OBJECT CALISTHENICS

Practices and learnings from the course

Mats Tyldum

Content

- What is Object Calisthenics?
- Examples from the Tic Tac Toe kata
- Learnings and reflections

What is Object Calisthenics?

- A set of nine simple rules
- Focused on writing more testable, maintainable, readable and comprehensive code
- Easy to understand
- Naturally change how you write code

Rules

1. Only one level of indentation per method
2. Do not use the ELSE keyword
3. Wrap all primitives and strings
4. First class collections
5. One dot per line
6. Do not abbreviate
7. Keep all entities small
8. No classes with more than two instance variables
9. No getters or setters

Tic Tac Toe example

- “Write the engine that would allow two people to play Tic Tac Toe”
- Implementation from course before and after Object Calisthenic lesson completed
- Illustrate 3 examples
- Link to GitHub for full implementation at last slide

Wrap all primitives and strings

- First implementation used strings as players O and X
- Improved clarity and less prone to error by using Enum Player instead

```
private string _currentPlayer = "X";
```

```
public enum Player
{
    None,
    X,
    O,
}
```

```
private Player _currentPlayer = Player.X;
```

Do not use the ELSE keyword

Previous

```
public void PlaceMarkerAt(int square)
{
    if (!string.IsNullOrWhiteSpace(_playedSquares[square]))
    {
        return;
    }

    _playedSquares[square] = _currentPlayer;

    if (_currentPlayer == "X")
    {
        _currentPlayer = "O";
    }
    else
    {
        _currentPlayer = "X";
    }
}
```

New

```
public void PlaceMarker(Square square)
{
    if(_board.IsSquarePlayed(square))
        return;

    _board.PlaceMarker(square, _currentPlayer);

    AlternatePlayer();
}
```

```
private void AlternatePlayer()
{
    if (_currentPlayer == Player.X)
    {
        _currentPlayer = Player.O;
        return;
    }

    _currentPlayer = Player.X;
}
```

Keep all entities small

- Tic Tac Toe class
- Tried to do my best to keep it simple and elegant
- Surprised of the level of improvement in the second attempt
- Example also provides overall look of how other Object Calisthenic rules improved the code

First implementation

```
namespace src
{
    public class TicTacToe
    {
        private string _currentPlayer = "X";
        private string[] _playedSquares = new string[9];

        public string GetCurrentPlayer()
        {
            return _currentPlayer;
        }

        public void PlaceMarkerAt(int square)
        {
            if (!string.IsNullOrWhiteSpace(_playedSquares[square]))
            {
                return;
            }

            _playedSquares[square] = _currentPlayer;

            if (_currentPlayer == "X")
            {
                _currentPlayer = "O";
            }
            else
            {
                _currentPlayer = "X";
            }
        }

        public string GetWinner()
        {
            var horizontalWinner = GetWinnerHorizontal();
            if (horizontalWinner != null) return horizontalWinner;

            var verticalWinner = GetWinnerVertical();
            if (verticalWinner != null) return verticalWinner;

            var diagonalWinner = GetWinnerDiagonally();
            if (diagonalWinner != null) return diagonalWinner;

            return string.Empty;
        }

        private string GetWinnerDiagonally()
        {
            if (_playedSquares[0] != null &&
                _playedSquares[0] == _playedSquares[4] &&
                _playedSquares[4] == _playedSquares[8])
            {
                return _playedSquares[0];
            }

            if (_playedSquares[2] != null &&
                _playedSquares[2] == _playedSquares[4] &&
                _playedSquares[4] == _playedSquares[6])
            {
                return _playedSquares[2];
            }
            return null;
        }

        private string GetWinnerVertical()
        {
            for (var columnStartIndex = 0; columnStartIndex < 3; columnStartIndex++)
            {
                if (_playedSquares[columnStartIndex] != null &&
                    _playedSquares[columnStartIndex] == _playedSquares[columnStartIndex + 3] &&
                    _playedSquares[columnStartIndex + 3] == _playedSquares[columnStartIndex + 6])
                {
                    return _playedSquares[columnStartIndex];
                }
            }
            return null;
        }

        private string GetWinnerHorizontal()
        {
            for (var rowStartIndex = 0; rowStartIndex < 9; rowStartIndex += 3)
            {
                if (_playedSquares[rowStartIndex] != null &&
                    _playedSquares[rowStartIndex] == _playedSquares[rowStartIndex + 1] &&
                    _playedSquares[rowStartIndex + 1] == _playedSquares[rowStartIndex + 2])
                {
                    return _playedSquares[rowStartIndex];
                }
            }
            return null;
        }
    }
}
```

```
1   namespace src
2   {
3       public class TicTacToeObjectCalisthenics
4       {
5           private Player _currentPlayer = Player.X;
6           private Board _board = new Board();
7
8           public Player GetCurrentPlayer()
9           {
10              return _currentPlayer;
11          }
12
13          public void PlaceMarker(Square square)
14          {
15              if(_board.IsSquarePlayed(square))
16                  return;
17
18              _board.PlaceMarker(square, _currentPlayer);
19
20              AlternatePlayer();
21          }
22
23          private void AlternatePlayer()
24          {
25              if (_currentPlayer == Player.X)
26              {
27                  _currentPlayer = Player.O;
28                  return;
29              }
30
31              _currentPlayer = Player.X;
32          }
33
34          public Player GetWinner()
35          {
36              return _board.GetWinner();
37          }
38      }
39  }
```

Second implementation, after Object Calisthenics lesson

- *92 vs 39 lines of code*
- *Simpler*

Learnings

- Introduction to TDD and its importance for writing good code
- Introduction to important concepts as Object Calisthenics and the Transformation Priority Premise
- Eye openers:
  - *Wrap all primitives and strings in classes*
  - *Focus on behavior*

Reflections

- First experience with mob programming (fun, but exhausting)
- Always room to improve
- More spare time between sessions to practice

Thank you for your attention!

Any questions?

Notes and links

- Presentation inspired by blog post Writing cleaner code with Object Calisthenics by Pierre Bouillon
- GitHub repository with complete Tic Tac Toe example: https://github.com/maattss/tic-tac-toe-kata