Games Technologies

Introduction to OOP

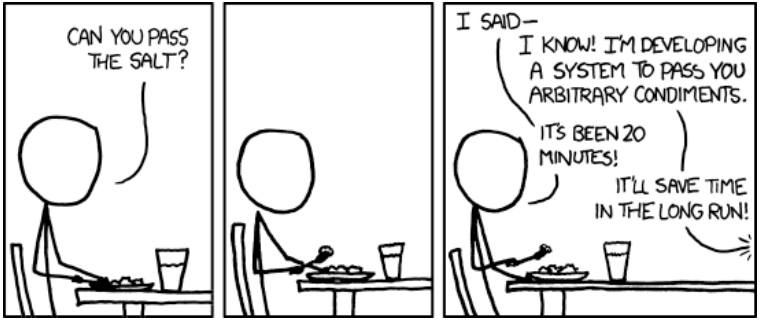
Vassilis Markos, Mediterranean College

Week 02

Contents

Object Oriented Programming

Programmer Humour



This might make more sense after this set of slides.

What Is OOP?

Object Oriented Programming, as per Wikipedia (https://en.wikipedia.org/wiki/Object-oriented_programming):

> *Object-oriented programming (OOP) is a programming paradigm based on objects – software entities that encapsulate data and function(s). An OOP computer program consists of objects that interact with one another.*

So, technically, data bound to actions taken with / about those data.

Why OOP?

- Most of the times, OOP saves up tons of LOC (lines of code).

Why OOP?

- Most of the times, OOP saves up tons of LOC (lines of code).
- In many cases, it is just natural to write in an OO way, since there is a clear mapping between the problem and classes / objects.

Why OOP?

- Most of the times, OOP saves up tons of LOC (lines of code).
- In many cases, it is just natural to write in an OO way, since there is a clear mapping between the problem and classes / objects.
- It (typically) makes code readable and easily extensible.

Why OOP?

- Most of the times, OOP saves up tons of LOC (lines of code).
- In many cases, it is just natural to write in an OO way, since there is a clear mapping between the problem and classes / objects.
- It (typically) makes code readable and easily extensible.
- However, things come at a cost, especially in terms of a relatively steep learning curve.

OOP in C#

- In C# the key OOP construct is the `class`.

OOP in C#

- In C# the key OOP construct is the `class`.
- You can think of classes as collections of:

OOP in C#

- In C# the key OOP construct is the `class`.
- You can think of classes as collections of:
  - tightly related data, alongside;

OOP in C#

- In C# the key OOP construct is the `class`.
- You can think of classes as collections of:
  - tightly related data, alongside;
  - relevant functions that operate on / with those data.

OOP in C#

- In C# the key OOP construct is the `class`.
- You can think of classes as collections of:
  - tightly related data, alongside;
  - relevant functions that operate on / with those data.
- But, an example might be worth a thousand words in this case…

C# Classes

A C# class consists of several sections:

- The class fields, which correspond to the attributes of each object.

C# Classes

A C# class consists of several sections:

- The class fields, which correspond to the attributes of each object.
- The class methods, which are just functions related to the objects themselves.

C# Classes

A C# class consists of several sections:

- The class fields, which correspond to the attributes of each object.
- The class methods, which are just functions related to the objects themselves.
- Typically, setters and getters for any private class fields.

Class Fields

```
using System;

namespace Animals {
    public class Animal {
        // Class attributes
        private string name;
        private int age;
        private double x;
        private double y;
```

Instance Constructor

```
1         // Constructor
2         public Animal(string name, int age, double x, double y
      ) {
3             this.name = name;
4             this.age = age;
5             this.x = x;
6             this.y = y;
7         }
```

More Constructors

```
1        // Inline constructor definition
2        public Animal(string name, int age) : this(name, age,
     0.0, 0.0) { }
3
4        // Default constructor
5        public Animal() {}
```

Class Method

```
1        // Simple class method to move animals around
2        public void MoveBy(double dx, double dy) {
3            x += dx;
4            y += dy;
5        }
```

Class Setters / Getters

```
1         // Getters and setters for some fields
2         public string GetName() {
3             return name;
4         }
5
6         public void SetName(string name) {
7             this.name = name;
8         }
9
10        public int GetAge() {
11            return age;
12        }
13
14        public void SetAge(int age) {
15            this.age = age;
16        }
```

Method Overriding

```
1        // Overriding Base ToString() method
2        public override string ToString() {
3            return String.Format("Name: {0}, Age: {1},
   Location: ({2}, {3})", name, age, x, y);
4        }
```

Main Entry Point

```
using System;

namespace Animals {
    class Test {
        public static void Main(string[] args) {
            Animal alice = new Animal("Alice", 8, 0.0, 1.0);
            Animal bob = new Animal("Bob", 7);
            Console.WriteLine("{0}\n{1}", alice, bob);
        }
    }
}
```

Fun Time!

In-class Exercise #001

Follow Lab instructions in

`lab/Game_Lab_01.pdf`

Use any only resources you might find useful.

Homework

Complete any incomplete lab exercises and then proceed to complete any missing parts of the game lab discussed in class today.

Any Questions?

Do not forget to fill in the questionnaire shown right!

https://forms.gle/dKSrmE1VRVWqxBGZA