# Breakout

## Installation

This project requires having `Mono` implementation of `.NET framework`.

To add the `Mono` repository to your system, run (for `Ubuntu 18.04`):

```
sudo apt install gnupg ca-certificates
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys
3FA7E0328081BFF6A14DA29AA6A19B38D3D831EF
echo "deb https://download.mono-project.com/repo/ubuntu stable-bionic main"
| sudo tee /etc/apt/sources.list.d/mono-official-stable.list
sudo apt update
```

To install `Mono`, run:

```
sudo apt install mono-devel
```

For other distributions, please see:

```
https://www.mono-project.com/download/stable/
```

## Running

To compile the code and run the game, use:

```
[user@device]$ csc index.cs && mono index.exe
```

## File structure

There is only one file and is composed of several parts mentioned in `Mechanics`.

## Mechanics

The whole game is based on `Microsoft's Window Forms` and main component holding everything together is the class `Breakout`, class which inherits from class `Form`.

As already mentioned in `File structure`, the code has several parts.

```
Declarations
```

In this part are declared `Windows Form` objects such as `Labels`, `Buttons` and `PictureBox`.

Furthermore, there are declared objects creating the game such as `Paddle`, `Ball` and `Blocks`; `Blocks` are contained in a `List` object. Another `List` object contains block-defining colors. This part contains other program-wide variables which change throughout its run.

```
Declarations - Constants
```

Here are declared variables which remain constant throughout the run of the program.

Among these declarations, there are declared colors used for `ForeColor` and `BackColor` properties of `WinForms` objects and are type `Colors`. There is declared game-wide font (`Font`).

```
Utils
```

Part `Utils` contains two methods, one is called when the contents of now-displayed window are about to change and the other closes the window.

```
Custom classes
```

This part holds classes which describe used objects in the game.

Class `Keyboard Manager` when initialized creates a `TextBox` object which then reacts on events `KeyDown` and `KeyUp`.

Class `BoxRestrictions` is a logic point for collisions of all objects in the game. Using method `GetCollisionLocation` it calculates the intersection area percentage of two given blocks and this value is then used to return the area of impact. This class checks collisions of `Ball` object with walls of the game.

Class `XY` is a class which represents a vector by which the object `Ball` upon impact moves.

Class `LeaderboardEntry` defines one entry in the leaderboard - a name, type `string`, and a score of a player, type `int`.

Class `Leaderboard` upon initialization creates `Array` object composed of `LeaderboardEntry` objects. This class takes care of updating, sorting the leaderboard and saving it to the file `leaderboard.txt`.

Class `Brick` holds all properties of each brick. Its constructor takes as arguments the form itself, left and top coordinate of each block, width and height of the block and then block color and its assigned value.

Class `Paddle` creates rectangular paddle with which the player navigates the ball. At the game's start, the paddle is in the middle of the playing area and the player moves it left or right only. Paddle does not shrink with receding number of blocks. Constructor of this class takes as arguments the form, left and top

coordinates of the paddle and its height and width. This class also moves the paddle within the left and right wall of the game.

Class `Ball` creates moving ball whose main task is to hit objects `Brick` and destroy them. This class has implemented methods which control the ball's movement before and after impact with another object or a wall. After impact with object type `Brick`, the ball changes direction based on information from class `BoxRestrictions`. After impact with object type `Paddle`, the ball changes direction and velocity based on point of impact on the paddle. Lastly, after impact with walls, the ball either changes its direction vertically or horizontally, based on the wall it collided with. This movement is defined in class `XY`.

```
Create Labels
```

This part of the code harbors methods for creating all labels used in the game.

```
Create PictureBox
```

This part of the code contains a method for creating a `Window Forms` object `Picture Box` which is used as a game logo.

```
Create TextBox
```

This part of the code contains a method for creating a `Window Forms` object `Text Box` which is used as a input point of the player's name. In this is also a method which checks wether there was some name inputted.

```
Create Button
```

This part of the code harbors methods for creating all buttons used in the game.

```
Windows
```

This part of the code contains several methods which create user's interface. In order of display: method `LogIn` takes player's name, method `BeforeGame` processes player's name and waits for player's initiation of the game. Then there are two methods - final windows - which display the final windows based on player's success in the game. Method `WinningGame` shows player's score and updated leaderboard (if any change occurred) and method `LosingGame` gives the player option to quit the game.

```
Create []
```

Following part of the code presents three methods which create main game objects - `CreateBricks`, `CreatePaddle`, `CreateBall`.

```
GAME
```

This part contains three methods: method `HasWon` indicates the player's success of destroying all the bricks. Method `PlayGame` invokes main loop method `Tick`, further described in `Main methods` part. And method `GameOver` takes care of assessing the player's luck and skills in the game.

```
Main methods
```

This part also contains three methods - the `Main` method - entry point of the whole code and method `Breakout` which initiates `Window Forms` form that controls the game. The most important of these three is method `Tick`. This method invokes movement of object `Paddle`, movement of object `Ball` and is also responsible for disposing of hit bricks. This method is repeatedly called using a `Timer` object.

## User interface

The player moves the object `Paddle` left using the left key. Number of presses indicates the number of moves of the object `Paddle` to the left.

The player moves the object `Paddle` right using the right key. Number of presses indicates the number of moves of the object `Paddle` to the right.

The player has `5` lives at the beginning of the game, after they fail to bounce the object `Ball` off the object `Paddle` they lose a life.