Telefon – aplikacja do bilingu.

1. Stwórz abstrakcyjną klasę Usluga.

a) dodaj w niej pole czas typu DateTime oraz cena typu double – oba z modyfikatorem protected;

b) dodaj deklarację abstrakcyjnej metody ObliczCene() typu void, metoda ma być bez parametru,

c) dodaj konstruktor parametryczny z jednym parametrem typu DateTime, parametr ma być ustawiony jako pole czas

2. Dodaj klasę Polaczenie dziedziczącą z klasy Usluga.

a) w klasie potomnej dodaj prywatne pola numer typu string oraz i czasP typu int,

b) w klasie Polaczenie dodaj implementację metody ObliczCene() tak, aby ustawiała za pole cena wartość obliczoną następującą: czasP * 0,29

c) w klasie Polaczenie dodaj konstruktor parametryczny z trzema parametrami (kolejno typy DateTime, string, int), w konstruktorze pobrane parametry należy ustawić jako odpowiednia pola czas, numer, czasP; ponadto w konstruktorze należy wywołać metodę ObliczCena(),

d) przesłoń metodę ToString(), aby zwracała informacje o połączeniu np.

Połączenie: numer 1234, data i godzina rozmowy: 9.01.2017 13:00, długość trwania:2, łączny koszt: 0,58.

3. Stwórz klasę Sms dziedziczącą z klasy Usluga.

a) w klasie potomnej dodaj prywatne pole numer typu string,

b) w klasie Sms dodaj implementację metody ObliczCene() tak, aby podstawiała za pole cena wartość 0,15

c) w klasie Sms dodaj konstruktor parametryczny z dwoma parametrami (kolejne typy DateTime, string), pierwszy parametr ma być ustawione jako pole czas, drugi jako pole numer; w konstruktorze należy także wywołać metodę ObliczCene()

d) przesłoń metodę ToString(), aby zwracała informacje o sms np.

Sms: numer 1234, data i godzina smsa: 9.01.2017 13:00, łączny koszt: 0,15.

4. Stwórz klasę Internet dziedziczącą z klasy Uslua.

a) w klasie potomnej dodaj prywatne pole iloscMB typu int,

b) w klasie Internet dodaj implementacje metody ObliczCene() tak, aby podstawiała za pole cena wartość obliczoną następująco: wartość pola iloscMB dzielimy na 756 i wynik jest zaokrąglony do dwóch miejsc po przecinku,

c) w klasie Internet dodaj konstruktor parametryczny z dwoma parametrami (kolejne typy DateTime, int), pierwszy parametr ma być ustawione jako pole czas, drugi jako pole iloscMB; w konstruktorze należy także wywołać metodę ObliczCene()

d) przesłoń metodę ToString(), aby zwracała informacje o internecie np.

Internet, data i godzina internetu: 9.01.2017 13:00, ilośćMB: 1512, łączny koszt: 2.

5. Stwórz dwa interfejsy:

- IBiling – dodaj w nim deklarację metody bez parametry ZapiszBiling(), typ zwracany void,

- IDodaj – dodaj w nim deklarację trzech metod typu void:

DodajPolaczenie(string numer, int czasP);

DodajSms(string numer);

DodajInternet(int iloscMB);

6. Stwórz klasę Telefon.

a) dodaj w niej prywatne pole biling typu List<Usluga>

b) zaimplementuj odpowiednio metody

- DodajPolaczenie – dodaje do bilingu polaczenie o podanych parametrach, data i godzina pobierana jest aktualna z systemu

- DodajSms – dodaje do bilingu sms o podanym parametrze, data i godzina pobierana jest aktualna z systemu

- DodajInternet – dodaje do bilingu internet o podanym parametrze, data i godzina jest pobierana jako aktualna z systemu

c) przesłoń metodę ToString() tak, aby wypisywała kolejne pozycje z pola biling w nowych wierszach

d) zaimplementuj metodę ZapiszBiling tak, aby zapisywała wynik metody ToString() do bilingu o nazwie biling.txt

e) podepnij do klasy Telefon interfejsu z punktu 5

7. Zaimplementuj interfejs WPF.

Logika działania aplikacji:

- są trzy kontrolki na dodawanie odpowiednie połączeń, smsów, internetu

- przed dodaniem na listę należy sprawdzić czy pola nie są puste, a liczby są dodatnie

- aplikacja ma nie wyrzucają wyjątków przy konwersji/parsowaniu i zapisywaniu plików tekstowych

Przykładowy screen:

