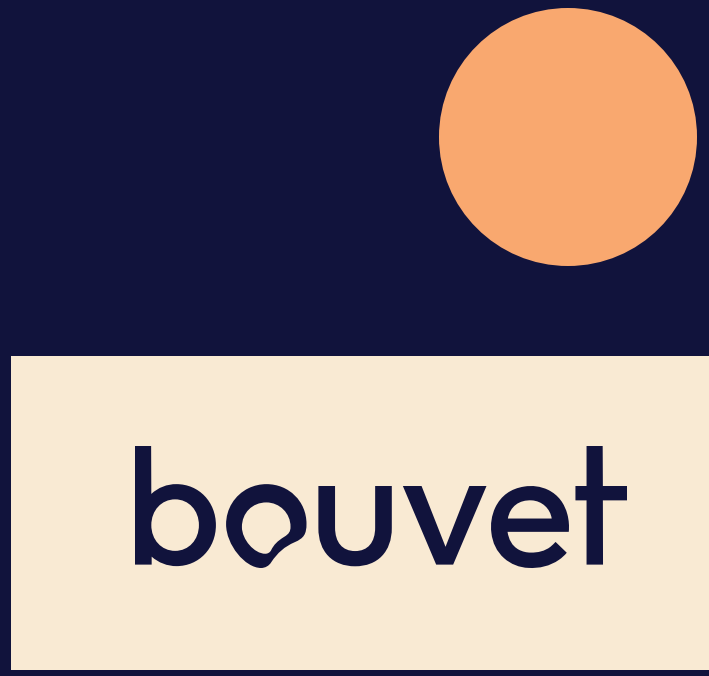
OBJECT CALISTHENICS



OLGA DOLGOVA

OLGA.DOLGOVA@BOUVET.NO

Rules

1. Only one level of indentation per method.
2. Don’t use the ELSE keyword.
3. Wrap all primitives and strings.
4. First class collections (wrap all collections).
5. Only one dot per line.
6. No abbreviations.
7. Keep all entities small.
8. No classes with more than two instance variables.
9. No public getters/setters/properties.

Only one level of indentation per method

```
1  using System;
2  using System.Text;
3
4  public class Board
5  {
6      private char[,] data = new char[10, 10];
7
8      public string BoardToString()
9      {
10         StringBuilder buf = new StringBuilder();
11
12         // 0
13         for (int i = 0; i < 10; i++)
14         {
15             // 1
16             for (int j = 0; j < 10; j++)
17             {
18                 // 2
19                 buf.Append(data[i, j]);
20             }
21             buf.Append("\n");
22         }
23
24         return buf.ToString();
25     }
26 }
```

bouvet

Only one level of indentation per method

```
using System;
using System.Text;

public class Board
{
    private char[,] data = new char[10, 10];

    public string BoardToString()
    {
        StringBuilder buf = new StringBuilder();

        // 0
        for (int i = 0; i < 10; i++)
        {
            // 1
            for (int j = 0; j < 10; j++)
            {
                // 2
                buf.Append(data[i, j]);
            }
            buf.Append("\n");
        }

        return buf.ToString();
    }
}
```

```
using System;
using System.Text;

public class Board
{
    private char[,] data = new char[10, 10];

    public string BoardToString()
    {
        StringBuilder buf = new StringBuilder();
        CollectRows(buf);
        return buf.ToString();
    }

    private void CollectRows(StringBuilder buf)
    {
        for (int i = 0; i < 10; i++)
        {
            CollectRow(buf, i);
        }
    }

    private void CollectRow(StringBuilder buf, int row)
    {
        for (int i = 0; i < 10; i++)
        {
            buf.Append(data[row, i]);
        }
        buf.Append("\n");
    }
}
```

bouvet

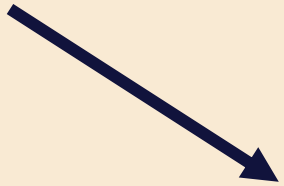
Don’t use the ELSE keyword

```
1   public void Login(string username, string password)
2   {
3       if (userRepository.IsValid(username, password))
4       {
5           Redirect("homepage");
6       }
7       else
8       {
9           AddFlash("error", "Bad credentials");
10          Redirect("login");
11      }
12  }
```

bouvet

Don’t use the ELSE keyword

```php
public void Login(string username, string password)
{
    if (userRepository.IsValid(username, password))
    {
        Redirect("homepage");
    }
    else
    {
        AddFlash("error", "Bad credentials");
        Redirect("login");
    }
}
```

