# Fred Memo

# Documentation

# By Team “Fred Flintstone”

# ­Team members:

Toni Ilieva / tonitagd

Stanislava Ivanova / cyberbox

Dimitar Radenkov / mitko100

# Project Description

The project is an interpretation of a classical one player memory game.

The objective of the game is to open all the pairs of cards on the game field. The player has to select two cards. If the selected cards match, they remain open, otherwise they turn back and the player has another try.

The game starts with initial score of 3000 points. Every successful guess adds 50 points to the current score. With every wrong attempt, the score decreases by 50 points. Additional bonus points are added if the level is completed for less than a minute.

Fred Memo has four levels with an ascending number of cards:

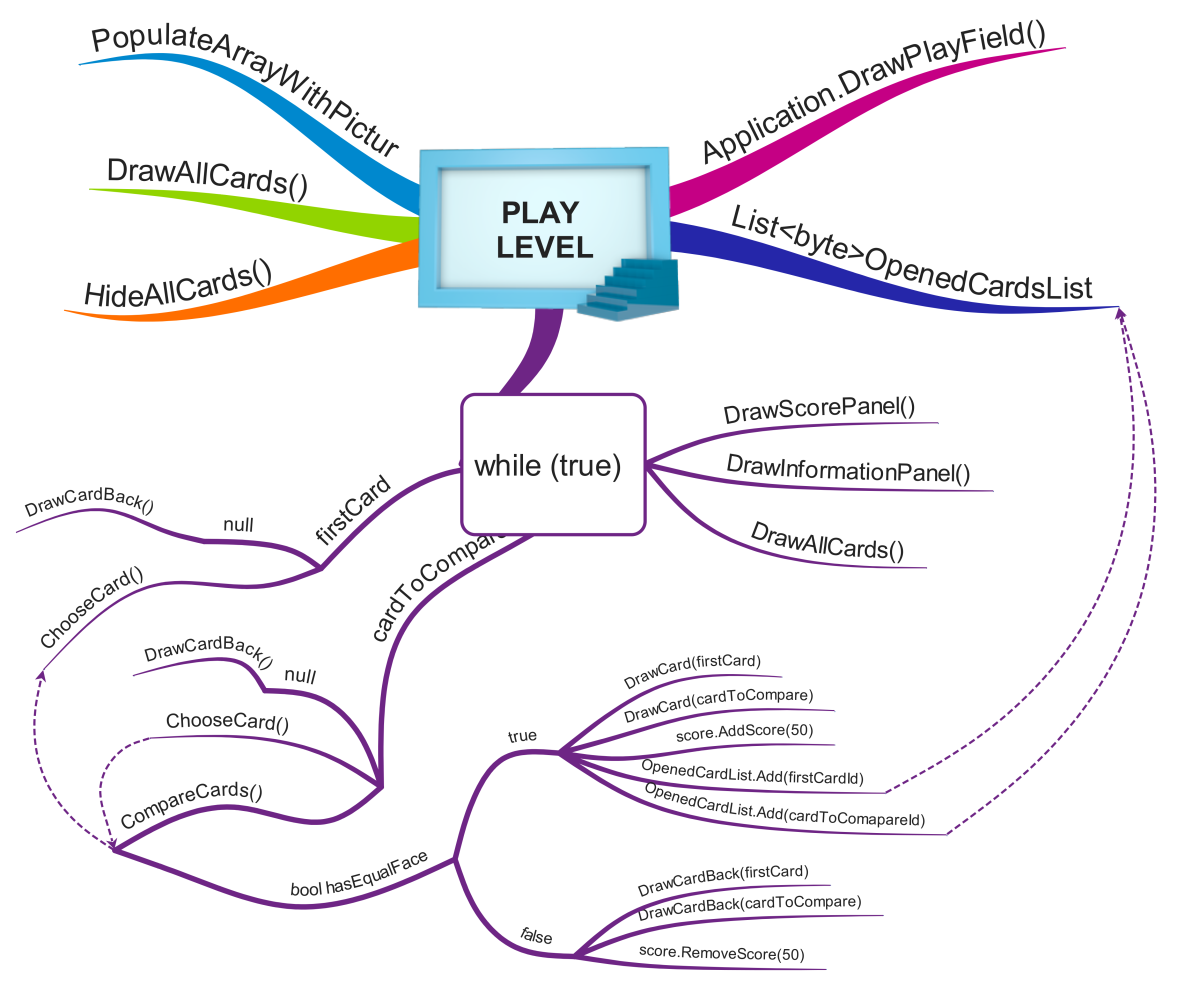
* Level 1 : 12 cards (4x3)
* Level 2 : 16 cards (4x4)
* Level 3 : 20 cards (4x5)
* Level 4 : 24 cards (4x6)

After every completed level the game shows information about:

* Current score
* Time taken for guessing all the cards
* Total number of tries
* Player’s memory status, based on his tries number.

Fred Memo finale comes at level four, after all the cards are opened or when the player score becomes 0 points. The memo keeps log of the 5 highest scores.

