Laborator 5 – Codificarea datelor

Obiective

Dupa completarea acestui laborator veti dobandi urmatoarele cunostinte:

- Implementarea unei aplicatii care sa arhiveze si sa dezarhiveze fisiere din sistem
- Implementarea unei aplicatii de criptare cu chei asimetrice
- Implementarea unei aplicatii de serializare si deserializare a obiectelor

➢ Exercitiul 1 – Arhivare

In acest exercitiu va trebui sa implementati o aplicatie Windows Forms care sa fie capabila de arhivarea unor fisiere la alegere, vizualizarea unei arhive din sistem si dezarhivarea acesteia intr-un folder ales de utilizator.

Pornind de la suportul dat, implementati urmatoarele subpuncte pentru a indeplini cerinta de mai sus:

1. Identificati comentariul „TODO 1” din fisierul „Form1.cs”. Metoda in care se regaseste acest comentariu reprezinta handler-ul pentru butonul „File(s)” din formular. **Salvati calea catre fisierele selectate de „openFileDialog” in „textBox1”, fiecare cale separata de urmatoarea prin „,” (virgula). In plus, introduceti o instructiune care sa marcheze faptul ca fisierele selectate NU sunt foldere.**

2. Identificati comentariul „TODO 2” din fisierul „Form1.cs”. **Salvati calea catre folderul selectat de „folderBrowserDialog1” in „textBox1”. De asemenea, introduceti o instructiune care sa marcheze faptul ca fisierul selectat este un folder.**

3. Identificati comentariul „TODO 3” din fisierul „Form1.cs”. **Creati arhiva folosind calea catre folderul de arhivat si calea catre arhiva ce urmeaza sa fie construita.** *Hint: Folositi o metoda statica a clasei ZipFile*.

4. Identificati comentariul „TODO 4” din fisierul „Form1.cs”.
   a. **Parsati textul din „textBox1” si obtineti toate fisierele selectate**
   b. **Creati o arhiva goala in care urmeaza sa introduceti fisierele de mai sus.** *Hint: Folositi o metoda statica a clasei ZipFile si salvati arhiva intr-un obiect de tip ZipArchive*
   c. **Pentru fiecare fisier din cele selectate adaugati o noua intrare in arhiva creata la punctul b.**
   d. **Eliberati resursele folosite de obiectul arhiva**

   In acest moment puteti compila si testa programul, verificand daca cele 3 butoane de mai sus functioneaza: File(s), Folder si Create.

5. Identificati comentariul „TODO 5” din fisierul „Form1.cs”.
   a. **Construiti o arhiva din calea catre fisierul selectat** (se ignora cazul in care fisierul selectat nu este o arhiva – nu faceti verificari). *Hint: Folositi o metoda statica a clasei ZipFile si salvati arhiva intr-un obiect de tip ZipArchive*
   b. **Pentru fiecare entry din arhiva** (*Hint: ZipArchiveEntry*) **adaugati numele intreg al fisierului in „listBox1”**(*Hint: Folositi proprietatea „Items” a acestui obiect*)

   In acest moment se poate testa programul prin deschiderea unei arhive de pe disc. Aceasta functionalitate ar trebui sa afiseze (**nu sa extraga**) continutul arhivei selectate.

6. Identificati comentariul „TODO 6 - <a>/<b>” din fisierul „Form1.cs”.
   a. **Salvati in „textBox2” calea catre arhiva ce urmeaza sa fie dezarhivata. Calea este obtinuta prin „openFileDialog1”.**
   b. **Dezarhivati arhiva selectata folosind o metoda statica a clasei „ZipFile”**

➢ Exercitiul 2 – Criptare

In acest exercitiu veti cripta o parte a unui fisier de tip xml. Fisierul original se numeste “mesajClar.xml” si se afla in folderol “bin/Debug” (calea default a unui proiect C#). Continutul lui este urmatorul:

```
<root>
        <name>Student</name>
        <password>test123#</password>
</root>
```