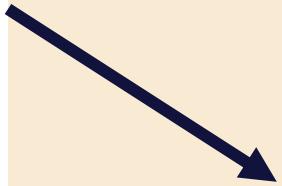


```php
public ActionResult Login(string username, string password)
{
    if (userRepository.IsValid(username, password))
    {
        return Redirect("homepage");
    }

    AddFlash("error", "Bad credentials");
    return Redirect("login");
}
```

Don’t use the ELSE keyword

```php
public void Login(string username, string password)
{
    if (userRepository.IsValid(username, password))
    {
        Redirect("homepage");
    }
    else
    {
        AddFlash("error", "Bad credentials");
        Redirect("login");
    }
}
```



```php
public void Login(string username, string password)
{
    string redirectRoute = "homepage";

    if (!userRepository.IsValid(username, password))
    {
        AddFlash("error", "Bad credentials");
        redirectRoute = "login";
    }

    Redirect(redirectRoute);
}
```

Wrap all primitives and strings

```
1  public class WrappedTypes
2  {
3      public int IntegerValue { get; }
4      public double DoubleValue { get; }
5      public string StringValue { get; }
6
7      public WrappedTypes(int integerValue, double doubleValue, string stringValue)
8      {
9          IntegerValue = integerValue;
10         DoubleValue = doubleValue;
11         StringValue = stringValue;
12     }
13
14     public void DisplayValues()
15     {
16         Console.WriteLine($"Integer Value: {IntegerValue}");
17         Console.WriteLine($"Double Value: {DoubleValue}");
18         Console.WriteLine($"String Value: {StringValue}");
19     }
20 }
21
22 class Program
23 {
24     static void Main()
25     {
26         WrappedTypes wrappedTypes = new WrappedTypes(42, 3.14159, "Hello, World!");
27         wrappedTypes.DisplayValues();
28     }
29 }
```

First class collections

```
1  public class Student
2  {
3      public string Name { get; }
4      public int Age { get; }
5
6      public Student(string name, int age)
7      {
8          Name = name ?? throw new ArgumentNullException(nameof(name));
9          Age = age;
10     }
11 }
12
13 public class Students
14 {
15     private readonly List<Student> _students = new List<Student>();
16
17     public void Add(Student student)
18     {
19         if (student == null) throw new ArgumentNullException(nameof(student));
20         _students.Add(student);
21     }
22
23     public IEnumerable<Student> GetAll()
24     {
25         return _students.AsReadOnly();
26     }
27 }
```

Only one dot per line

```
1  public class Location
2  {
3      public Piece Current { get; set; }
4  }
5
6  public class Piece
7  {
8      public string Representation { get; set; }
9  }
10
11 public class Board
12 {
13     public string BoardRepresentation()
14     {
15         StringBuilder buf = new StringBuilder();
16
17         foreach (Location loc in Squares())
18         {
19             if (loc.Current?.Representation != null && loc.Current.Representation.Length > 0)
20             {
21                 buf.Append(loc.Current.Representation.Substring(0, 1));
22             }
23         }
24
25         return buf.ToString();
26     }
27 }
```

