

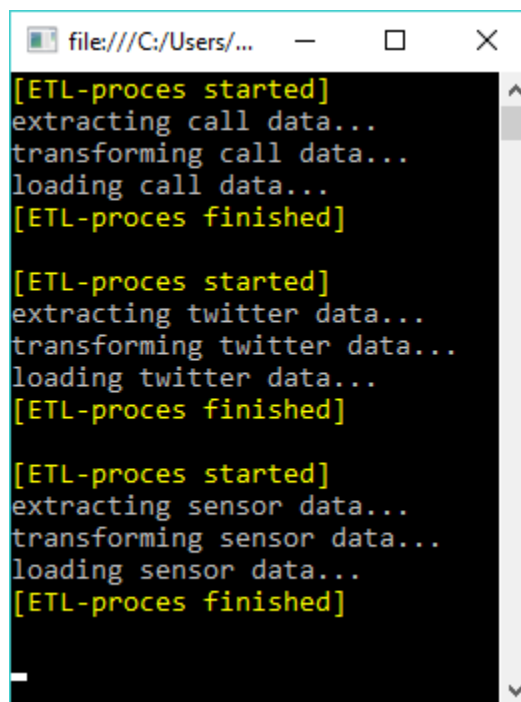
Assignment 1 ('Template Method pattern')

In the world of Big Data the loading of large amounts of data into a DataWareHouse is normally done by the 'Extract-Transform-Load' (ETL) principle, where the data is made available in the correct format. Extract is the reading of data from a data source, Transform is the conversion of the data to the required target format (sometimes the data is merged with other data), and Load is the writing of the converted data to the target database. These 3 steps are executed in a fixed order, but the implementation of each step depends on the data being read.

Create a class `BigDataLoader` containing an ETL-method that uses the 'Template Method' pattern; no objects should be made from this class. Implement 3 specific classes: `CallDataLoader`, `TwitterDataLoader` and `SensorDataLoader`, each providing a (specific) implementation of the 3 ETL-steps.

Implement a (simple) class `BatchProcessor` that can process a list of `BigDataLoaders`; give this `BatchProcessor`-class a method to add a `BigDataLoader` object to the list, and a method to process the complete batch.

Create a (Console) program in which a `BatchProcessor` is created, and produces the output as shown below. In the example, the `BatchProcessor` processes 3 `BigDataLoader` objects (1. *call*, 2. *twitter*, 3. *sensor*).



```
file:///C:/Users/... [ETL-proces started]
extracting call data...
transforming call data...
loading call data...
[ETL-proces finished]

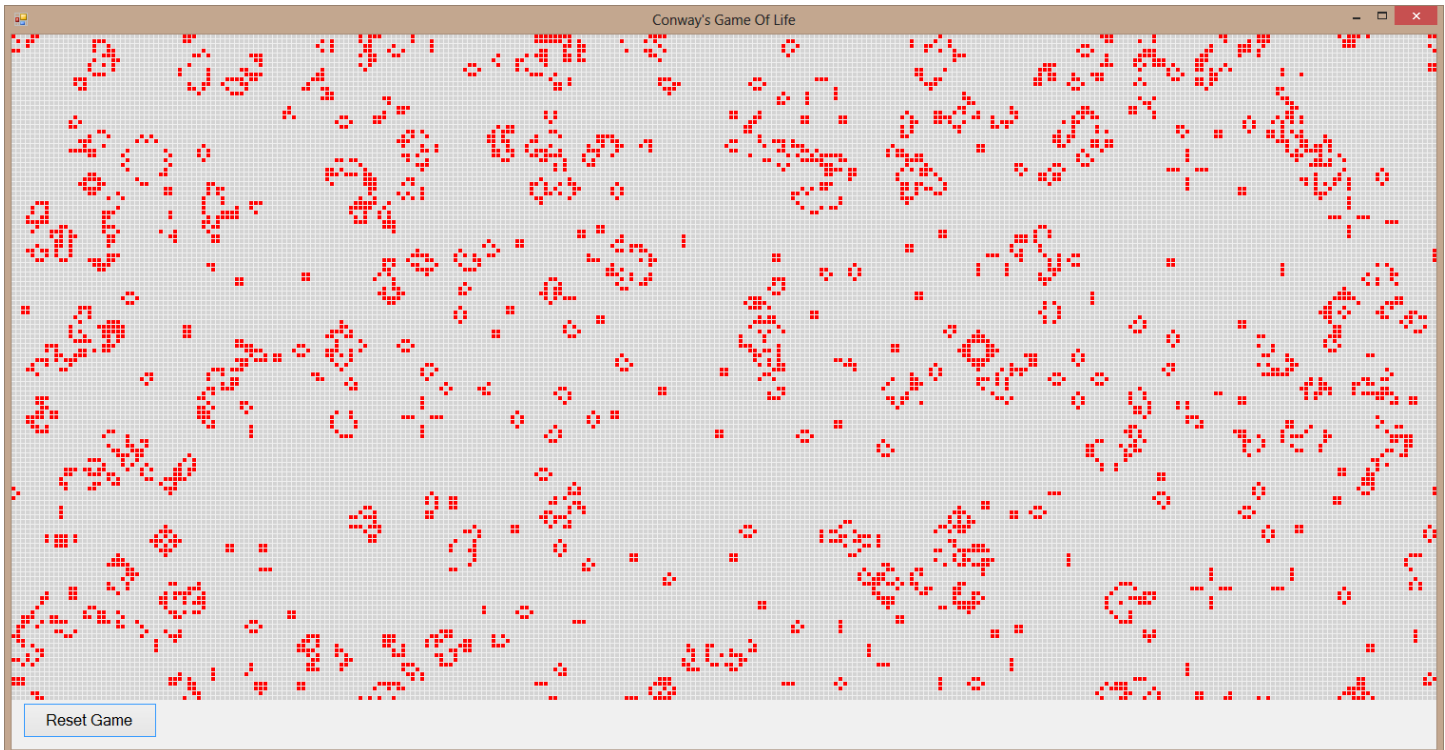
[ETL-proces started]
extracting twitter data...
transforming twitter data...
loading twitter data...
[ETL-proces finished]

[ETL-proces started]
extracting sensor data...
transforming sensor data...
loading sensor data...
[ETL-proces finished]
```

