Laborator 1 – Tipuri și contracte standard

Obiective

După completarea acestui laborator veți dobândi următoarele cunoștințe:

- Utilizarea tipurilor *nullable*
- Definirea tipurilor generice
- Implementarea interfețelor standard .NET Framework
- Aruncarea și prinderea excepțiilor
- Ridicarea și manipularea evenimentelor

Scenariul de laborator

Ești dezvoltator pentru o organizație care implementează aplicații .NET Framework. Ai fost rugat să realizezi o aplicație Windows Forms numită AccountManager în care vei include diferite concepte de programare.

Exercițiul 1- Utilizarea tipurilor Nullable

În acest exercițiu vă veți familiariza cu aplicația AccountManager, după care veți modifica aplicația astfel încât să accepte tipuri nullable pentru un anumit tip valoarea.

- Modificați variabila **_mostRecentTransaction** din clasa **Account** astfel încât să accepte și valoarea null
- Inițializați această valoare în constructor cu **null**
- Modificați metoda ToString astfel: în cazul în care _mostRecent Transaction este nenulă returnați un string care conține numele contului, soldul și valoarea tranzacției cea mai recentă; altfel returnați mesajul „nicio tranzacție recentă”
- Rulați și testați aplicația

Exercițiul 2- Utilizarea tipurilor generice

În acest exercițiu veți modifiva structura **Transaction** astfel încât să devină un tip generic. În acest moment structura reprezintă **amount** prin **double**, însă după modificare aceasta va fi reprezentată de un tip generic **T** (dat ca parametru de tip structurii). Astfel clientul va putea specifica tipul pe care îl preferă pentru a reprezenta suma (double, float, int).

- Definiți structura **Transaction** ca un tip generic
- Utilizați noua structură peste tot în codul client specificând un parametru de tip (double, int, float)

- Definiți o constrângere asupra tipului T conform căreia, tipul trebuie să implementeze interfața Icomparable
- Rulați și testați aplicația

* Exercițiul 3 – Implementarea interfețelor standard .NET Framework

În acest exercițiu veți implementa interfețele standard IComparable și IFormattabble pentru tipul **Transaction**. Interfața IFormattable va permite codului client să specifice unul dintre următoarele formate:

- „{0} ” – Niciun format specificat - se cer toate informațiile legate de tranzacție
- „{0:d} ” – Se va returna doar amprenta de timp a tranzacției
- „{0:a} ” – Se va returna doar suma tranzacției

- Implementați metoda IFormattable conform descrierii de mai sus. **Atenție!** nu ștergeți metoda ToString deja existentă
- Pentru a verifica modificările, localizați în clasa **Main.cs** metoda **DetailsButton_Click** și folosiți în apelul metodei **String.Format** diversele formate create de mai sus.
- Implementați interfața **IComparable**: veți compara obiectul curent și obiectul primit ca parametru după **amount**.
- În clasa **Account** localizați proprietatea **AllTransactions**. Sortați tranzacțiile utlizând metoda **Array.Sort**
- Rulați și testați aplicația

Exercițiul 4 – Aruncarea și prinderea excepțiilor

- Declarați o clasă **TransactionException** (într-un fișier separat) pentru aplicația de față care va fi folosită în situații de eroare. Clasa moștenește **Exception** și are 3 constructori după cum urmează: un constructor default, un constructor care primește un string ca parametru numit message, și altul care primește un parametru string și încă o excepție
- În clasa **Account**, găsiți metoda **Withdraw**. La începutul acestei metode verificați daca balance este negativ caz în care aruncați o excepție de tipul descris mai sus.
- În **MainForm.cs** localizați metoda **WithdrawButton_Click** și modificați-o astfel încât să prinda excepțiile care ar putea să apară (vor fi doua filtre : unul pentru **TransactionException**, altul generic). În blocul catch pentru excepția specifică afișați un mesaj utilizând metoda **MessageBox.Show**

• Exerciţiul 5 – Ridicarea şi manipularea evenimentelor

În acest exerciţiu veţi adăuga în clasa **Account** două evenimente: unul care se declanșează dacă s-a depus o sumă de bani mai mare decât 1000, iar altul dacă s-a retras o sumă de bani mai mare de 1000.

- Creați o clasă **TransactionEventArgs** care moștenește clasa **EventArgs** și are următorii membri:
  - Un câmp privat *double* numit **_amount**
  - Un constructor care primește un parametru de tip *double* și inițializează variabila **_amount**
  - O proprietate *read-only* numită **Amount** care întoarce valoarea câmpului
- Definiți 2 evenimente în clasa **Account** numite **LargeDebit** și **Large Credit**. Specificați ca tipul ambelor evenimente să fie **EventHandler<TransactionEventArgs>**
- În metoda **Deposit**, la sfârșitul codului verificați dacă suma este mai mare de 1000 caz în care ridicați evenimentul **LargeCredit**.
- În mod analog procedați pentru metoda **Withdraw**, cu evenimentul **LargeDebit**
- În **MainForm.cs** ștergeți codul cuprins între **#if…#endif** inclusiv aceste directive și realizați următoarele metode handler:
  - **OnLargeCredit** – în interiorul metodei veți scrie un mesaj într-un fișier text care va indica faptul că s-a depus o sumă semnificativă de bani
  - **OnLargeDebit** – în interiorul metodei veți scrie un mesaj în același fișier ca ma sus care va indica faptul că s-a extras o sumă semnificativă de bani
  - Pentru scriere în fișier puteți folosi metoda **File.AppendAllText**
- În metoda **MainForm_Load** din **MainForm.cs** specificați cele două metode handle pentru tratarea evenimentelor obiectului **_theAccount**.

** Exercitiile marcate cu * sunt optionale.*