Zadanie na Lab. 4

Zadanie I (za 6 pkt.):

Zadanie

W „zadaniu przed lab. 4" dostępnym pod adresem www

https://ktrojanowski.blog.uksw.edu.pl/2022/11/14/ztp2022-zadania-przed-lab-4/

znajduje się przykład programowania z wykorzystaniem cech charakterystycznych.

Do istniejącego zbioru klas, reprezentujących pojęcia: *temperatura wody* oraz *kostka do gry*, dodaj klasy reprezentujące pojęcia:

1. minuta dnia,
2. liczba sekund potrzebnych zegarowi ściennemu na wybicie aktualnej godziny zakładając, że uderzenie trwa 0.8 sek., a pauza między uderzeniami – 1.2 sek.,
3. ilość piwa mierzona w litrach, jaką można zamówić w pubie w Wielkiej Brytanii, zakładając że 1 pinta to 568 ml (przyjmij, że w pubie nie można zamówić 0 pint piwa),
4. liczba pierwsza,

oraz odpowiadające im konkretyzacje szablonu cech **`Cechy`**. Być może konieczne będzie dodanie nowych cech, aby wyrazić właściwości nowych pojęć. Nie może jednak pojawić się redundancja, tj. nowe cechy nie mogą wyrażać tego samego, co cechy już istniejące.

1. Do każdej z klas reprezentujących pojęcia dodaj przeciążony operator wypisywania wartości do strumienia, który również korzysta z cechy, aby wypisać odpowiednią liczbę cyfr po przecinku. Dodaj tę cechę do klasy bazowej.
2. W metodach `push` szablonu klasy `SzablonStosu` przyjmujących jako argument wartość typu `int` oraz typu `double` rozbuduj kod dokonujący walidacji liczby przekazanej w argumencie tak, aby uwzględnić nowe cechy. Uwaga: metody `push` nie podejmują naprawy nieprawidłowych wartości, a jedynie sprawdzają poprawność i tylko te poprawne umieszczają na stosie (pozostałe ignorują). W metodzie `push` przyjmującej argument typu `const T&` pozostaw kod bez zmian (przyjmij, że przekazany obiekt z założenia musi być poprawny i nie wymaga walidacji).
3. W konstruktorach nie implementujemy kodu dokonującego walidacji liczby przekazanej w argumencie, ponieważ przyjmujemy zasadę, że tworzenie nowego obiektu i związane z tym działanie konstruktora jest zawsze poprzedzone sprawdzeniem warunków w kodzie wywołującym tworzenie nowego obiektu i dlatego argumenty wywołania konstruktora są _zawsze_ poprawne.

> Hint:
> przy sprawdzaniu wartości zmiennej rzeczywistoliczbowej nie stosujemy porównania: **`if(x==y)`**,
> ale sprawdzamy wartość różnicy z pewną ustaloną tolerancją: **`if(abs(x-y)<0.001)`**.

Zademonstruj w funkcji `main` poprawne działanie napisanego kodu. Utwórz kilka kontenerów do przechowywania nowych typów danych i dodaj do nich po kilka nowych elementów. Wypisz zawartość kontenerów z nowymi typami danych w oknie konsoli.