którymkolwiek z parametrów niedodatnim, zgłaszany jest wyjątek `ArgumentOutOfRangeException`.
- W przypadku próby utworzenia pudełka z którymkolwiek z parametrów większym niż 10 metrów, zgłaszany jest wyjątek `ArgumentOutOfRangeException`.

3. Zaimplementuj *properties* o nazwach `A`, `B` i `C` zwracające wymiary pudełka w metrach (z dokładnością do 3 miejsc po przecinku).

4. Reprezentacja tekstowa, `ToString`
   - Zapewnij reprezentację tekstową obiektu według formatu:

     `«liczba» «jednostka» x «liczba» «jednostka» x «liczba» «jednostka»`
   - znak rozdzielający wymiary, to znak mnożenia `x` (Unicode: U+00D7, *multiplication sign*, *times*)
   - pomiędzy liczbami, nazwami jednostek miar i znakami `x` jest dokładnie jedna spacja
   - domyślne formatowanie liczb (przesłonięcie `ToString()`) w metrach, z dokładnością 3. miejsc po przecinku
   - przeciążenie: funkcja `ToString(string format)` przyjmuje następujące kody formatów: `m`, `cm` oraz `mm`.
     - dla formatu `"m"` wartości podawane są ze stałą dokładnością 3. miejsc po przecinku
     - dla formatu `"cm"` wartości podawane są ze stałą dokładnością 1. miejsca po przecinku
     - dla formatu `"mm"` wartości podawane są bez miejsc po przecinku
     - dla błędnie podanego formatu zgłaszany jest wyjątek `FormatException`
   - Przykłady:
     - dla pudełka o wymiarach kolejno `2.5`, `9.321` i `0.1`, `ToString()` zwraca napis `"2.500 m x 9.321 m x 0.100 m"`
     - dla pudełka o wymiarach kolejno `2.5`, `9.321` i `0.1`, `ToString("m")` zwraca napis `"2.500 m x 9.321 m x 0.100 m"`
     - dla pudełka o wymiarach kolejno `2.5`, `9.321` i `0.1`, `ToString("cm")` zwraca napis `"250.0 cm x 932.1 cm x 10.0 cm"`
     - dla pudełka o wymiarach kolejno `2.5`, `9.321` i `0.1`, `ToString("mm")` zwraca napis `"2500 mm x 9321 mm x 100 mm"`
   - formalnie zadanie to polega na implementacji interfejsu `IFormatable`

5. Zaimplementuj *property* `Objetosc` zwracające objętość pudełka w m^3. Wynik zaokrąglij (`Math.Round`) do 9. miejsc po przecinku.

6. Zaimplementuj *property* `Pole` zwracające pole powierzchni całkowitej pudełka (prostopadłościanu) w m^2. Wynik zaokrąglij (`Math.Round`) do 6. miejsc po przecinku.

7. Equals
   - Dwa pudelka są takie same, jeśli mają takie same wymiary w tych samych jednostkach, z dokładnością do kolejności wymiarów, tzn. pudełko `P(1, 2.1, 3.05)` jest takie samo jak pudełko `P(1, 3.05, 2.1)`, a to jest takie samo jak `P(2.1, 1, 3.05)`, a to jest takie samo jak `P(2100, 1000, 3050, unit: UnitOfMeasure.milimeter)`, i.t.d.
   - Zaimplementuj interfejs `IEquatable<Pudelko>`.
   - Zaimplementuj `Equals(object)` i `GetHashCode()`.
   - Zaimplementuj przeciążone operatory `==` oraz `!=`.

8. Zdefiniuj przeciążony operator łączenia pudełek (`+`) działający wg zasady: wynikiem `p1 + p2` jest najmniejsze pudełko o takich wymiarach, które pomieści oba pudełka (w sensie: o najmniejszej objętości). Wyobraź sobie zapakowanie pudełek `p1` oraz `p2` w jedno pudełko odpowiednio większe, ale o najmniejszych możliwych wymiarach.

