Hotel

0. Stwórz projekt – aplikacja konsolowa lub WPF (przemyśl wybór, zmiana można być czasochłonna).

1. Stwórz klasę Gosc. W nowostworzonej klasie wykonaj następujące czynności:

a) dodaj dwa prywatne pola imie i nazwisko typu string,

b) dodaj konstruktor parametryczny z dwoma parametrami typu string, parametry należy kolejno ustawić jako pola imie i nazwisko,

c) przesłoń metodę ToString() tak, aby zwracała informacje o gościu np.

Gość, Jan Kowalski.

2. Stwórz klasę Pokoj. W nowo stworzonej klasie wykonaj następujące czynności:

a) dodaj dwa prywatne pola: nrPokoju (typ int) oraz cenaZaDzien (typ double)

b) dodaj konstruktor z dwoma parametrami (kolejno typy parametrów int, double), pobrane parametry należy ustawić jako pola nrPokoju i cenaZaDzien.

c) przesłoń metodę ToString() tak, aby zwracała informacje o pokoju np.

Pokój, nr: 23, cena za dzień: 120.

d) dodaj metodę zwracającą nr pokoju

e) dodaj metodę zwracającą cenę za dzień

3. Do klasy Pokoj podepnij interfejs IComparable lub IComparable<T> i zaimplementuj metodę CompareTo tak, aby sortowała po numerze pokoju (od najmniejszego numeru do największego).

4. Stwórz dwa interfejsy:

a) IHotel – dodaj w nim deklaracje metod typu void:
DodajRezerwacje(string imie, string nazwisko, int nr, double cena)
OdwołajRezerwacje()

b) IData – dodaj w nim deklarację dwóch metod:
UstawDate(DateTime Time) – typ void;
SprawdzDate() – typ zwracany bool.

5. Stwórz klasę Hotel. W nowoutworzonej klasie wykonaj następujące czynności:

a) dodaj prywatne pole rezerwacje będące sortowaną listą z kluczem typu Pokoj i wartościami typu Gosc (SortedList)

b) dodaj prywatne pole zysk typu double i nadaj mu wartość początkową -100;

c) dodaj prywatne pole data typu DateTime

d) podepnij do klasy interfejs IHotel i zaimplementuj następująco metody:

- DodajRezerwacje – dodaj na sortowaną listę nową pozycję o pobranych parametrach, dodatkowo pole zysk powinno zostać powiększone o wartość parametru cena,

- OdwolajRezerwacje – usuwa ostatnią pozycje z sortowanej listy rezerwacje, zysk nie ulega zmianie,

e) podepnij do klasy interfejs IData i zaimplementuj następująco metody:

- UstawDate – ustawia jako pole data pobrany parametr

- SprawdzDate – zraca true jeśli parametr jest większy niż aktualna data pobrana z systemu; zwraca false w przeciwnym wypadku

f) przesłoń metodę ToString() tak, aby zwracała informacje o rezerwacjach np.

Rezerwacje:

Data: 14.01.2017

[Pokoj, nr:12, cena za dzień: 200; Gość: Jan Kowalski]

[Pokoj, nr:13, cena za dzień: 123; Gość: Anna Nowak]

Zysk: 223.

6. Stwórz aplikację WPF lub konsolową do przetestowania stworzonych metod.

Logika aplikacji:

- najpierw użytkownik powinien ustawić datę, użytkownik musi mieć informację w jaki formacie ma być wprowadzona data (np. dd.mm.yyyy lub yyyy/mm/dd, itp.), tu powinien być obsłużony wyjątek przy wprowadzeniu błędnej daty