Scopul acestui exercitiu este criptarea elementului “password” **(nu numai a continutului)**, crearea unui nou fisier xml cu elementul criptat. Dupa generarea acestui fisier, se doreste decriptarea acestuia in cel de-al treilea fisier pentru a se vedea daca se obtine mesajul original. Exercitiul foloseste o combinatie de criptare simetrica cu criptare asimetrica. Mai exact, se foloseste o criptare asimetrica pentru transmiterea cheii simetrice (cheia simetrica este criptata folosind sistemul de chei public/privat si transmisa persoanei cu care se doreste comunicarea), urmand ca la viitoarele mesaje sa se foloseasca doar cheia simetrica (pentru o mai buna performanta). Optional, exercitiul cripteaza de la inceput elementul “password”, impreuna cu cheia simetrica.

Pentru implementarea acestui exercitiu puteti porni de la proiectul de start oferit. Acesta contine clasa “Program” cu 3 metode: **Main**, **Encrypt** si **Decrypt.** Implementati urmatoarele subpuncte:

1. Identificati comentariul “TODO 1”. **Incarcati continutul fisierului xml in obiectul “xmlDoc”**. *Hint: Exista in pdf-ul prezentarii un exemplu la criptarea cu chei simetrice*

2. Identificati comentariul “TODO 2”. **Identificati elementul XML de criptat dupa parametrul “ElementToEncrypt” primit ca parametru.**

3. Identificati comentariul “TODO 3”. **Creati o cheie Rijndael de 256 de biti**. Aceasta este cheia simetrica ce va fi utilizata pentru criptarea mesajelor intre transmitator si destinatar.

4. Identificati comentariul “TODO 4”. **Folositi variabila “eXml” pentru criptarea elementului dorit. Atribuiti rezultatul variabilei “encryptedElement”.**

5. Identificati comentariul “TODO 5”. **Folositi variabila “eXml” pentru criptarea cheii de sesiune folosind cheia RSA primita ca parametru.** In acest punct folosim cheia publica RSA sa criptam cheia simetrica. **Atribuiti rezultatul variabilei “encryptedKey”.**

6. Identificati comentariul “TODO 6”. **Inlocuiti elementul original cu elementul criptat.** *Hint: Folositi o metoda statica a clasei “EncryptedXml”*.

7. Identificati comentariul “TODO 7”. **Creati un obiect xml criptat folosind documentul xml primit ca parametru. Adaugati acestui obiect o mapare cheie-nume si decriptati documentul.**

Compilati si testate programul.

➢ Exercitiul 3 – Serializare XML

Exercitiul 3 presupune implementarea unei aplicatii care serializeaza un obiect intr-un fisier XML si respectiv deserializeaza un fisier XML intr-un obiect. Pentru acest exercitiu aveti la dispozitie un proiect startup. Acesta contine structura unei comenzi din lumea reala: Clasa **PurchaseOrder** ce va fi serializata si care contine, printre altele, o instanta a clasei **Address** (adresa unde se vor livra produsele) si o lista de obiecte **OrderedItem** ce modeleaza un produs.

1. **Analizati clasele aplicatiei, adnotarile ce apar la nivelul claselor sau a campurilor si comentariile asociate acestora**

2. Identificati comentariul „TODO 1”. **Construiti un obiect ce poate serializa o instanta a clasei „PurchaseOrder”.** *Hint: Pdf-ul prezentarii*

3. Identificati comentariul „TODO 2”. **Atribuiti valori campurilor obiectelor „billAddress” (de tip Address) si „i1” (de tip OrderedItem)**.

4. Identificati comentariul „TODO 3”. **Folositi obiectul creat la punctul 2 pentru a serializa comanda. Eliberati resursele folosite pentru acest lucru!**

5. Identificati comentariul „TODO 4”. **Deserializati fisierul xml in obiectul „po” de tip PurchaseOrder. Atentie, este nevoie de un stream pentru citirea din fisier.**

Compilati si testati programul.