Laborator 6– Crearea și utilizarea claselor

Obiective

După completarea acestui laborator veți dobândi următoarele cunoștințe:

- Crearea claselor și instanțierea obiectelor
- Utilizarea datelor și metodelor non-statice
- Utilizarea datelor și metodelor statice

Condiţii prealabile

Înainte de a realiza acest laborator trebuie să fiţi familiarizaţi cu următoarele concepte:

- Crearea metodelor în C#
- Transmiterea argumentelor ca parametri ai metodelor în C#

➢ Exerciţiul 1 – Crearea și utilizarea claselor

În acest exerciţiu, veţi porni de la structura **BankAccount,** pe care ați creat-o într-o ședință precedentă și o veți transforma într-o clasă. Veți declara datele sale ca membrii privați, dar veți pune la dispoziție utilizatorului posibilitatea de a le accesa prin metode publice, non-statice. Veți testa clasa prin crearea unui obiect ce aparține acesteia, popularea câmpurilor cu valori și afișarea sa la final.

Conversia de la structură la clasă

- Rulați suportul pentru laboratorul 6. Programul crează o structură pe tip **BankAccount** și afișează valorile acesteia.
- Modificați cuvântul cheie **struct** cu alt cuvânt cheie astel încât să obțineți o clasă. Executați din nou programul.
- Veți obține o eroare care spune că variabila **account** nu este asignată. Acest exemplu pune foarte clar în evidență faptul că o structură este de tip valoare (păstrată pe stivă și toate câmpurile vor fi inițializate cu zero), pe când o clasă este de tip referință, ceea ce presupune o altfel de abordare. Prin declararea variabilei **account** am creat doar o referință către un obiect care nu există încă. Pentru a crea un obiect din acea clasă folosiți cuvântul cheie **new**.
  Salvați și compilați programul.

Încapsularea clasei BankAccount

- Toate câmpurile membre clasei **BankAccount** sunt publice. Modificați determinantul de acces în **private**.
- Rulați programul. Veți obține o eroare cauzată de faptul că prin modificarea pe care ați făcut-o mai sus ați impus ca aceste câmpuri să fie private oricărei încercări de accesare din afara clasei (doar metodele din interiorul clasei le pot accesa).
- Va trebui să scrieți o metodă publică prin care să se realizeze popularea acestor câmpuri. Semnătura acestei metode va fi următoarea:

```
public void Populate(long nr, decimal amount, AccountType tip) {}
```

Observaţie!
Nu uitați că membrii privați ai unei clasei vor fi întotdeauna la sfârșitul clasei. În situația de fața veți avea mai întâi declarate metodele (care sunt publice), după care urmează câmpurile (care sunt private)

- Comentați cele 3 linii din **Main()** care realizau asignarea de valori câmpurilor (sursa erorii de mai sus) și utilizați metoda **Populate** în acest scop.
- În continuare dacă veți încerca să rulați programul veți obține aceeași eroare. Motivul este ca la afișare încercați să accesați direct câmpurile clasei. Prin urmare va fi nevoie de încă 3 metode publice și non-statice care să returneze valoarea fiecarui membru al clasei.

```
Public long AccNumber() {}
Public decimal Amount() {}
Public AccountType ReturnType() {}
```

- Salvați și compilați programul.

➢ Exerciţiul 2 – Generarea numerelor de cont

În acest exerciţiu veţi schimba clasa **BankAccount** de la **Exercițiul 1** astfel încât să genereze numere de cont unice. Veți realiza acest lucru prin utilizarea unei variabile statice declarată în clasă și a unei metode care incrementează și returnează valoarea acestei variabile.

Asigurarea unicității fiecărui număr de cont

- Creați o variabilă statică și privată de tip **long** numită **succNrCont**.
- Adăugați o metodă (statică și publică, fără niciun argument) care realizează următoarele operatii: incrementează valoarea lui **succNrCont** după care returnează această valoare.

```
public static long NextNumber( )
```

- Utilizați această metodă în locul citirii de la tastatură a numărului de cont.
- Valoarea default de inițializare a unei variabile statice este 0 (sau valoarea nulă corespunzătoare tipului respectiv). Modificați programul astfel încât valoarea de plecare a variabilei **succNrCont** să fie **123**.
- Salvați și compilați programul.

Încapsularea clasei BankAccount

- Modificați metoda **Populate()** astfel încât să primească doar doi parametrii (**amount** și **accType**). În interiorul metodei asignați o valoare câmpului **accNumber** utilizând metoda **NextNumber()**.
- Modificați metoda **NextNumber()** în privată.
- Actualizați **Main()**-ul astfel încât să țineți cont de modificările făcute.
- Salvați și compilați programul.

➢ Exercițiul 3 – Adăugarea de metode publice

În acest exercițiu veți adăuga două metode clasei **BankAccount ()** numite **Withdraw** și **Deposit**.

Withdraw va primi un parametru de tip **decimal** si va deduce suma de bani pe baza soldului existent. Însă, va verifica înainte că există suficiente fonduri , întrucât nu este permisă descoperirea unui cont. Metoda va returna un **bool** prin care se specifică dacă retragerea a fost corectă sau nu.

Deposit va primi de asemenea un parametru de tip **decimal** a cărui valoare va fi adăugată la soldul existent în cont. Va avea ca valoare de return noul sold al contului.

Metoda Deposit

- Adăugați o metodă publică non-statică numită **Deposit** la clasa **BankAccount**. Aceasta va primi un parametru de tip decimal și va returna noua valoare a soldului. Signatura metodei va fi următoarea:

```
Public decimal Deposit(decimal amount){}
```

- Adăugați o metodă publică statică numită **TestDeposit()** clasei **BankAccount**. Această metodă va fi de tip **void** și va primi ca parametru un obiect de tip **BankAccount**. În corpul metodei se vor face următoarele operații:
  - afișarea unui mesaj la consolă prin care se cere utilizatorului să introducă o sumă de bani (de tip **decimal**);
  - citirea acestei sume într-o variabilă de tip **decimal**;
  - recalcularea contului utilizând metoda **Deposit**;

- Signatura metodei va fi:

```
public static void TestDeposit(BankAccount acc)
```

- Testați metodele create în **Main()**.

Metoda Withdraw

- Adăugați o metodă publică non-statică numită **Withdraw** la clasa **BankAccount**. Aceasta va primi un parametru de tip **decimal** și va recalcula soldul pe baza celui existent. În cazul în care are loc o descoperire de cont (suma retrasă este mai mare decât soldul) metoda va returna **false**; altfel va returna **true**.

```
public bool Withdraw(decimal suma)
```

- Adăugați o metodă publică statică numită **TestWithdraw()** clasei **BankAccount**. Această metodă va fi de tip **void** și va primi ca parametru un obiect de tip **BankAccount**. În corpul metodei se vor face următoarele operații:
  - afişarea unui mesaj la consolă prin care se cere utilizatorului să introducă o sumă de bani (de tip **decimal**);
  - citirea acestei sume într-o variabilă de tip **decimal**;
  - recalcularea contului utilizând metoda **Withdraw**. În cazul în care are loc descoperire de sold va fi afişat un mesaj utilizatorului prin care va fi anunţat de aceasta.
- Signatura metodei va fi:

```
public static void TestWithdraw(BankAccount acc)
```

- Testați aceste subpuncte în **Main()**.