Cukiernia WPF.

Stworzona cukiernia ma magiczne właściwości, które nie zawsze mają zastosowanie w rzeczywistości.

1. Stwórz klasę Skladnik.

a) dodaj w niej prywatnej pola nazwaSkladnika, ilosc (oba typu string) oraz cenaSkladnika (typ double). Uwaga: ilość ma być typu string, bo mogą być wartości typu „szczypta”.

b) dodaj konstruktor parametryczny ustawiający jednoczenie nazwę, ilość i cenę składnika (kolejność dowolna, ale dalej trzeba być konsekwentnym).

c) przesłoń metodę ToString(), aby zwracała informację o składniku np. Nazwa: Sól, ilość: szczypta, cena: 0.10

d) dodaj metodę zwracającą cenę składnika

2. Do klasy Składnik podepnij interfejs IComparable<T> lub IComparable oraz zaimplementuj metodę CompareTo tak, aby sortowała składniki po ich nazwie alfabetycznie.

3. Stwórz klasę Przepis.

a) dodaj w niej następujące prywatne pola:

- nazwa typu string;

- suma typu double i nadaj jej wartość początkową zero;

- składniki będące listą i przechowującą obiekty typu Skladnik

- czasPrzygotowania typu int (jest czas potrzebny na przygotowanie w minutach)

b) dodaj metodę DodajSkladnik typu void z parametrami typów string, string, double, która doda do pola składniki kolejną pozycję po podanych parametrach. Poza dodaniem powiększ wartość pola suma o cenę składnika.

c) dodaj metodę UstawNazweICzas typu void z parametrami typów string, int, która ustawia nazwę i czas przygotowania przepisu.

d) przesłoń metodę ToString() tak, aby zwracała informację o przepisie (wypisane wszystkie składniki posortowane po nazwie, po nich ma być suma cen).

Np. Przepis:

Nazwa: Mąka, ilość: 0.5 kg, cena: 2

Nazwa: Sól, ilość: szczypta, cena: 0.10

Suma: 2.10

W przypadku gdy lista jest pusta, ma zwracać pusty string.

e) dodaj metodę bez parametru CzyCzas typu zwracanego bool, która zwraca true jeśli czasPrzygotowania jest liczbą dodatnią, a false w przeciwnym wypadku.

f) dodaj metodę IleSkladników() bez parametru zwracają liczbę typu int, która informuje o licznie dodanych składników.

4. Dodaj abstrakcyjną klasę Zamowienie.

a) dodaj w niej pole czasDostawy typu DateTime z modyfikatorem protected;

b) dodaj w tej klasie wirtualną metodę PoprawnyCzas bez parametru, zwracającą typ bool. Ma ona zwracać true w sytuacji kiedy czasDostawy jest większy od aktualnej daty i godziny pobranej z systemu. False ma być zwracane w przeciwnym wypadku.

c) dodaj zwykłą metodę typu void UstawCzasDostawy z parametrem typu DateTime, która ustawia parametr jako pole czasDostawy.

5. 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 było zwracane w sytuacji jeśli czasDostawy jak większy o 2 godziny od aktualnego czasu pobranego z systemu.

6. Zaimplementuj aplikację WPF do przetestowania powyższego kodu. Nie należy tworzyć metod, które powielają metody stworzone w punktach 1-5.

Logika działania aplikacji:

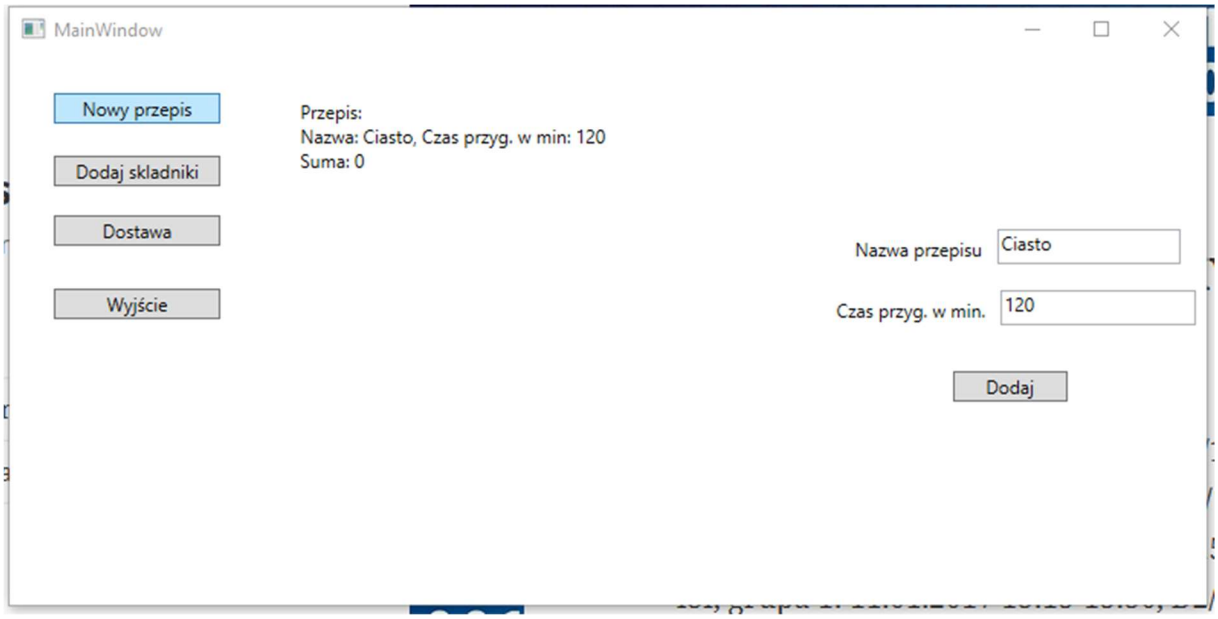
- najpierw należy poprosić użytkownika o podanie nazwy przepisu i czasu przygotowania, przy próbie zapisania tych danych należy sprawdzić, że nazwa nie jest pusta, a czas liczbą dodatnią mniejszą niż 300.

