Laborator – Proprietăți și indecși

Obiective

După completarea acestui laborator veți dobândi următoarele cunoștințe:

- Crearea proprietăților pentru încapsularea datelor într-o clasă
- Definirea indecșilor pentru accesarea claselor la fel ca la vectori

Condiţii prealabile

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

- Crearea și utilizarea claselor
- Utilizarea vectorilor și a colecțiilor

➢ Exerciţiul 1 – Crearea de proprietăți pentru clasa BankAccount

În acest exerciţiu veți șterge metodele care returnau valorile celor trei câmpuri ale unui cont bancar (**amount**, **accType**, **accNumber**) și le veți înlocui cu proprietăți *read-only*. Veți adăuga încă un câmp de tip **string**, respectiv o proprietate pentru acesta, care va păstra numele celui care deține contul.

Modificarea metodelor în proprietăți

- Deschideți suportul laboratorului și înlocuiți metodele **AccNumber**, **Amount**, **ReturnType** cu proprietăți (păstrând aceeași denumire pentru acestea) care au numai accesor **get**.
- Înlocuiți peste tot în aplicație apelul acestor metode cu apelul proprietăților noi create. Exemplu:

```
long accNumber = cont.AccNumber();  //va deveni
long accNumber = cont.AccNumber;
```

- Salvați și compilați programul.
- Rulați pentru a verifica corectitudinea modificărilor.

Adăugarea unui nou câmp

- Adăugați un nou câmp clasei **BankAccount** de tip **string**, privat, nestatic și a cărui denumire este **accountOwner**.
- Creați o proprietate numită **Owner**, de tip **string** care va avea cei doi accesori:

  - **set** pentru a popula câmpul **accountOwner**;
  - **get** pentru a întoarce valoarea câmpului **accountOwner;**

- Modificați metoda **ToString** a lui Object (suprascriere) pentru a returna valorile celor trei câmpuri plus numele proprietarului.
- Salvați și compilați programul.
- Verificați acest subpunct prin afișarea informațiilor despre un cont utilizând metoda **ToString**.

➢ Exercițiul 2 – Modificarea clasei BankTransaction

În acest exercițiu veți schimba clasa **BankTransaction**, pe care ați creat-o într-unul dintre laboratoarele trecute. După cum vă amintiți clasa **BankTransaction** a fost creată pentru a păstra informații despre tranzacțiile financiare relative la un anumit cont.

Veți înlocui metodele **Date** și **Amount** cu două proprietăți *read-only*.

Modificarea metodelor în proprietăți

- Deschideți clasa **BankTransaction** și înlocuiți metodele **Date** și **Amount** cu proprietăți (păstrând aceeași denumire pentru acestea) care au numai accesor **get**. (sunt *read-only*)
- Înlocuiți peste tot în aplicație apelul acestor metode cu apelul proprietăților noi create.
- Salvați și compilați programul.
- Rulați pentru a verifica corectitudinea modificărilor.
- În metoda **Main** creați două conturi pentru care realizați mai multe tranzacții (retragere și depozit). Printați pentru fiecare cont istoricul de tranzacții.

➢ Exercițiul 3 – Crearea și utilizarea unui index

În acest exercițiu veți adăuga un index clasei **BankAccount** pentru a accesa obiectele de tip **BankTransaction** păstrate într-o structură internă.
Tranzacțiile relative la un anumit cont sunt păstrate într-o coada **System.Collection.Queue** aflată în clasa **BankAccount**. Veți defini un index pentru clasa BankAccount care întoarce din această coadă obiectul de tip **BankTransaction** specificat printr-o anumită poziție. Veți utiliza metoda **GetEnumerator** a clasei **Queue** în acest exercițiu.

Declararea unui index read-only

- În clasa **BankAccount**, declarați un index public care returnează un obiect de tip **BankTransaction** și primește un singur parametru de tip **int** numit **ind**.

```
public BankTransaction this[int ind]
{...}
```

- Adăugați un accesor **get** în corpul indexului și implementați-l conform explicațiilor de mai jos.
- Tranzacțiile pentru clasa **BankAccount** sunt păstrate într-o coadă numită **transactionQueue** (**Queue**). Clasa **Queue** nu are un index implementat, prin urmare pentru a ajunge la un anumit element trebuie manual să iterați prin coadă. Pentru a face acest lucru:

  - Verificați dacă parametrul **ind** este în limitele corecte (sa nu fie mai mic ca 0 sau mai mare decât numărul de elemente existent în coadă). Pentru a afla numărul de elemente din coadă veți utiliza metoda **Count**, apelată pentru **transactionQueue**.
  - Dacă indexul, **ind**, nu se află în intervalul corespunzător returnați **null**.
  - Altfel, declarați o variabilă de tip **IEnumerator** și inițializați-o utilizând metoda **GetEnumerator**, apelată pentru obiectul **transactionQueue**.
  - Metoda **MoveNext** a enumeratorului declarat mai sus vă va ajuta să vă deplasați la poziția următoare în coadă. Cu ajutorul acestei metodei parcurgeți elementele până ajungeți la poziția **ind** din **transactionQueue**. Veți returna această valoare utilizând metoda **Current** apelată pentru enumerator.

- Modificați metoda **Write** din clasa **BankAccount** astfel încât să utilizeze indexul creat.
- Verificați acest indexul în metoda **Main**:

  - Creați un obiect **BankAccount** numit **account**;
  - Realizați mai multe tranzacții bancare asupra acestui cont;
  - Afișați la sfârșit informațiile despre cont utilizând metoda **Write**;

- Salvați și compilați aplicația.