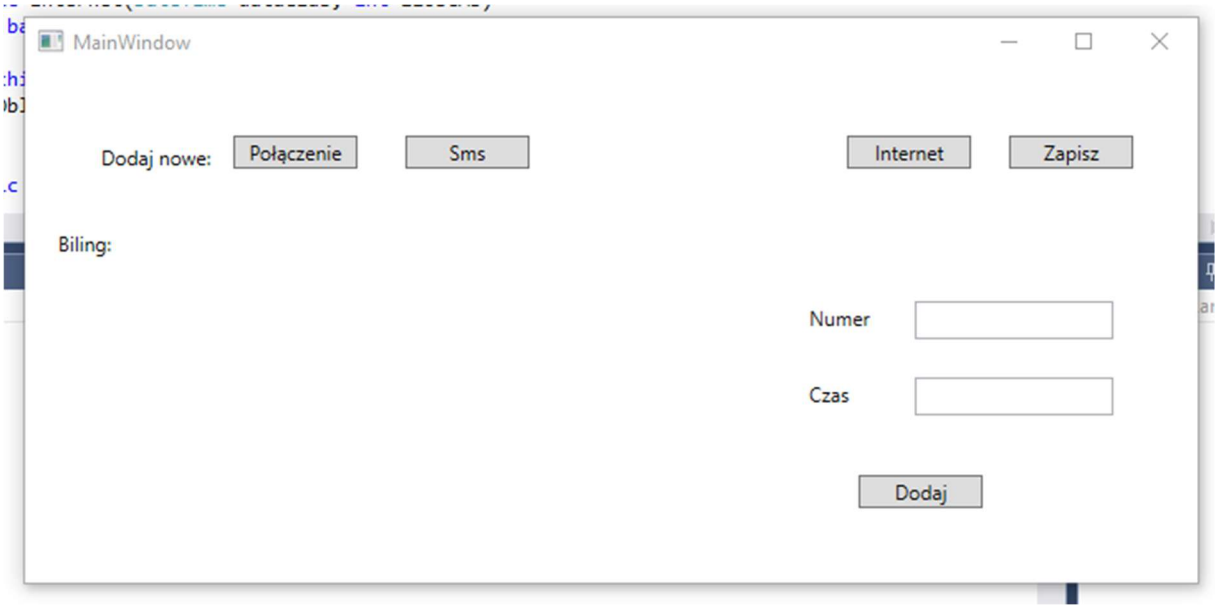
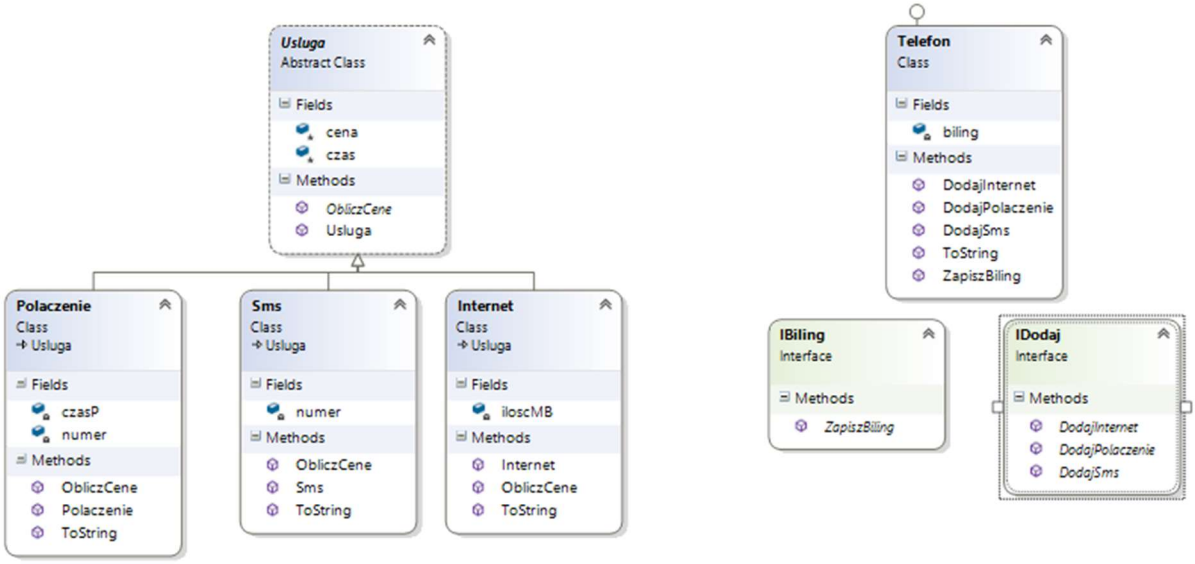


Diagram UML



Polecenie 1 – 1pkt

Polecenie 2 – 1pkt

Polecenie 3 – 1 pkt

Polecenie 4 – 1pkt

Polecenie 5 – 1 pkt

Polecenie 6 – 1pkt

Polecenie 7- 1pkt

+2 pkt za poprawna obsługę wyjątków

+1 pkt za zachowanie logiki aplikacji