Unit 2:

How to Think in Terms of Objects

Juan Espejo

Go to Classroom

March 18, 2019

Content

- Knowing the Difference Between the Interface and the Implementation
- Using Abstract Thinking When Designing Interfaces
- Providing the Absolute Minimal User Interface Possible

Introduction

“The fundamental unit of OO design is the class..., before you start to design a system, or even a class, think the problem through and have some fun!"[1]
“When moving to an OO language, you must first go through the investment of learning OO concepts and the corresponding thought process."[2]

[1] Pag. 37 of [1]
[2] Pag. 38 of [1]

Introduction

“Three important things you can do to develop a good sense of the OO thought process:

- Knowing the difference between the interface and implementation
- Thinking more abstractly
- Giving the user the minimal interface possible"[3]

[3] Pag. 38 of [1]

Knowing the Difference Between the Interface and the Implementation

"...when designing a class, what the user needs to know and what the user does not need to know are of vital importance."[4]

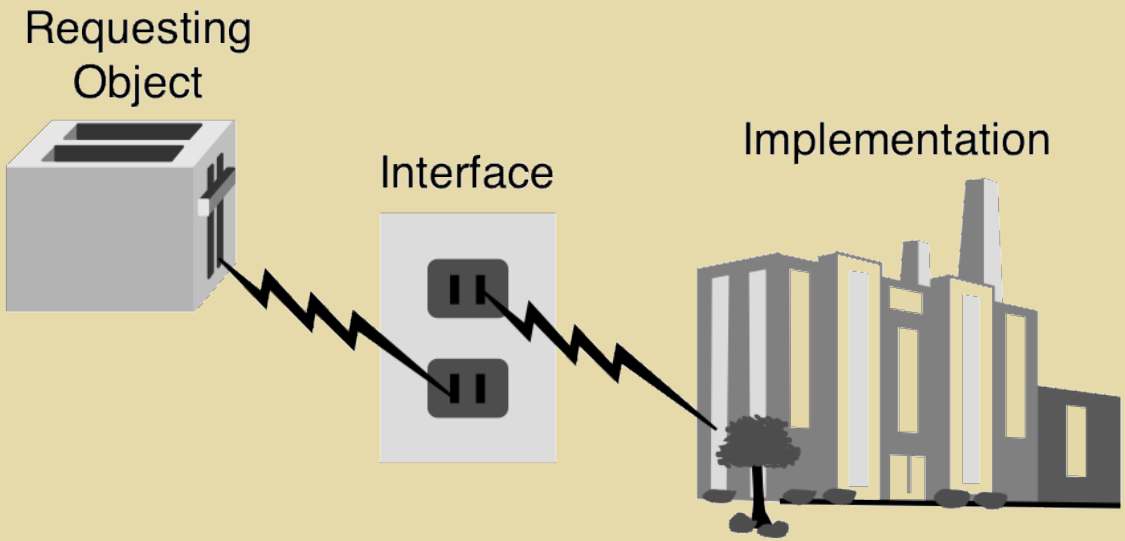

Figure: Power plant (Figure 2.1 of [1])

[4] Pag. 38 of [1]

Knowing the Difference Between the Interface and the Implementation

"A change in the implementation should have no impact on the driver, whereas a change to the interface might."[5]

Figure: Car interior.

[5] Pag. 39 of [1]

Knowing the Difference Between the Interface and the Implementation

"The programmer is the one who needs to understand the interfaces of a class. Therefore, when we talk about users in this chapter, we primarily mean designers and developers–not necessarily end users. Thus, when we talk about interfaces in this context, we are talking about class interfaces, not GUIs."[6]

[6] Pag. 39 of [1]

Knowing the Difference Between the Interface and the Implementation

"Perhaps the most important consideration when designing a class is identifying the audience, or users, of the class... A change to the implementation should not require a change to the user's code... The interface stays the same regardless of how the implementation changes."[7]

