Podróż

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

1. Stwórz klasę Stacja. W nowoutworzonej klasie wykonaj następujące czynności:

a) dodaj prywatne pola nazwaStacji (typ string) i oplataKlimatyczna(bool)

b) dodaj konstruktor z dwoma parametrami (kolejne typy string, bool),pobrane parametry należy ustawić jako odpowiednie pola w klasie,

c) przesłoń metodę ToString() by zwracała informacje o stacji np.

Stacja: Poznań, opłata klimatyczna: nie.

d) dodaj zwykłą metodę JakaOplata(), bez parametru, zwracającą wartość pola oplataKlimatyczna.

2. Podepnij do klasy Stacja interfejs ICloneable i zaimplementuj mechanizm płytkiej kopii.

3. Stwórz interfejs IStacja i dodaj w nim deklaracje następujących metod (wszystkie void):

DodajStacje(string nazwa, bool oplata)

Powrot()

UsunStacje()

ZapiszPodroz()

4. Stwórz klase Podroz. W nowoutworzonej klasie wykonaj następujące czynności:

a) dodaj pole stacje typu Stack<Stacja> - mod protected

b) dodaj pole koszt typu double (mod. protected) i nadaj mu wartość początkową 77,

c) dodaj pusty konstruktor domyślny

d) dodaj wirtualną metodę KosztPoRabacie (typu double), która zwraca wartość pola koszt pomniejszoną o 12% (ma tylko zwrócić wartość, bez modyfikacji pola koszt)

e) przesłoń metodę ToString() tak, aby wypisywała informacje o stacjach i koszt po rabacie np.

Podróż:

Stacja: Poznań, opłata: nie,

Stacja Wrocław, opłata tak.

Koszt po rabacie: 115.

5. Do klasy Podroz podepnij interfejs IStacja i zaimplementuj metody następująco:

a) DodajStacje – dodaje na stos odpowiedni obiekt o pobranych parametrach, dodatkowo jeśli oplataKlimatyczna ma wartość true należy powiększyć koszt o 8

b) UsunStacje – usuwa ze stosu ostatnią stację, koszt powinien pozostać bez zmian.

c) Powrot – kopiuje pierwszy element na kolejce poprzez mechanizm płytkiej kopii, a potem dodaje go do stosu stacje (podpowiedź: można użyć First() by pobrać pierwszy element ze stosu)

d) ZapiszPodroz() – zapisuje do pliku tekstowego o nazwie „podroze.txt" umieszczonego w folderze aplikacji string, który powstanie po wywołaniu metody ToString() w tej klasie

6. Dodaj klasę klasę LastMinute dziedziczącą z klasy Podroz. W nowoutworzonej klasie wykonaj czynności:

- przesłoń metodę KosztPoRabacie tak, aby zwracała wartość pola koszt pomniejszonego o 33% (pole koszt ma nie być zmieniane)

- przesłoń metodę ToString() tak, aby zwracała string powstały po wywołania metody ToString() uzupełniony o łańcuch „LastMinute".

7. Stwórz aplikację WPF lub konsolową do testowania metod.

Logika aplikacji:

- najpierw należy poprosić użytkownika o wybranie rodzaju podróży (zwykła lub Last Minute)

- następnie powinna być opcja na ustawienie stacji poprzez odpowiednią metodę, analogicznie z usuwaniem i wywołaniem metody Powrot

- dodatkowo powinna być funkcjonalność zapisu aktualnej podróży do pliku tekstowego

Uwaga: aplikacja nie musi obsługiwać wyjątków.

