Laborator – Delegați și Evenimente

Obiective

După completarea acestui laborator veți dobândi următoarele cunoștințe:

- Publicarea evenimentelor
- Înregistrarea pentru un eveniment
- Trimiterea de parametri evenimentelor

Condiţii prealabile

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

- Crearea claselor în C#
- Definirea constructorilor
- Compilarea în C#

➢ Exerciţiul 1- Definirea de operatori pentru clasa BankAccount

Acest exercițiu extinde în continuare exemplul legat de sistemul bancar. Acum, veți crea o clasă numită **Audit**. Scopul acestei clase este de a înregistra schimbările petrecute cu soldurile conturilor într-un fișier text. Această operație se va executa la momentul lansării unui eveniment de către clasa **BankAccount**.

Veți folosi metodele **Deposit** și **Withdraw** ale clasei **BankAccount** pentru a declanșa evenimentul, numit **Auditing**, la care s-a înregistrat un obiect de tipul **Audit**.
Evenimentul **Auditing** va primi un parametru ce reprezintă un obiect de tipul **BankTransaction**, ce conține toate detaliile tranzacției.

Definirea clasei de parametri ai evenimentului

În acest exercițiu, evenimentului ce este aruncat îi este transmis un obiect de tipul **BankTransaction** ca parametru. Parametrii evenimentelor trebuie să fie derivați din clasa **EventArgs**, de aceea o nouă clasă trebuie creată

- Adăugați o nouă clasă proiectului, numită **AuditEventArgs**, ce derivează **System**.**EventArgs**
- Creați o variabilă readonly de tipul **BankTransaction** numită **transData**; inițializați-o la **null**
- Modificați constructorul implicit la un constructor ce primește un parametru de tipul **BankTransaction** și setează valoarea variabilei **transData** la această valoare
- Scrieți o metodă publică **getTransaction** ce întoarce valoarea variabilei interne transData
- Salvați și compilați proiectul. Rezolvați eventualele erori ce apar.

Pentru a defini clasa Audit

- Adăugați o nouă clasă proiectului, **Audit**. Această clasă se va înregistra la evenimentul **Auditing** și va scrie tranzacțiile într-un fișier
- Adăugați o directivă **using** pentru **System.IO**
- Adăugați o variabilă privată clasei **Audit** de tip **string**, **fileName**
- Adăugați o vatiabilă privată clasei **Audit** de tip **StreamWriter** numită **auditFile**

Notă: clasa **StreamWriter** vă permite să scrieți date într-un fișier. Ați folosit deja **StreamReader** pentru citirea unui fișier în laboratorul 6. În acest exercițiu, veți folosi metoda **AppendText** a clasei **StreamWriter.** Această metodă deschide un fișier pentru adăugarea de text în el. Va scrie datele la finalul fișierului.

- Modificați constructorul implicit al clasei **Audit** la unul ce primește ca parametru un **string**,numele fișierului în care se va face scrierea.
    - Setați variabila **fileName** la valoarea parametrului primit
    - Deschideți fișierul pentru scriere folosind modul **AppendText**
    - Codul final al acestui constructor ar trebui să arate așa:

```
private string fileName;
private StreamWriter auditFile;

public Audit(string fileToUse)
{
   this.filename = fileToUse;
   this.auditFile = File.AppendText(fileToUse);
}
```

- În clasa **Audit**, scrieți metoda ce va fi folosită pentru a face înregistrarea la evenimentul **Auditing**. Această metodă se va executa în momentul în care **BankTransaction** aruncă un eveniment. Metoda trebuie să fie publică, **void** și numită **RecordTransaction.** Va primi doi parametri, un object numit sender și un **AuditEventArgs** numit **eventData**
- În corpul metodei, obțineți din parametrul **eventData** referința la tranzacția bancară ce a avut loc. Folosind metoda **WriteLine** a instanței **StreamWriter-**ului, scrieți în fișier suma și data tranzacției
- În clasa **Audit**, creați o metodă publică **void Close** ce închide fișierul **this.auditFile**:

```
public void Close()
{
   if (!closed)
   {
         this.auditFile.Close();
      closed = true;
   }
}
```

Definirea eventului Auditing

- În fișierul **BankAccount**.cs**,** deasupra clasei **BankAccount,** declarați un delegat public de tipul **void** numit **AuditEventHandler** ce primește 2 parametri: un **object** numit **sender** și un **AuditEventArgs** numit data

- În clasa **BankAccount** definiți un eveniment de tipul **AuditEventHander** numit **AuditingTransaction** și inițializați-l la **null**
- Adăugați o metodă publică **AddOnAuditingTransaction(AuditEventHandler handler)**. Această metodă va adăuga pe **handler** la lista de delegați ce sunt înregistrați la event **AuditingTransaction**
- Adăugați o metodă publică **RemoveOnAuditingTransaction(AuditEventHandler handler)**. Această metodă va scoate pe **handler** din lista de delegați ce sunt înregistrați la event **AuditingTransaction**
- Adăugați o a treia metodă, **OnAuditingTransaction**. Această metodă va primi ca parametru o **BankTransaction** și va întoarce void. Metoda va analiza pe **AuditingTransaction** și, dacă delegatul nu este **void**, va instanția un obiect de tipul **AuditEventArgs** și îl va folosi pentru a apela toți delegații.
- Metodele **Deposit** și **Withdraw** vor trebui modificate, pentru a apela **OnAuditingTransaction** în momentul în care s-a realizat o modificare a soldului.

Înregistrarea la evenimentul Auditing

- Ultima etapă constă în instanțierea obiectului **Audit** ce va asculta la evenimentul **Auditing**. Obiectul **Audit** va face parte din clasa **BankAccount** și va fi creat la instanțierea contului, astfel încât fiecare cont să aibe propriul fișier.
- Definiți o variabilă publică **accountAudit** de tip **Audit** în clasa **BankAccount**
- Definiți o metodă numită **AuditTrail**. Această metodă va instanția obiectul **Audit** și îl va înscrie evenimentului **Auditing**. Metoda primește un parametru de tip **string**, calea către fișier și va întoarce **void**. Metoda va:
    - Instanția **accountAudit** folosind stringul furnizat ca parametru
    - Crea o variabilă de tipul **AuditEventHandler** numită **doAuditing** și inițializați-o folosind metoda **RecordTransaction** a obiectului **accountAudit**
    - Adăugați pe **doAuditing** la lista de obiecte înregistrate la eveniment. Folosiți metoda **AddOnAuditingTransaction**.

Pentru a testa evenimentul

- Scrieți o clasă de test, ce conține o metodă **Main**
- Creați două conturi bancare
- Folosiți metoda **AuditTrail** pentru a înregistra obiectele la evenimentul **Auditing**
- Operați o serie de depozite și retrageri.
- Închideți ambele conturi.
- Compilați proiectul și corectați eventualele erori
- Rulați aplicația
- Deschideți fișierele ce au fost create pentru a vedea conținutul ce a fost scris pe hard