Klasy – podstawy

Zadanie 9.1 Klasa `Konto`

Krok 1 Utworzyć klasę `Konto` zawierającą następujące pola: `imie` (typ `string`), `nazwisko` (typ `string`), `numer` (typ `long`) i `saldo` (typ `decimal`).

Krok 2 Dla klasy Konto napisać konstruktor, przyjmujący dane właściciela jako argument.

Krok 3 Dla klasy Konto napisać następujące metody:

- `void Wplac (decimal kwota)` – metoda przyjmuje jeden argument (kwotę wpłaty) i zwiększa saldo konta o podaną kwotę.
- `bool MoznaWyplacic(decimal kwota)` – metoda przyjmuje jeden argument (kwotę do wpłaty) i sprawdza, czy z konta można wypłacić żądana kwotę. Jeśli tak – zwraca wartość `true`, w przeciwnym przypadku – `false`.
- `void Wyplac(decimal kwota)`– metoda przyjmuje jeden argument (kwotę do wypłaty), sprawdza, czy można wypłacić. Jeśli tak – zmiejsza saldo konta o podaną kwotę. W przeciwnym przypadku wypisuje komunikat „Operacja wypłaty xxx nie może być wykonana”.
- `void Przelej(Konto konto, decimal kwota)`– metoda przyjmuje dwa argumenty: obiekt klasy `Konto` (na który należy dokonać przelewu) i kwotę przelewu. Metoda sprawdza, czy można wypłacić żądaną kwotę, jeśli tak – zmiejsza saldo na koncie źródłowym i zwieksza na koncie docelowym. Jeśli nie – wypisuje komunikat „Przelew w kwocie xxx nie może być wykonany”.
- `void StanKonta()`– metoda wypisuje podstawowe dane o koncie, takie jak: numer konta, wlasciciel, saldo.

Krok 4 Zaimplementować mechanizm automatycznego nadawania numerów kont.
Wskazówka: Wykorzystać zmienną statyczną do przechowywania aktualnie nadanego (lub nadawanego) numeru konta

Krok 5 Przetestować klasę `Konto` w następujący sposób:

- utworzyć dwa konta
- wpłacić na pierwsze konto kwotę 1000 zł, a na drugie 500 zł
- wypłacić z pierwszego konta kwotę 300 zł
- przelać z pierwszego konta na drugie kwotę 500 zł
- wykonać próbę wypłaty 400 zł z pierwszego konta

Po każdej z powyższych operacji wypisać stan konta.

Zadanie 9.2 Klasa `Bank`

Krok 1 Utworzyć klasę `Bank` o następujących polach: `nazwa` (typ `string`), `tablicaKont` (10-cio elementowa tablica obiektów typu `Konto`)

Krok 2 Napisać konstruktor, który przyjmuje nazwę banku jako argument. Konstruktor winien utworzyć tablice kont.

Krok 3 Dla klasy Bank napisać następujące metody:

- `Konto ZalozKonto(string imie, string nazwisko)` – metoda tworzy obiekt klasy `Konto` i zapisuje go do tablicy kont. Metoda powinna zadbać, by nie przekroczyć rozmiaru tablicy kont.
- `void ListaKont()` – metoda wyświetla podstawowe informacje o każdym z kont (wykorzystując metode `StanKonta`() z klasy `Konto`).
- `Konto ZnajdzKonto(long numer)` – metoda znajduje w `tablicaKont` konto o podanym numerze. Jeśli takiego konta nie ma, wypisuje odpowiedni komunikat „Konto o numerze xxx nie istnieje”.

Krok 3 Do klasy Konto dodać metodę akcesorową do odczytywania numeru konta.

```
public long Numer
{
      get { return _numer; }
}
```

Krok 4 Do klasy Konto dodać następującą metodę:

- `void Przelej(Bank bank, long numer, decimal kwota)` – Metoda przyjmuje trzy argumenty: obiekt klasy `Bank`, numer konta na ktory należy dokonać przelewu i kwotę przelewu. Metoda powinna znaleść konto o podanym numerze i dokonać przelewu.

Krok 5 Przetestować klasy `Bank` i `Konto` w następujący sposób:

- utworzyć obiekt klasy `Bank` i dwa obiekty klasy `Konto`
- wpłacić na pierwsze konto kwotę 1000 zł, a na drugie 500 zł
- przelać z pierwszego konta na drugie kwotę 500 zł, korzystając z nowej metody `Przelej`

Po każdej operacji wypisać stan konta (korzystając z metody `ListaKont`)

Zadanie 9.3 Klasa `Komorka` (telefon komórkowy)

Krok 1 Utworzyć klasę Komorka o następujących polach: `numerTelefonu` (typ `long`) i `odebranySMS` (typ `string`).

Krok 2 Napisać konstruktor przyjmujący numer telefonu jako argument.

Krok 3 Napisać metodę akcesorowa do odczytywania numeru telefonu.

Krok 4 Napisać następujące metody:

- `void OdbierzSMS(string sms)` – metoda przyjmuje jeden argument (`sms`) i wstawia go do pola `odebranySMS`.
- `void CzytajSMS()` – metoda wypisuje treść smsa (zawartość pola `odebranySMS`).
- `void WyslijSMS(Komorka komorka, string sms)` – metoda przyjmuje dwa argumenty: obiekt klasy `Komorka` (na który należy przesłać sms) i `sms` – zawierający treść smsa. Metoda powinna wypisać komunikat „Wysyłanie sms od xxx do yyy o treści: zzz” i wywołać metodę `OdbierzSMS` na obiekcie docelowym.

Krok 5 Przetestować klasę `Komorka` w następujący sposób:

- Utworzyć dwa obiekty klasy `Komorka`
- Wysłać jeden sms z pierwszej komórki na drugą
- Na drugiej komórce odczytać sms
- Wysłać dwa smsy z pierwszej komórki na drugą
- Odczytać sms na drugiej komórce

Zadanie 9.4 Klasa `OperatorGSM`

Krok 1 Utworzyć klasę `OperatorGSM` o następujących polach: `tablicaKomorek` (10-cio elementowa tablica obiektów typu `Komorka`)

Krok 2 Napisać bezparametrowy konstruktor, którego zadaniem jest utworzenie 10-elementowej tablicy obiektów typu `Komorka`.

Krok 3 Dla klasy `OperatorGSM` napisać następujące metody:

- `void Rejestruj(Komorka komorka)` – metoda wstawia do `tablicaKomorek` obiekt `komorka`.
- `Komorka Wyszukaj(long numer)` – metoda wyszukuje w `tablicaKomorek` obiekt klasy Komorka o numerze telefonu równym zmiennej `numer`. Jeśli komórka o podanym numerze nie istnieje metoda powinna wypisać komunikat „Abonent o numerze xxx nie istnieje bądź nie jest zarejestrowany w sieci”

Krok 4 Do klasy Komorka dopisać metodę:

- `void WyslijSMS(OperatorGSM operator, long numer, string sms)` – metoda znajduje (korzystając z metody `Wyszukaj`) obiekt klasy `Komorka` o podanym numerze i wysyła sms (korzystając z metody `OdbierzSMS`)

Krok 5 Przetestować klasy `Komorka` i `OperatorGSM` w następujący sposób:

- Utworzyć obiekt klasy `OperatorGSM` dwa obiekty klasy `Komorka`
- Wysłać jeden sms z pierwszej komórki na drugą, korzystając z nowej metody `WyslijSMS`
- Na drugiej komórce odczytać sms
- Wysłać sms z pierwszej komórki na nieznany numer