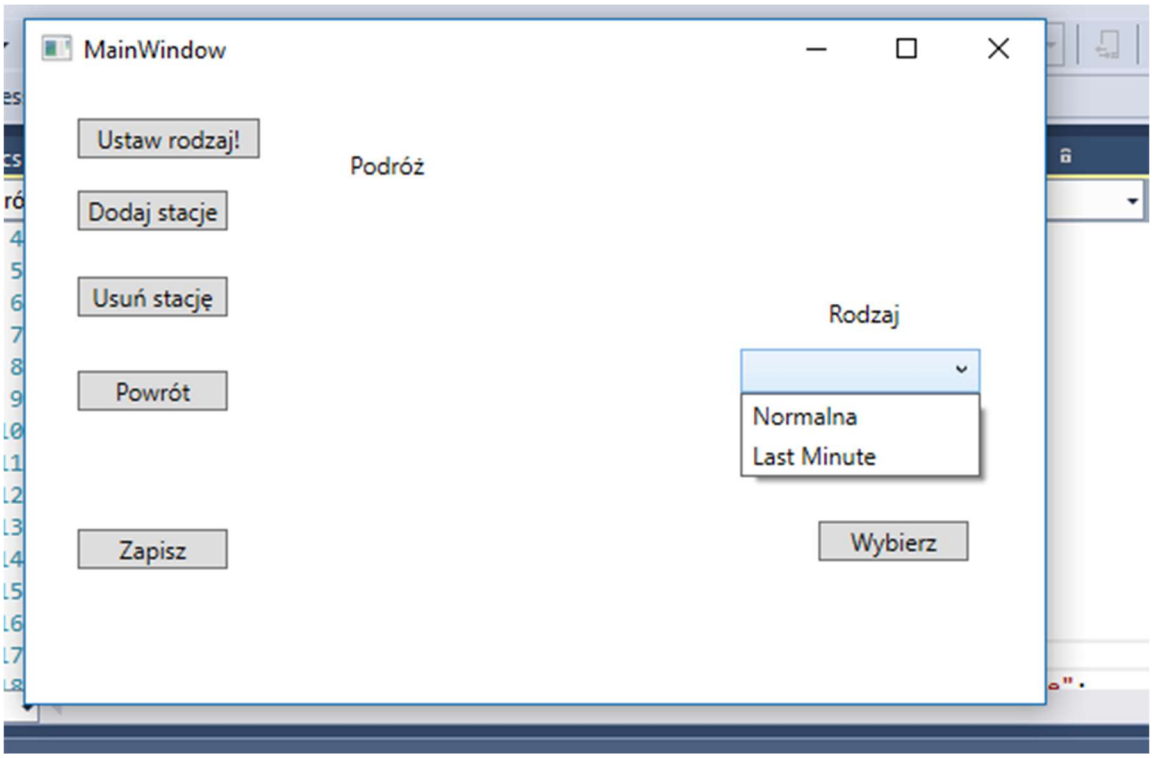
Punktacja:

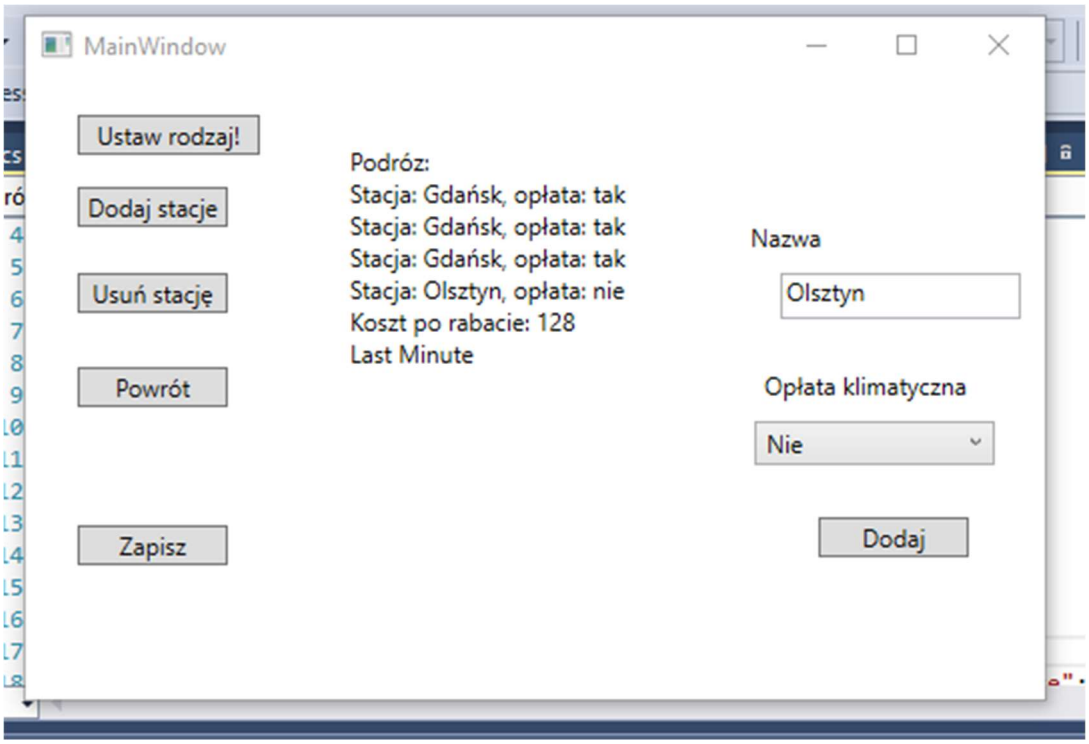
Polecenia 1-6 – po 1pkt

+2pkt za poprawne użycie kontrolek(wpf)/menu aplikacji(konsola)

+2 pkt za logikę aplikacji

Przykładowy screen(wpf):





Przykładowy screen (konsola):

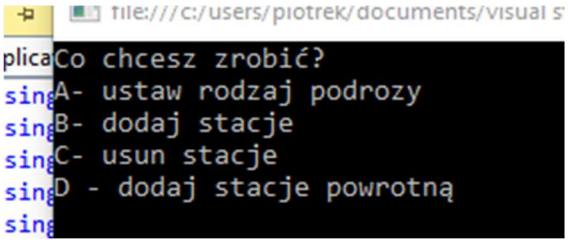


Diagram UML: