Object Oriented Programming 1

Gérald Montúfar 13.10.2025

Object Oriented Programming

What is OOP?

- Programming paradigm based on "objects"
- Objects contain data (attributes) and methods (functions)
- Models real-world concepts

Key OOP Principles

- Encapsulation
- Abstraction
- Inheritance
- Polymorphism

Classes vs Objects

Class	Object
Blueprint for creating objects	Instance of a class
Defines attributes (data) and methods (behaviours)	Contains actual data

Defining a class

```
public class ClassName {
    public int MemberVariable;

    public void Method1() {
        // The methods behaviour
    }
}
```

- Usually a class is defined in its own script e.g. `ClassName.cs`

Creating an Object

```
ClassName instanceName = new ClassName();
instanceName.MemberVariable = 42;
```

Task

1. Write a C# class `Player`
2. Give the `Player` attributes like `Name`, `Health`, and `Position`
3. Give the `Player` methods like `Move()` and `Attack()`
4. Create and instance of the `Player`
5. Set the attributes of the `Player` to a value of your choosing

Constructors

What is a constructor?

- A special method used to initialize objects
- Automatically invoked when an object is created

Declaring a constructor

```
public class ClassName {
    public int MemberVariable;

    // declaring a default constructor
    public ClassName() {
        MemberVariable = 21;
    }
    // declaring a custom constructor
    public ClassName(int memberVariable) {
        MemberVariable = memberVariable;
    }

    public void Method1() {
        // The methods behaviour
    }
}
```

Using a constructor

```
// use of the default constructor
ClassName instanceName = new ClassName();
// use of our custom constructor
ClassName secondInstanceName = new ClassName(7);
```

Task

1. Declare a default constructor that sets the values of your `Player` to sensible values
2. Declare a custom constructor that sets all values of your `Player` object
3. Create an instance of the `Player` using the custom constructor

Static vs Non-Static

	Static	Non-Static
Variable	Holds a value that is the same for all instances	Each instance holds it's own values
Method	Can be called without creating an instance of the class	Can only be called on a specific object after creating an instance
Usage	Utility methods or data that should be shared among all instances	Data and methods that are unique for each instance

Task

1. Add methods to the `Counter` class
    1. A method that resets the `objectCount`
    2. A method that changes the `Name`
2. Call both methods

Object-Oriented Modeling

What is OOM?

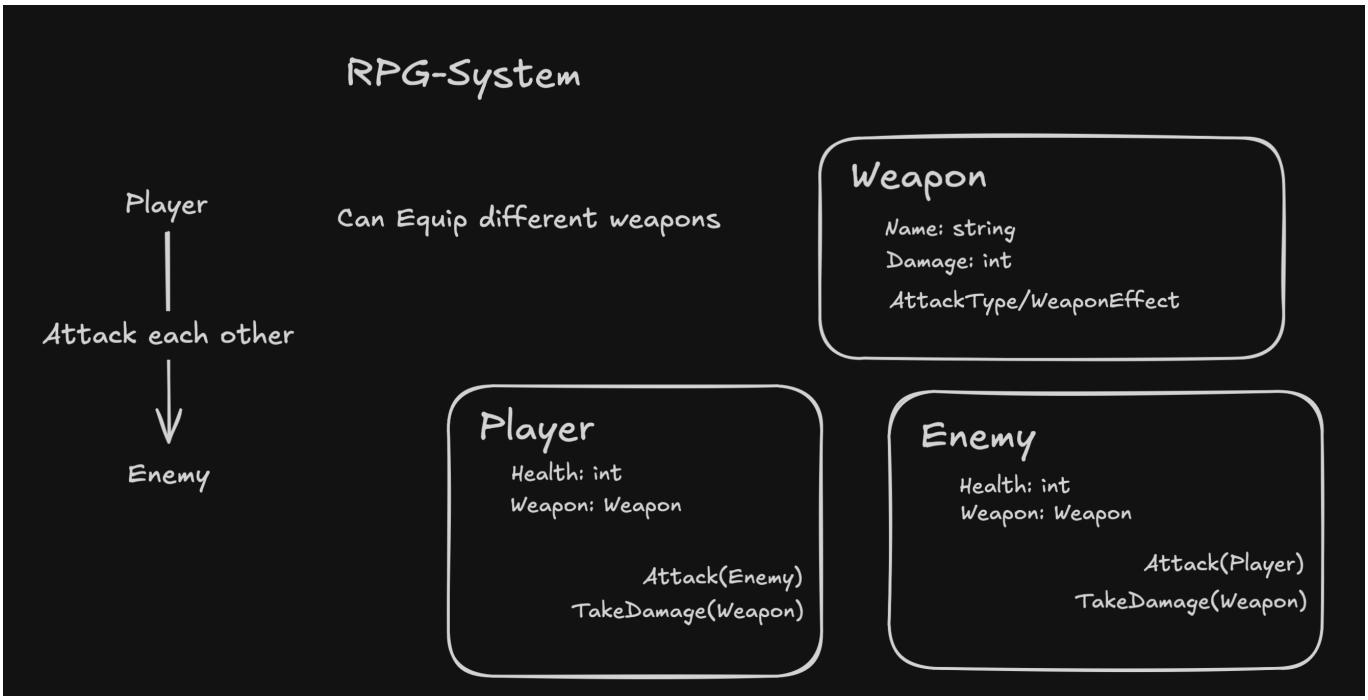
- OOM is a process of designing a system using objects
- Each object represents an entity (e.g., player, enemy, weapon)
- OOM helps structure complex game systems

Why Use OOM?

- **Modularity:** Break the game into smaller, manageable pieces
- **Reusability:** Reuse code for different game elements (e.g., a `Character` class for both players and enemies)
- **Extensibility:** Easily extend functionality by adding new features
- **Maintainability:** Organize code to make it easier to debug and improve

Task-Together

- Design a simple RPG system where:
    - The player can fight enemies
    - The player can equip different weapons
    - Define the necessary classes, attributes, and methods



Code-Together

- Implement the simple RPG-system you designed previously

Recap

- Classes
- Objects
- Constructors
- Static vs Non-Static
- Object-Oriented Modeling

Q&A