Laborator05

Laborator05
- Interfețe în C#
  - Logger
  - LogType
  - ILogger
  - ConsoleLogger
  - FileLogger
  - Program
  - app.config

Interfețe în C#

Tipul `interface` este mecanismul prin care putem factoriza toate metodele publice comune mai multor clase pe care putem factoriza toate metodele publice comune mai multor clase pe care dorim să le expunem clienților.

O interfață evidențiază un fel de "contract" între implementări concrete (clase concrete) și utilizatorii acestora.

Utilizarea interfețelor are ca efect scrierea aplicațiilor cu un caracter
general, ușor de extins (extensibile) și ușor de testat (testabile)

De ce? Deoarece clasele client (o clasă client este o clasă care utilizează metode ale altei clase ce implementează interfața!) nu depind în mod direct de clasele concrete. Ci sunt dependente doar de interfață!

Cum se declară o interfata in C#:

```
public interface NumeInferfata {
  //declarari de metode (fara implementare)
  void InterfaceMethod();
}
```

Scenariu:

Să se implementeze posibilitatea înregistrării activităților și/sau erorilor care apar în cadrul unei aplicații. (mecanism care poartă numele de `logging`)
Un posibil model de `log` ar putea fi următorul:

Logger

```
namespace Logger {
  public class LogModel {
    public string Message { get; set; }
    public LogType Type { get; set; }
    public DateTime Date { get; set; }
    public string ApplicationName { get; set; }
  }
}
```

LogType

```
public enum LogType {
  Info = 1,
  Warn = 2,
  Debug = 3,
  Error = 4
}
```

Deorece dorim ca librăria noastră `Logger` să fie extensibilă (să putem utiliza diverse medii de storace a logurilor: consolă, fișier, bază de date...), vom crea o interfață comună `ILogger` cu o metodă Log care efectuează operația de logging.

Aceasta interfață va trebui să fie implementată de clasele concrete:

ILogger

```
public interface ILogger {
  // Metoda comuna claselor "Logger"
  void Log(List < LogModel > logs);
}
```

ConsoleLogger

```
namespace Logger {
  public class ConsoleLogger: ILogger {
    // Metoda din Interfata
    public void Log(List < LogModel > logs) {
      if (logs != null) foreach(var log in logs) {
        log.ApplicationName =
ConfigurationManager.AppSettings["ApplicationName"];
        log.Date = DateTime.Now;
        Console.WriteLine(log);
      }
    }
  }
}
```

FileLogger

```
namespace Logger {
  public class FileLogger: ILogger {
    public void Log(List < LogModel > logs) {
      if (logs != null) {
```

```
        var fileName = ConfigurationManager.AppSettings["LogFile"];
        foreach(var log in logs) {
          log.ApplicationName =
ConfigurationManager.AppSettings["ApplicationName"];
          log.Date = DateTime.Now;
          File.AppendAllText(fileName, $ "{log}\n");
        }
      }
    }
  }
}
```

Program

```
class Program {
  static void Main(string[] args) {
    ILogger logger = new ConsoleLogger();
    logger.Log(new List < LogModel > () {
      new LogModel() {
        Message = "action1",
        Type = LogType.Info
      },

      new LogModel() {
        Message = "action2",
        Type = LogType.Error
      }
    });
    Console.ReadKey();
  }
}
```

app.config

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.6.1" />
    </startup>
    <appSettings>
        <add key="LogFile" value="Log.txt" />
        <add key="ApplicationName" value="LogApplication" />
    </appSettings>
</configuration>
```