## Game logic



## General Requirements

Please define and implement the following assets in your project:

* At least **1 multi-dimensional array**
* At least **3 one-dimensional arrays**
* At least **10 methods** (separating the application’s logic)
* At least **3 existing .NET classes** (like *System.Math* or *System.DateTime*)
* At least **2 exception handlings**
* At least **1 use of external text file**

## Game Repository

* **TFS repository of the project :**

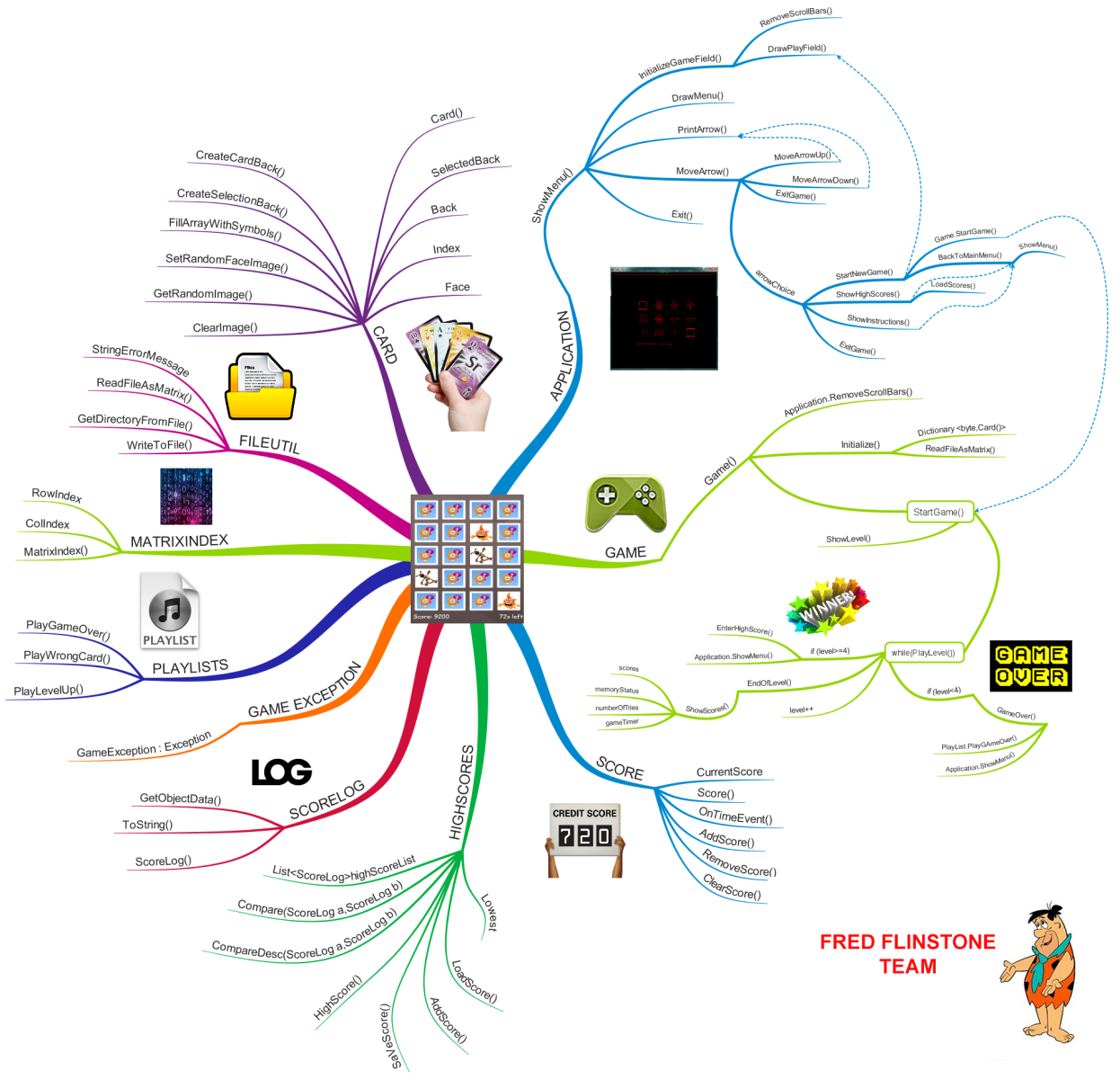
<https://syssboxx.visualstudio.com/DefaultCollection/Fred%20Flinstone%20Project>

* **User (read only) :** [**fredflinston@gmail.com**](mailto:fredflinston@gmail.com)**, password : fredmemo2013**

## Optional Requirements

The project includes the following optional requirements:

* **Object Oriented Programming** - The game structure is organized into 10 classes :
* Application.cs – contains the Main() method, starts the application and shows the user menu. It includes methods for initializing the playfield and moving through the items from the user interface.
* Game.cs – includes all the methods for initializing and playing the game.
* Card.cs – holds the object Card used for the card items in the game and methods for its initialization.
* FileUtil.cs – utility class for reading/writing files.
* MatrixIndex.cs – utility class for populating matrix with random values.
* Score.cs – contains fields and methods for score handling – add or remove score to the current user score.
* ScoreLog.cs – handles the logs creation and stores data for the high scores and serialization.
* HighScore.cs – holds methods for adding, comparing, and loading score into the score log.
* GameException.cs – exception for the game. Holds information about message and origin.
* PlayList.cs – includes the methods for the game’s sound effects.



Class Organization

* **Serialization** – the high scores are written into a binary file
* **Sound effects** areincludedduring the gameplay to indicate:
* Wrong card selection
* Passing to the next level
* Game over