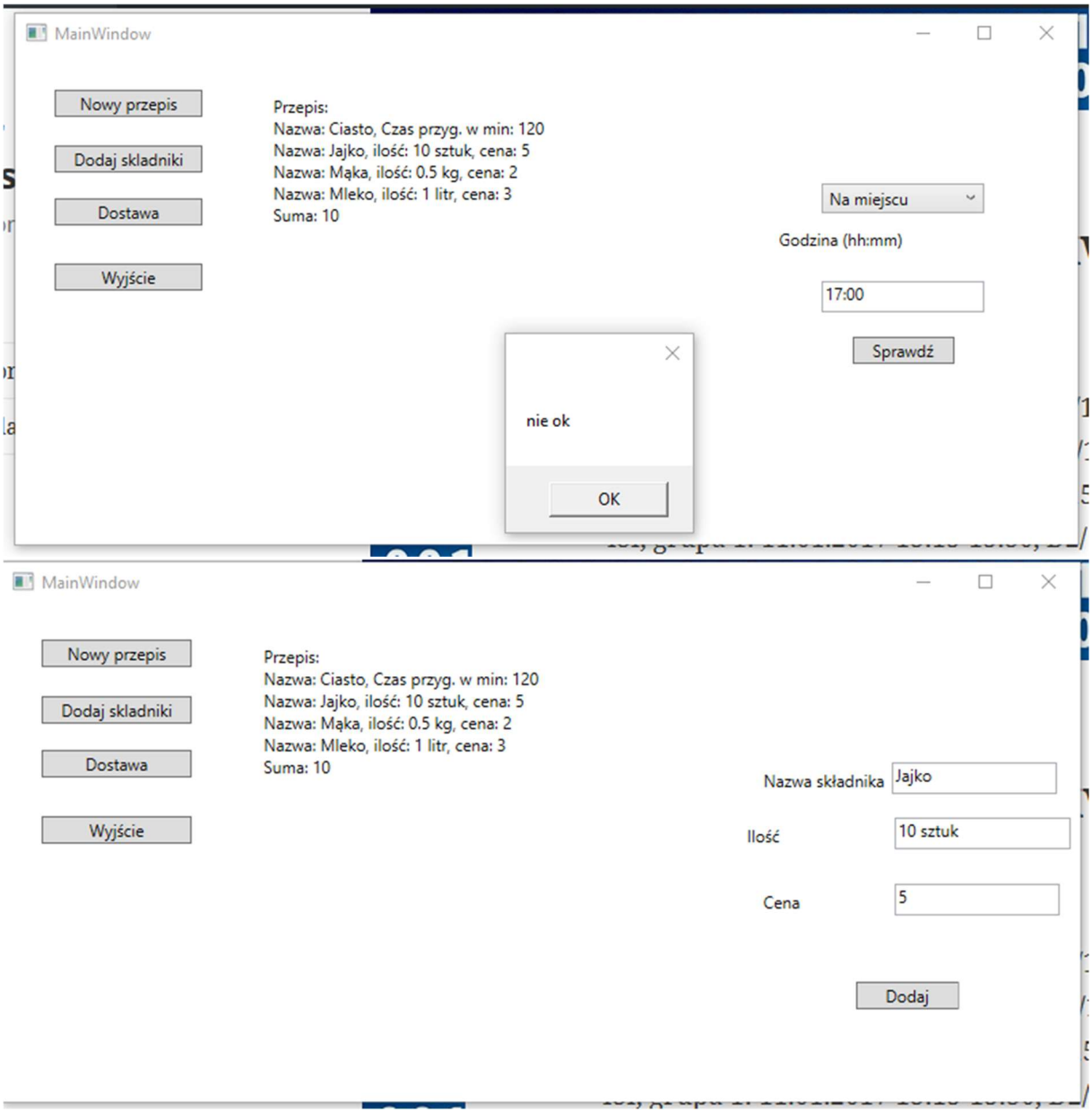
- jeśli to jest spełnione, można wyświetlić kontrolki do dodawania składników, trzeba pobrać nazwę, ilość, cenę, informacje po dodaniu składnika mają się wyświetlać w jakimś TexBoxie lub TexBlocku; należy sprawdzić, że nazwa i ilość nie są puste, a cena liczbą dodatnią.

- kontrolki do ustalenia dostawy mogą się pojawić tylko w wypadku jeśli zostały dodane co najmniej 3 składniki

- użytkownik powinien mieć możliwość wskazania dwóch rzeczy: rodzaju dostawy (na miejscu, na wynos) i wskazania godziny (format dowolny, np. HH:mm), po ich wprowadzeniu i naciśnięcia przycisku – powinna wyświetlić się informacja czy dostawa jest możliwa czy nie (należy wykorzystać tu metodę PoprawnyCzas()).

Przykładowe screeny:





Punktacja:

Polecenie 1 – 1pkt

Polecenie 2 – 1pkt

Polecenie 3 – 1pkt

Polecenie 4 – 1pkt

Polecenie 5 – 1pkt

+1 pkt za poprawne użycie kontrolek

+3 pkt za poprawną logikę aplikację

+1 za brak wyjątków przy konwersji stringów na liczbę/datę

Diagram UML