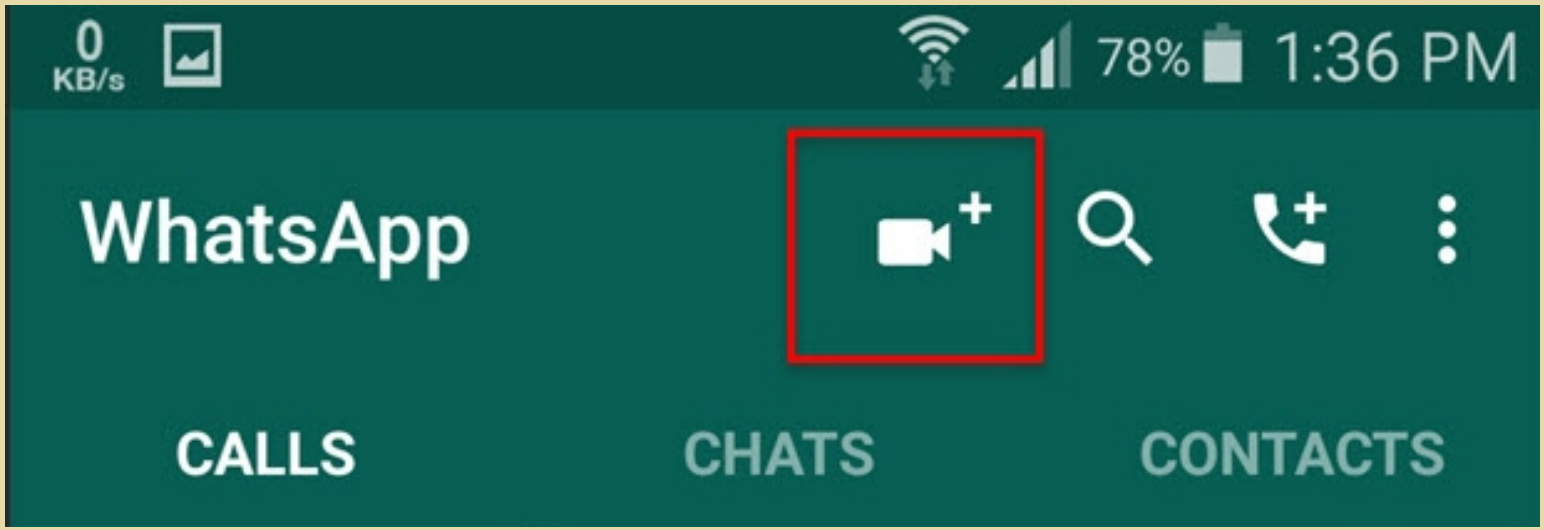

Figure: Whatsapp video call.

[7] Pag. 40 of [1]

Knowing the Difference Between the Interface and the Implementation

"...knowing your end users is always the most important issue when doing any kind of design... Thus, you add interfaces only when it is requested. Never assume that the user needs something."[8]

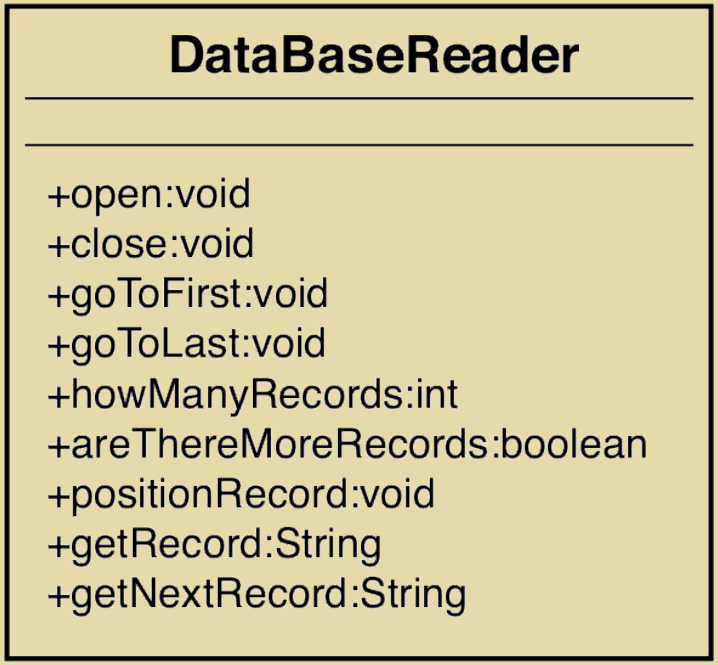

Figure: A class diagram representing an interface to the `DataBaseReader` class (Figure 2.2 of [1])

[8] Pag. 41-42 of [1]

Knowing the Difference Between the Interface and the Implementation

"By separating the user interface from the implementation, we can save a lot of headaches down the road."[9]