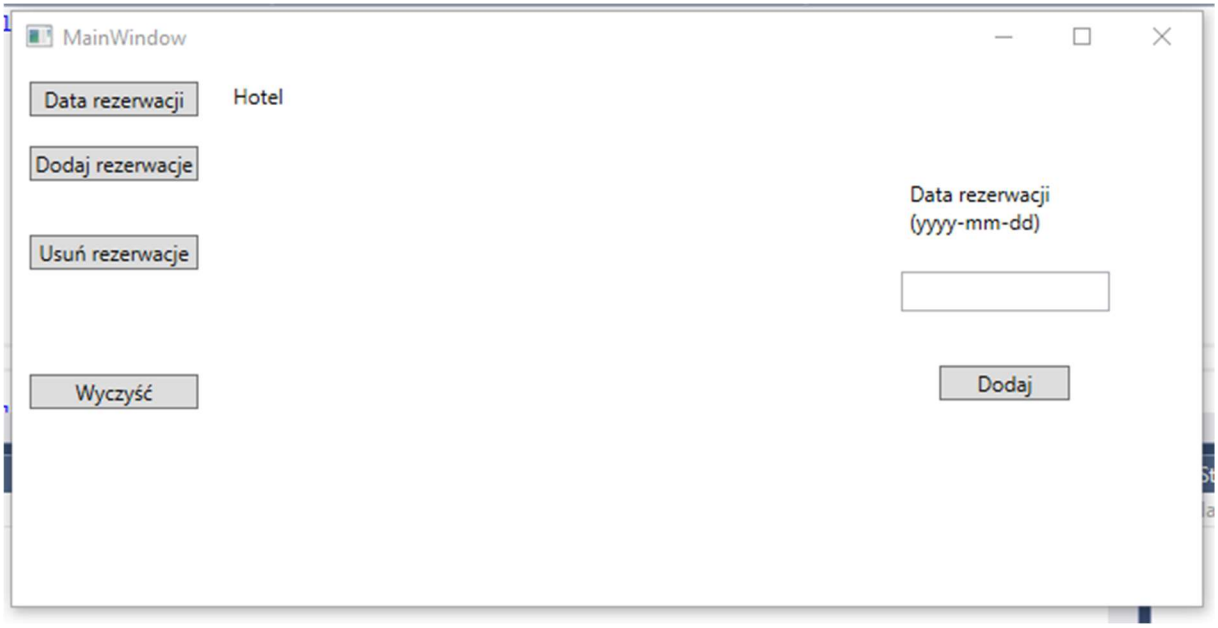
- aby przejść dalej data musi mieć poprawny format i być większa od aktualnej z systemu (wykorzystać metodę SprawdzDate())

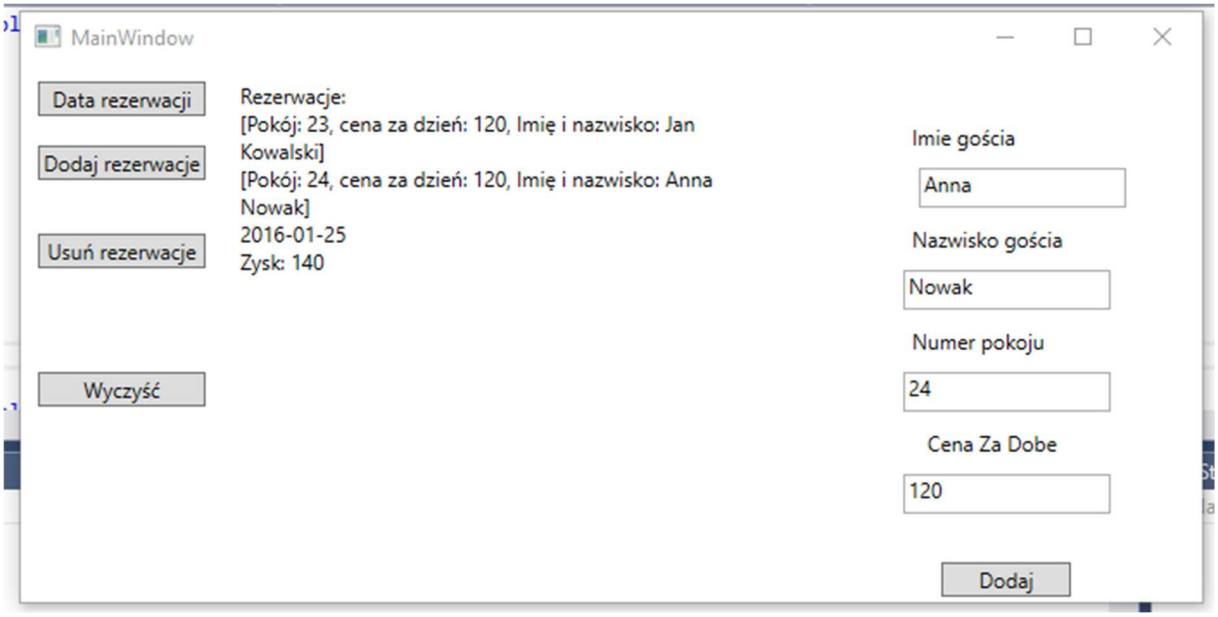
- dodawanie rezerwacji ma jednocześnie pobrać nr pokoju, cenę za dobę, imię i nazwisko gościa, należy sprawdzić, że pola nie są puste, a pola liczbowe zawierają liczby dodatnie

