# Unit 1:

# Introduction to Object Oriented Concepts

## Welcome to CC211!

Juan Espejo

Go to Classroom

March 18, 2019

# Content

- The Fundamental Concepts
- Objects and Legacy Systems
- Procedural Versus OO Programming
- Moving from Procedural to OO Development
- What Exactly Is an Object?

# Introduction

“The rise of OO methodologies coincides with the emergence of the Internet as a business and entertainment platform."[1]

Figure: The internet

[1] Pag. 5 of [1]

# Introduction

“Technologies change very quickly in the software industry, whereas **concepts evolve**. I use the term ‘evolve’ because, although they remain relatively stable, they do change."[2]

[2] Pag. 5 of [1]

# The Fundamental Concepts

“my list of object-oriented concepts looks like this:

- Encapsulation
- Inheritance
- Polymorphism
- Composition"[3]

[3] Pag. 6 of [1]

# Objects and Legacy Systems

“As OO moved into the mainstream, one of the issues facing developers was the integration of new OO technologies with existing systems...object-oriented code is not meant to replace structured code. Many non-OO legacy systems (that is, older systems that are already in place) are doing the job quite well, so why risk potential disaster by changing or replacing them?"[4]

[4] Pag. 6 of [1]

# Objects and Legacy Systems

“...there is a trend to wrap the legacy systems in object wrappers (object-oriented code that includes other code inside)"[5]

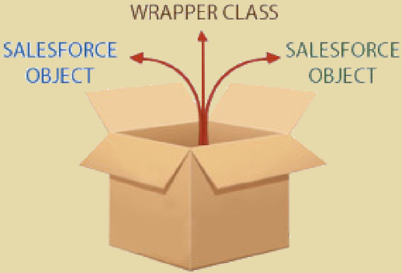


Figure: An object Wrapper

[5] Pag. 7 of [1]

# Procedural Versus OO Programming

“In OO design, the attributes and behaviors are contained within a single object, whereas in procedural, or structured, design, the attributes and behaviors are normally separated."[6]

[6] Pag. 8 of [1]

# Procedural Versus OO Programming

“...in structured programming the data is often separated from the procedures, and often the data is global, so it is easy to modify data that is outside the scope of your code.... Second...testing and debugging are much more difficult. Objects address these problems by combining data and behavior into a nice, complete package."[7]

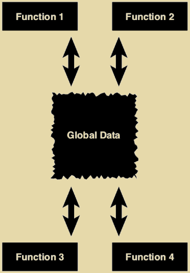


Figure: Figure 1.2 of [1]

[7] Pag. 8 of [1]

# Procedural Versus OO Programming

“(the figure above) shows how the two objects communicate with each other via their methods...The beauty of this is that `myObject` does not need to know how the sum is calculated."[8]

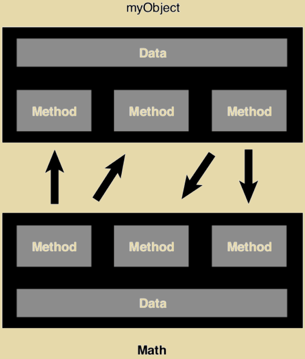


Figure: Figure 1.3 of [1]

[8] Pag. 10 of [1]

# Moving from Procedural to OO Development

## Procedural Programming

“Procedural programming normally separates the data of a system from the operations that manipulate the data. For example, if you want to send information across a network, only the relevant data is sent..."[9]

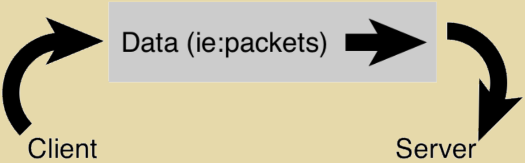


Figure: Figure 1.4 of [1]

[9] Pag. 11 of [1]

# Moving from Procedural to OO Development

## OO Programming

“The fundamental advantage of OO programming is that the data and the operations that manipulate the data (the code) are both encapsulated in the object. For example, when an object is transported across a network, the entire object, including the data and behavior, goes with it."[10]

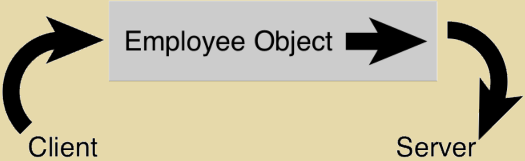


Figure: Figure 1.4 of [1]

[10] Pag. 12 of [1]

# What Exactly Is an Object?

## Object Data

“The data stored within an object represents the state of the object. In OO programming terminology, this data is called *attributes*."[11]

Attributes
SocialSecurityNumber
Gender
DateOfBirth

Figure: Figure 1.6 of [1]

[11] Pag. 12 of [1]

# What Exactly Is an Object?

## Object Behaviors

“The *behavior* of an object represents what the object can do. In procedural languages, the behavior is defined by procedures, functions, and subroutines. In OO programming terminology, these behaviors are contained in *methods*, and you invoke a method by sending a message to it."[12]

“The concept of getters and setters supports the concept of data hiding. Because other objects should not directly manipulate data within another object, the getters and setters provide controlled access to an object’s data. Getters and setters are sometimes called accessor methods and mutator methods, respectively."[13]

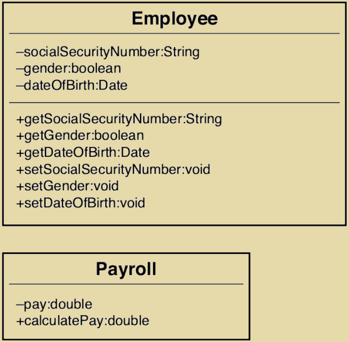
[12] Pag. 13 of [1]

[13] Pag. 14 of [1]

# What Exactly Is an Object?

## UML Class Diagrams

“UML class diagrams are used as a tool to help visualize classes and their relationships to other classes. The use of UML in this book is limited to class diagrams."[14]



**Employee**

–socialSecurityNumber:String
–gender:boolean
–dateOfBirth:Date

+getSocialSecurityNumber:String
+getGender:boolean
+getDateOfBirth:Date
+setSocialSecurityNumber:void
+setGender:void
+setDateOfBirth:void

**Payroll**

–pay:double
+calculatePay:double

Figure: Figure 1.8 of [1]

[14] Pag. 16 of [1]

# References

WEISFELD, M.
*The Object-Oriented Thought Process*, 4th ed.
Developer's Library. Addison-Wesley Professional, 2013.