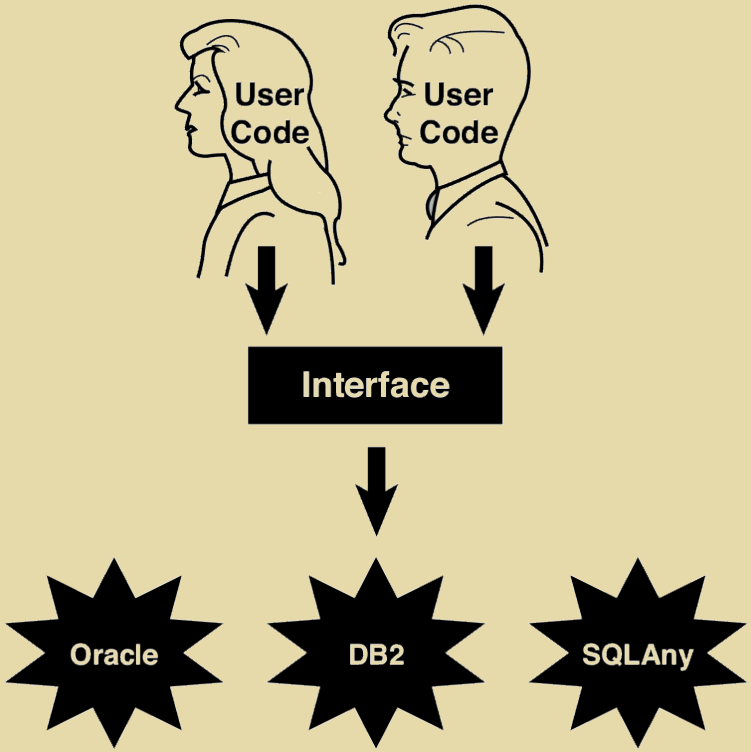

Figure: The database implementations are transparent to the end users, who see only the interface. (Figure 2.3 of [1])

[9] Pag. 45 of [1]

Using Abstract Thinking When Designing Interfaces

"One of the main advantages of OO programming is that classes can be reused. In general, reusable classes tend to have interfaces that are more abstract than concrete. Concrete interfaces tend to be very specific, whereas abstract interfaces are more general."[10]

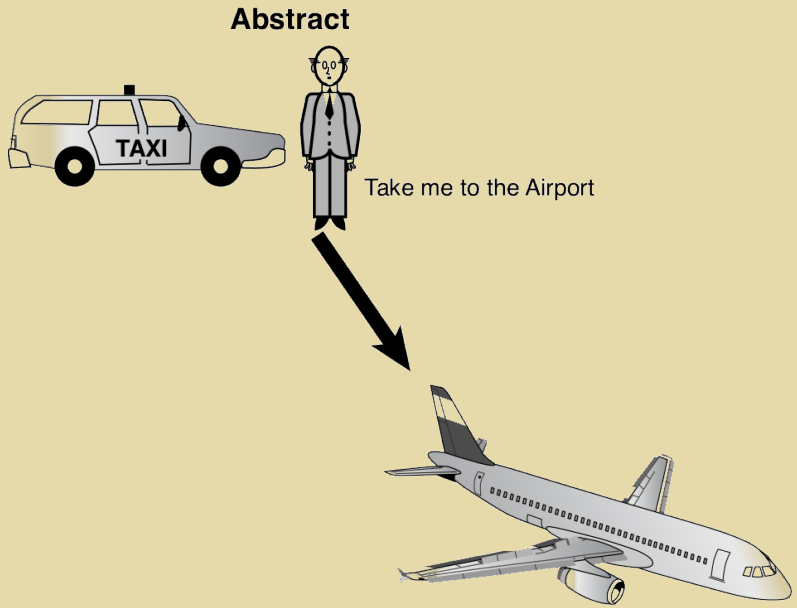

Figure: Figure 2.4 of [1])

[10] Pag. 45 of [1]

Using Abstract Thinking When Designing Interfaces

"...which phrase is more reusable: 'Take me to the airport,' or 'Turn right, then right, then left, then left, then left'?."[11]