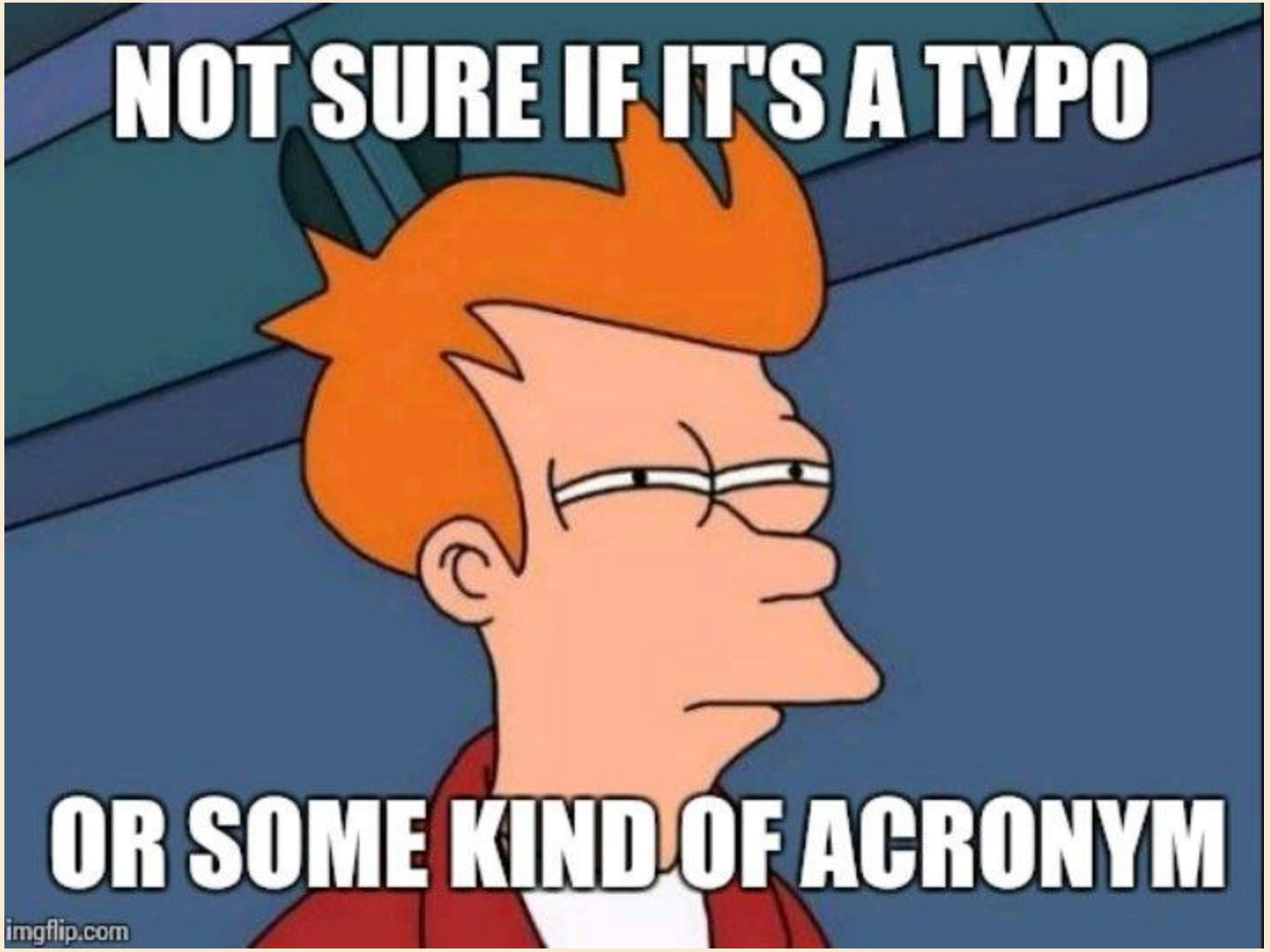
bouvet

Only one dot per line

```
1  public class Location
2  {
3      private Piece _current;
4
5      public void AddTo(StringBuilder buf)
6      {
7          _current?.AddTo(buf);
8      }
9  }
10
11 public class Piece
12 {
13     private string _representation;
14
15     public string Character()
16     {
17         return _representation?.Substring(0, 1);
18     }
19
20     public void AddTo(StringBuilder buf)
21     {
22         buf.Append(Character());
23     }
24 }
25
26 public class Board
27 {
28     public string BoardRepresentation()
29     {
30         StringBuilder buf = new StringBuilder();
31
32         foreach (Location location in Squares())
33         {
34             location.AddTo(buf);
35         }
36
37         return buf.ToString();
38     }
39 }
```

