Laborator 9 – Reflection

Obiective

Dupa completarea acestui laborator veti dobandi urmatoarele cunostinte:

- Implementarea unei aplicatii care sa incarce o unitate de asamblare in mod dinamic
- Implementarea unei aplicatii care sa foloseasca reflection pentru a-si schimba in mod dinamic comportamentul
- Implementarea unei aplicatii pentru implementarea design pattern-ului „Factory Pattern" folosind Reflection

• Exercitiul 1

In acest exercitiu veti analiza o unitate de asamblare si veti descoperi ce tipuri cuprinde si ce proprietati si metode au acestea.

Taskul 1:

- Deschideti solutia oferita ca startup pentru laborator.
- Observati proiectul **CalculatorApp**
- Expandati proiectul si faceti dublu click pe Form1.cs. Acesta reprezinta o fereastra de WindowsForms ce reprezinta un calculator basic. Va trebui sa implementati comportamentul la apasarea butoanelor de pe fereastra.
- Deschideti Form1.Designer.cs. Observati metodele pe care va trebui sa le completati.

Taskul 2:

- Adaugati un nou fisier Details.cs.
- Incarcati, in mod dinamic, unitatea de asamblare **Test.dll**. Aceasta se afla un folderul /bin/Debug.
- Obtineti toate tipurile din assembly si afisati numele acestora.
- Pentru fiecare tip gasit, afisati numele metodelor, proprietatilor si campurilor pe care le contine.

• Exercitiul 2

In acest exercitiu veti complete metodele din proiectul **CalculatorApp,** astfel incat sa obtineti o aplicatie functionala.

Taskul 1:

- In constructorul clasei Form1, incarcati dll-ul Test.dll, obtineti tipul **Test.Calculator** si creati o instanta a acestui tip.

- Pentru a vedea rezultatele operatiilor, trebuie implementata metoda **SetResult.** Aceasta va prelua valoare proprietatii **Number** a instantei **calcInstance** si o va seta in textbox-ul **restulTb.**

- Implementati comportamentul la apasarea butonului “+” (metoda **AddValue).** Obtineti metoda necesara (dintre cele afisate la exercitiul 1) pentru a efectua operatia de adunare. Chemati acesta metoda, trimitand ca parametru valoarea scrisa in textbox-ul **valueTb.**

- Implementati comportamentul pentru operatia de scadere (**SubstractValue**).

- Implementati metoda Clear, care va reseta valoarea retinuta in instanta Calculator.

- La implementarea metodei Equal trebuie sa tineti cont de ultima comanda apasata. De aceea, trebuie sa verificati daca **lastCommand** este Add sau Substract si, in functie de asta, sa apelati metoda necesara.

Taskul 2:

- Compilati programul si rezolvati eventualele erori
- Testati aplicatia facand diverse calcule de adunare si scadere.

• Exercitiul 3

In acest exercitiu veti implementa Factory Pattern, folosind reflection. **Reminder:** pattern-ul trebuie sa furnizeze, la cerere, o instanta diferita, in functie de comportamentul dorit.

- Creati o interfata cu numele **„Animal”** care sa contina o metoda **„Speak”**. Metoda intoarce void si nu primeste niciun parametru.

- Creati 3 clase concrete, **„Bird”**, **„Dog”** si **„Cat”** care implementeaza interfata de mai sus. Datorita procesului de mostenire, aceste clase vor fi nevoite sa ofere implementare metodei **„Speak”**.
- In clasa Bird oferiti implementare metodei Speak: afisati la consola textul **„Cirip”.**
- In clasa Dog oferiti implementare metodei Speak: afisati la consola textul **„Ham”**.
- In clasa Cat oferiti implementare metodei Speak: afisati la consola textul **„Miau”**.
- Creati o clasa cu numele **„AnimalFactory”**. Aceasta clasa va fi fabrica („factory”) de instante ce implementeaza interfata „Animal”. Clasa va avea o metoda **statica**, cu numele **„CreateAnimal”** ce intoarce **Animal** si primeste ca argument un **string**.
    - In aceasta metoda, stringul primit ca parametru va reprezenta numele unei clase concrete.
    - Metoda va crea o instanta a acestei clase folosind acest string si o va returna.
    - Pentru a face acest lucru este nevoie de construirea unui obiect de tip **Type** cu ajutorul string-ului (**Atentie: trebuie furnizat full name-ul clasei, adica cu tot cu namespace**). Cu obiectul de tip Type se construieste **o instanta** a acelui tip si se returneaza.
- Creati (daca nu exista deja) o clasa ce contine metoda **Main**.
    - Testati aplicatia apeland **AnimalFactory.createInstance(string)** cu parametrii „Bird”, „Dog” si „Cat”.

Nota: *Aceasta implementare ne scuteste de introducerea unei instructiuni **„if”** sau **„switch”** in metoda statica a fabricii de instante, insa este o sursa de erori din cauza stringului primit ca parametru. Daca acest parametru nu reprezinta exact numele unui clase existente, atunci va fi aruncata o exceptie.*