Assignment 2 ('Template Method pattern')

On Moodle you can find application 'Game of Life'. This application uses the default 'rules of life' of John Conway (B3/S23), see screenshot below. Change this application in order to have 2 different 'life behaviours': a 'standard life' and a 'high life' variant (B36/S23, see <http://www.conwaylife.com/wiki/HighLife>). Use the Template Method pattern in class `ConwayGameOfLife`, method 'CellShouldLive' must be implemented by 2 derived classes (StandardLife and HighLife). This method 'CellShouldLive' is called in method 'Evolve'.

More information about Conway's Game of Life can be found at https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life.



Make sure you understand which method is the 'Template Method'.

Assignment 3 ('Observer pattern')

A (Console) application needs to be written in which an MP3-player can play songs. Several displays are displaying information of the current song of the MP3-player, each display in a different way.

The following classes need to be implemented: 'MP3Player', 'SimpleMP3Display' and 'FancyMP3Display'.

The MP3Player class implements the IObservable interface:

```
public interface IObservable
{
    void AddObserver(IObserver observer);
    void RemoveObserver(IObserver observer);
    void NextSong();
}
```

The MP3Player class has the following members:

```
public Song CurrentSong { get; private set; }
public void NextSong() { ... }
```

Method 'NextSong' selects (randomly) a new song and informs all subscribed observers (that a new song has been selected). All songs are stored in a list (`List<Song>` songs).

The 2 display classes (SimpleMP3Display and FancyMP3Display) both implement the IObserver interface (so the MP3player can update/inform them):

```
public interface IObserver
{
    void Update(Song song);
}
```

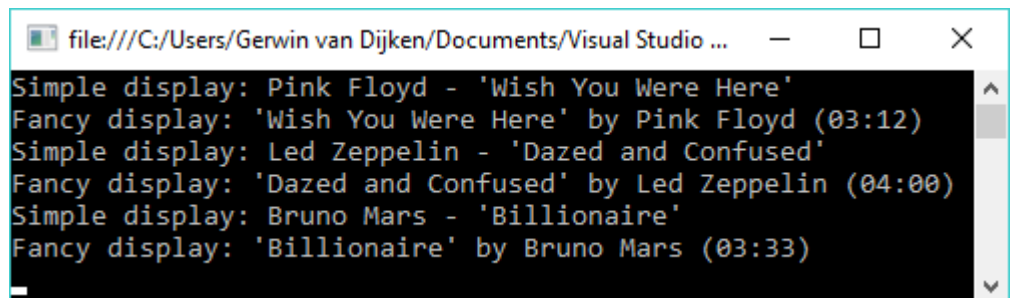
Below you find the main program and the expected output:

```
class Program
{
    static void Main(string[] args)
    {
        // create an MP3 player
        IObservable player = new MP3Player();

        // create the displays
        IObserver mp3Display1 = new SimpleMP3Display(player);
        IObserver mp3Display2 = new FancyMP3Display(player);

        // change song a few times
        // (since there is no hardware, we do it hardcoded here...)
        player.NextSong();
        player.NextSong();
        player.NextSong();

        Console.ReadKey();
    }
}
```



```
file:///C:/Users/Gerwin van Dijken/Documents/Visual Studio ...
Simple display: Pink Floyd - 'Wish You Were Here'
Fancy display: 'Wish You Were Here' by Pink Floyd (03:12)
Simple display: Led Zeppelin - 'Dazed and Confused'
Fancy display: 'Dazed and Confused' by Led Zeppelin (04:00)
Simple display: Bruno Mars - 'Billionaire'
Fancy display: 'Billionaire' by Bruno Mars (03:33)
```