- należy zadbać aby aplikacja nie wyrzucała wyjątku przy dodaniu duplikatu klucza (klucza o takiej samej wartości jak już dodany)

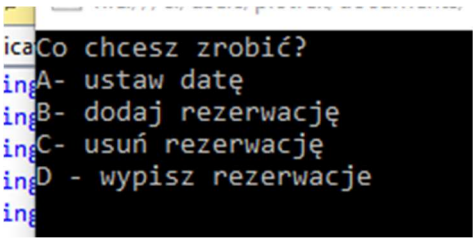
- dodatkowa aplikacja ma mieć możliwość usunięcie ostatniej rezerwacji, tu należy obsłużyć możliwy wyjątek przy usuwaniu z pustej kolekcji

Przykładowy screen WPF:





Przykładowa strona (konsola):



Punktacja:

Polecenie 1 – 1pkt

Polecenie 2 – 1pkt

Polecenie 3 – 1pkt

Polecenie 4 – 1pkt

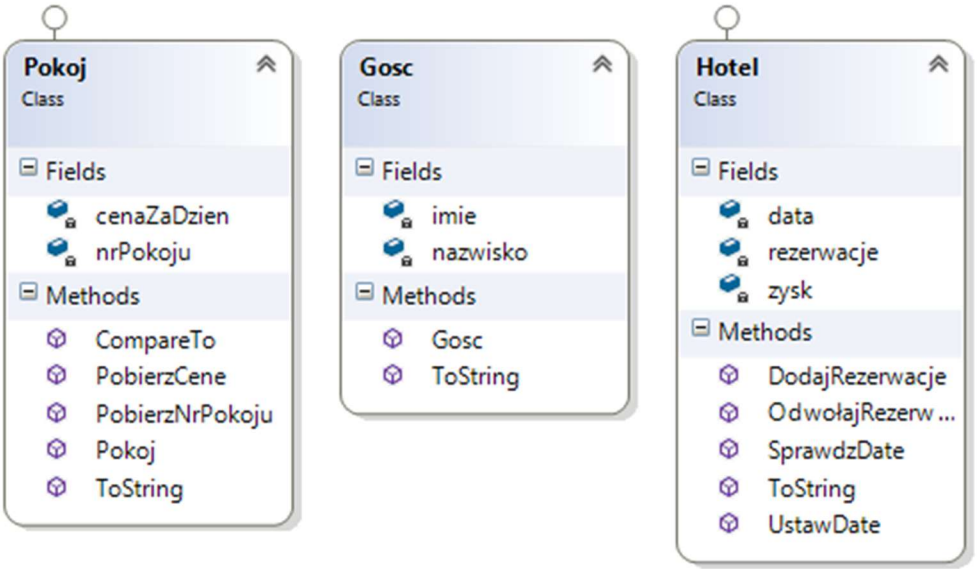
Polecenie 5 – 2 pkt

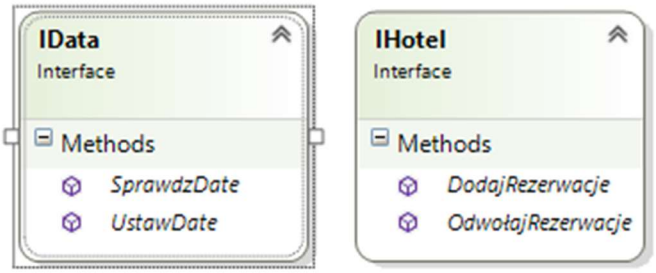
+1pkt za poprawną obsługę kontrolek(wpf)/poprawne przechodzenie po menu(konsola)

+2pkt za obsługę wyjątków

+1pkt za logikę działania aplikacji

Diagram UML:




IData
Interface
Methods
SprawdzDate
UstawDate
IHotel
Interface
Methods
DodajRezerwacje
OdwołajRezerwacje