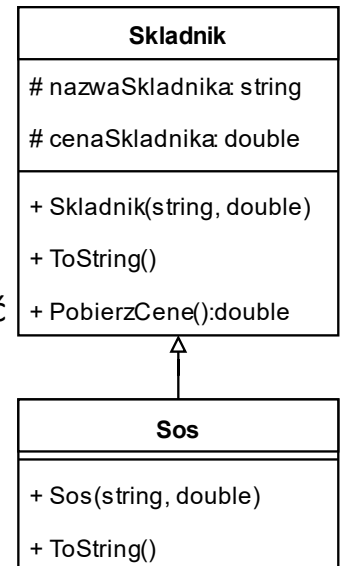


Kolokwium 2 – Wersja próbna

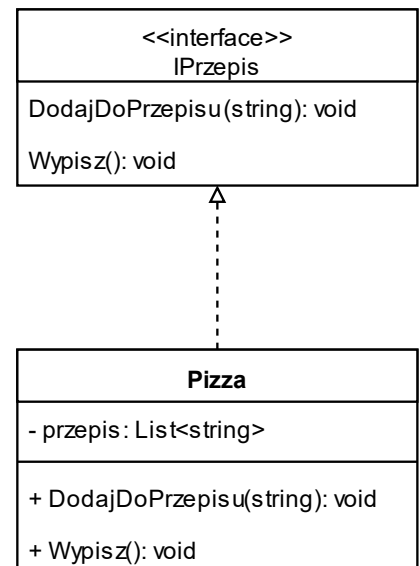
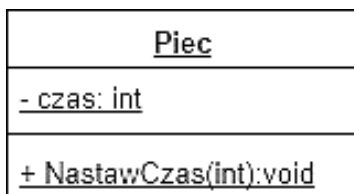
Utwórz nowy projekt – nazwij go swoim numerem albumu. Wszystkie polecenia wykonaj w ramach jednego projektu. Uwaga: nie możesz tworzyć nowych pól w klasach, modyfikatory dostępu mają nie być zmieniane. Metody własne można dodawać (bez konstruktorów), ale ich liczba powinna być jak najmniejsza.

1. Wykonaj kod zgodnie z diagramem UML:
 - a. wszystkie konstruktory mają inicjować oba pola w klasie
 - b. metoda ToString() w zależności od klasy ma zwrócić następujące string:
Składnik: {nazwaSkładnika}, cena: {cenaSkładnika}
Sos: {nazwaSkładnika}, cena: {cenaSkładnika}
 - c. metoda PobierzCene() zwraca wartość pola cenaSkładnika



2. W metodzie Main wykonaj następujące czynności:
 - a. stwórz listę pizza (List<Skladnik>)
 - b. na listę dodaj co najmniej 5 składników i jeden sos
 - c. wypisz na konsoli zawartość listy
3. W metodzie Main wykonaj następujące czynności:
 - a. posortuj listę pizza (stworzoną w punkcie 2) wg pola cena od najdroższego do najtańszego, w razie potrzeby zaimplementuj odpowiedni interfejs tylko w klasie Składnik
 - b. wypisz na konsoli elementy listy pizza o parzystych indeksach
 - c. odwróć kolejność elementów na liście pizza.
4. Wyrzuć wyjątek przy próbie stworzenie obiektu typu Składnik z ceną będą wartością liczbą ujemną. Następnie obsłuż wyjątek za pomocą bloku try...catch, tak aby w przypadku wyjątku wyświetlał odpowiedni komunikat na konsoli.
5. Dodaj abstrakcyjną klasę Zamowienie.
 - a. dodaj w niej pole czasDostawy typu string z modyfikatorem protected (przechowujemy tu słowny opis daty, np. dziś po południu, na jutro itp.)

- b. dodaj w tej klasie wirtualną metodę `PoprawnyCzas` bez parametru, zwracającą typ `bool`. Ma ona zwracać `true` w sytuacji kiedy `czasDostawy` jest niepusty (różny od `null`). `False` ma być zwracane w przeciwnym wypadku.
 - c. dodaj zwykłą metodę typu `void` `UstawCzasDostawy` z parametrem typu `string` która ustawia parametr jako pole `czasDostawy`.
6. Dodaj klasy potomne dziedziczące z klasy `Zamowienie`:
 - a. klasę `NaMiejscu` (można w środku zostawić ją pustą)
 - b. klasę `NaWynos`, w niej przesłoń metodę `PoprawnyCzas` tak, aby `true` był zwracane w sytuacji jeśli `czasDostawy` jest równy stringowi „jutro”, `false` w przeciwnym wypadku.
7. W metodzie `Main` wykonaj czynności:
 - a. oddziel liniijką komentarza kod poprzednich poleceń
 - b. stwórz generyczną kolejkę o nazwie zamówienia przechowującą obiekty typu `Zamowienie`
 - c. dodaj na kolejkę co najmniej 5 obiektów z klas potomnych do klasy `Zamowienie`, w dowolny sposób ustaw dla nich jakiś czas
 - d. dla wszystkich elementów z kolejki wywołaj metodę `PoprawnyCzas()`, wartości zwracane wyświetl na konsoli
 - e. za pomocą metody dostępnej tylko dla kolejek usuń wszystkie elementy z kolejki
8. W tym samym projekcie dodaj interfejs i klasę zgodnie z diagramem UML.
 - a. przepis ma być kolejką na stringi (zainicjuj ją w klasie)
 - b. metoda `DodajDoPrzepisu` ma dodawać parametr do pola `przepis`
 - c. metoda `Wypisz` ma wyświetlić aktualną zawartość pola `przepis` na konsoli
 - d. w `Main` przetestuj działanie metod
9. Dodaj kod z diagramu po lewej do projektu. Wywołaj metodę z diagramu wewnątrz `Main`.



10. Prześlij kod projektu (może być spakowany) na dowolny serwis hostingujący, link wyślij na piotr.jastrzebski@uwm.edu.pl