bouvet

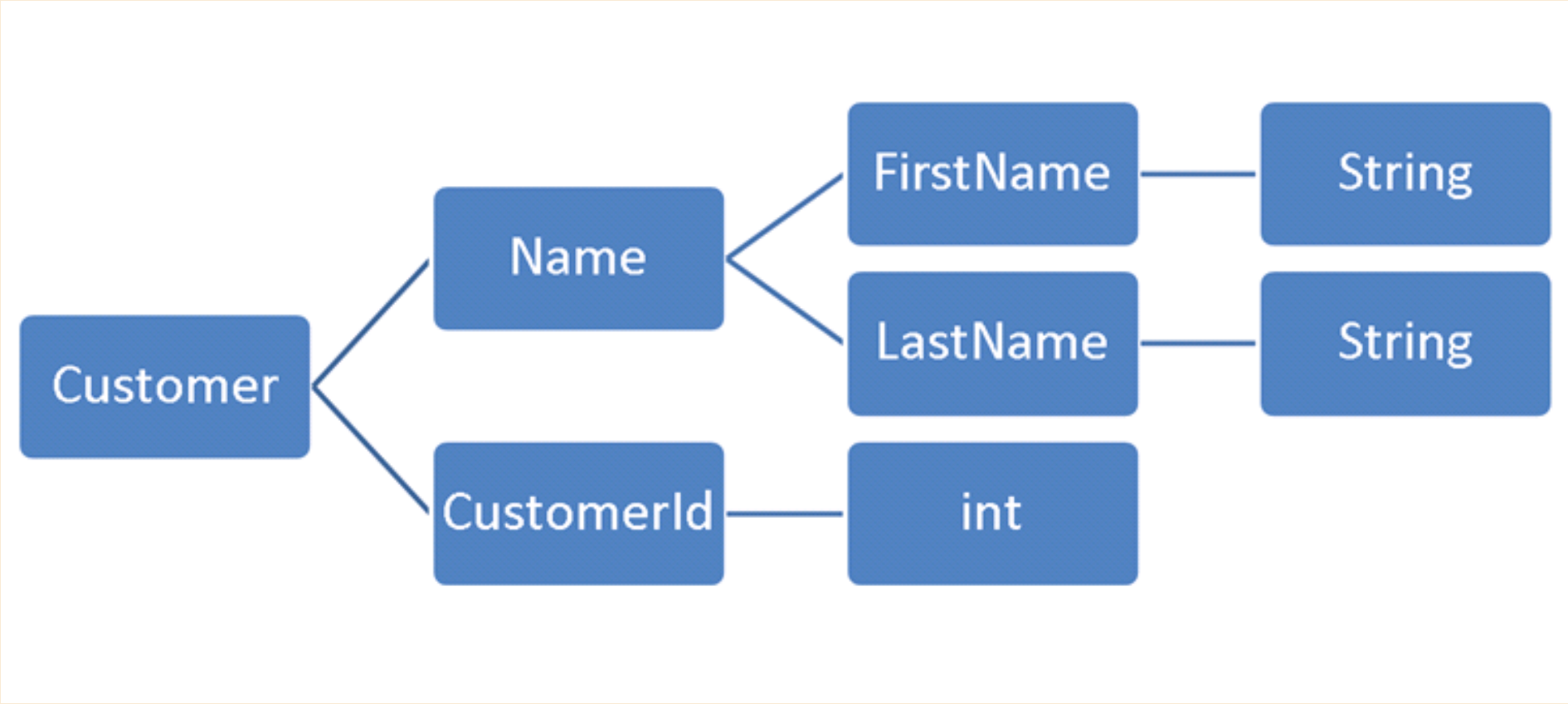
No abbreviations



Keep all entities small

bouvet

No classes with more than two instance variables



bouvet

No public getters/setters/properties

```
1   // Game
2   private int score;
3
4   public void SetScore(int score)
5   {
6       this.score = score;
7   }
8
9   public int GetScore()
10  {
11      return score;
12  }
13
14  // Usage
15  game.SetScore(game.GetScore() + ENEMY_DESTROYED_SCORE);
16
```

bouvet

No public getters/setters/properties

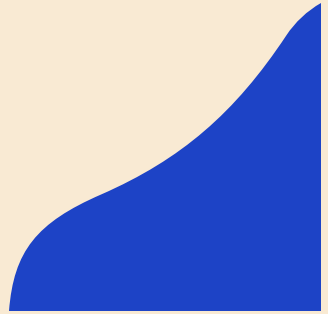
```
1  // Game
2  private int score;
3
4  public void SetScore(int score)
5  {
6      this.score = score;
7  }
8
9  public int GetScore()
10 {
11     return score;
12 }
13
14 // Usage
15 game.SetScore(game.GetScore() + ENEMY_DESTROYED_SCORE);
16
```

```
1  // Game
2  private int score;
3
4  public void AddScore(int delta)
5  {
6      score += delta;
7  }
8
9
10 // Usage
11 game.AddScore(ENEMY_DESTROYED_SCORE);
```

Summary

bouvet

QUESTIONS?



References

- The ThoughtWorks Anthology book (Jeff Bay's piece "Object Calisthenics")

https://www.amazon.com/ThoughtWorks-Anthology-Technology-Innovation-Programmers/dp/193435614X

bouvet

TUSEN TAKK FOR MEG!

olga.dolgova@bouvet.no