9. Operacje konwersji
   - Zdefiniuj konwersję jawną (*explicit*) z typu `Pudelko` na typ `double[]`, zwracającą tablicę wartości długości krawędzi pudełka w metrach, w kolejności `A`, `B`, `C`

krawędzi pudełka w metrach, w kolejności `A`, `B`, `C`.

- Zdefiniuj konwersję niejawną (*implicit*) z typu `ValueTuple<int,int,int>` na typ `Pudelko`, przyjmując, że podawane wartości są w milimetrach.

10. Przeglądanie długości krawędzi - indexer

    - Zaimplementuj mechanizm przeglądania (tylko do odczytu, bo obiekt ma być *immutable*) długości krawędzi poprzez odwołanie się do indeksów (`p[i]` oznacza *i*-ty wymiar pudełka `p`).

11. Przeglądanie długości krawędzi - pętla `foreach`

    - Zaimplementuj mechanizm przeglądania długości krawędzi pudełka za pomocą pętli `foreach` w kolejności od `A` do `C` (np. `foreach(var x in p) { ... }`). Formalnie, jest to implementacja interfejsu `IEnumerable`.

12. Metoda parsująca ze `string`

    - Zaimplementuj statyczną metodę `Parse` komplementarną do tekstowej reprezentacji pudełka (`ToString()` oraz `ToString(format)`). Przykładowo `new P(2.5, 9.321, 0.1) == P.Parse("2.500 m × 9.321 m × 0.100 m")`.
    - Rozważ różne przypadki jednostek miar (patrz: konstruktor i formatowana metoda `ToString`).

13. Pamiętaj o zapewnieniu pełnej niezmienniczości obiektom klasy `Pudelko` oraz o zapieczętowaniu klasy.

14. Utwórz testy jednostkowe (*unit tests*) dla:

    - *properties* `Objetosc` i `Pole`,
    - operatora łączenia pudełek,
    - operatora równości pudełek,

    przy kilku, reprezentatywnych zestawach danych.

15. Metody rozszerzające

    - W projekcie typu *Console App* utwórz **metodę rozszerzającą** klasę `Pudelko` o nazwie `Kompresuj`, która zwraca pudełko sześcienne o takiej samej objętości, jak pudełko oryginalne.

16. Sortowanie pudełek

    - W funkcji `Main` programu głównego (aplikacja konsolowa) utwórz listę kliku różnych pudełek, używając różnych wariantów konstruktora.
    - Wypisz pudełka umieszczone na liście (po jednym w wierszu).
    - Posortuj tę listę według następującego kryterium: `p1` poprzedza `p2` jeśli:
      - objętość `p1` < objętość `p2`
      - jeśli objętości są równe, to decyduje pole powierzchni całkowitej,
      - jeśli również pola powierzchni całkowitej są równe, to decyduje suma długości krawędzi `A+B+C`.
    - Kryterium sortowania dostarcz jako delegat `Comparison<Pudelko>`.
    - Wypisz listę posortowaną.

W funkcji `Main` napisz kod potwierdzający poprawność wykonania poszczególnych implementacji.

Ocena

Do oceny przesyłasz *solution* z projektami, skompresowane w formie archiwum ZIP. Przed spakowaniem sprawdź, czy kod się kompiluje i usuń pliki binarne (*Build-> Clean Solution*). Usuń foldery `bin/`, `obj/` czy `TestResults` zawierające pliki binarne - przesyłasz tylko strukturę projektu z plikami konfiguracyjnymi oraz kod w plikach `.cs`. Kod z błędami nie będzie oceniany.

Rozmiar klasy `Pudelko` - w moim przypadku - nie przekroczył 200 linii kodu, przy zachowaniu standardowego, automatycznego formatowania C#. Nie zwracałem uwagi na maksymalne uproszczenie czy kompresję zapisu.

⚠ Ocenie podlega kompletność i jakość kodu (w szczególności nie powielanie kodu, łączenie kodu).