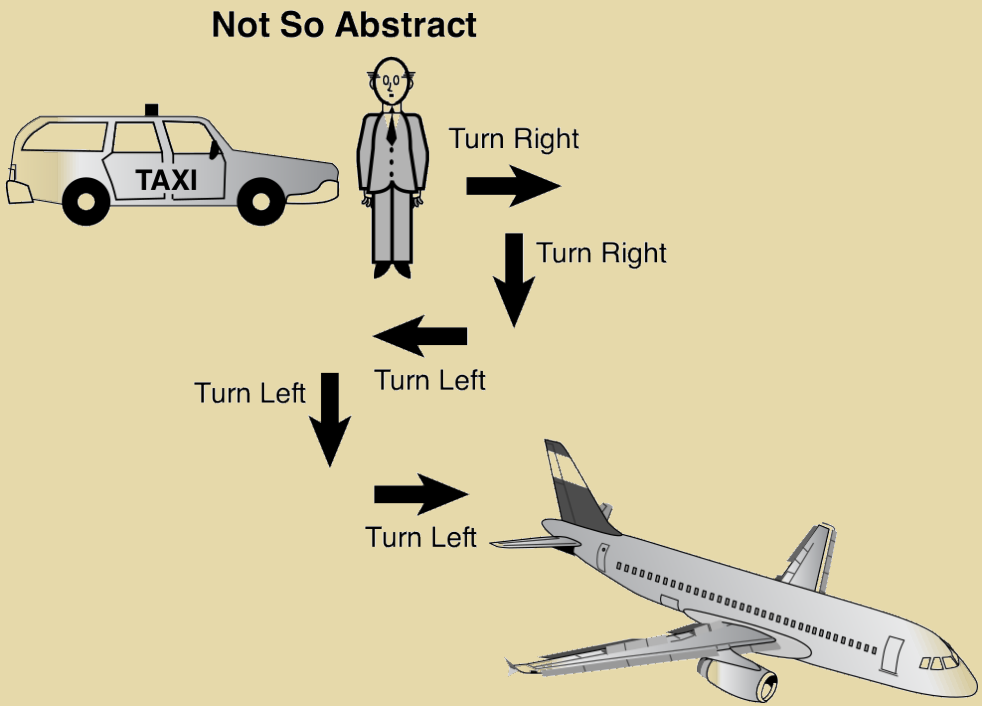

Figure: Figure 2.5 of [1])

[11] Pag. 46 of [1]

Providing the Absolute Minimal User Interface Possible

"When designing a class, the general rule is to always provide the user with as little knowledge of the inner workings of the class as possible. To accomplish this, follow these simple rules:

- Give the users only what they absolutely need
- Public interfaces define what the users can access
- Design classes from a user's perspective
- go over the requirements and the design with the people who will actually use it"[12]

[12] Pag. 47-48 of [1]

Providing the Absolute Minimal User Interface Possible

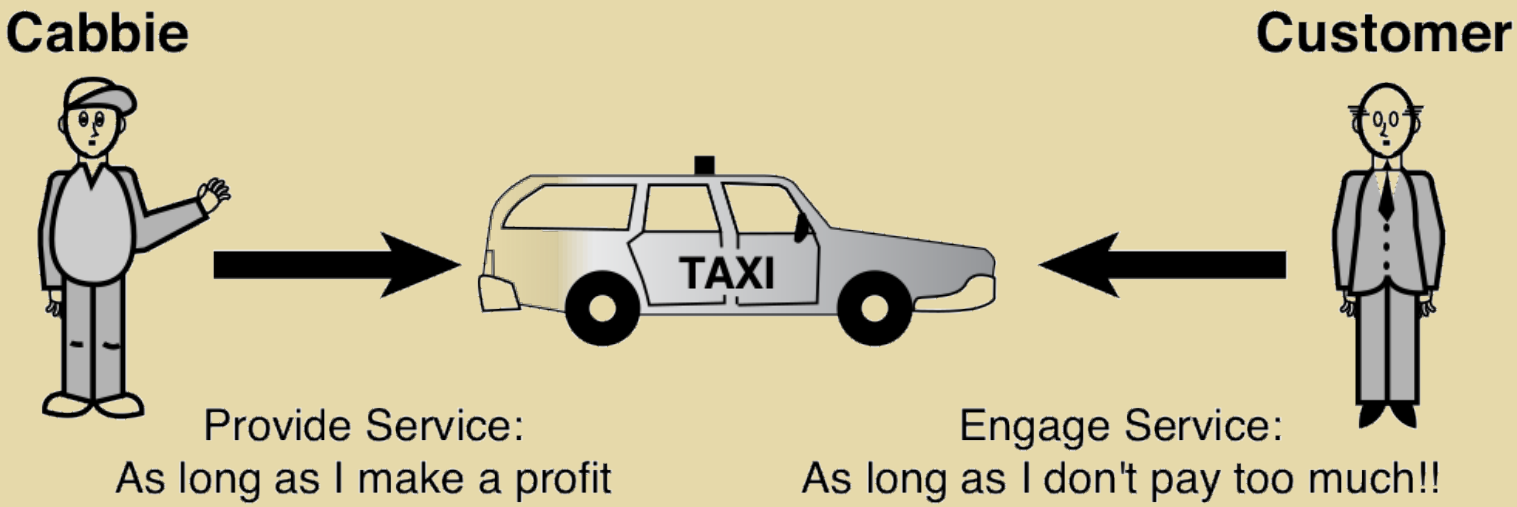

Figure: Any object that sends a message to the taxi object is considered a user (figure 2.6 of [1])

Providing the Absolute Minimal User Interface Possible

- “Determining the Users
- Object Behavior
- Environmental Constraints
- Identifying the Public Interfaces
- Identifying the Implementation"[13]

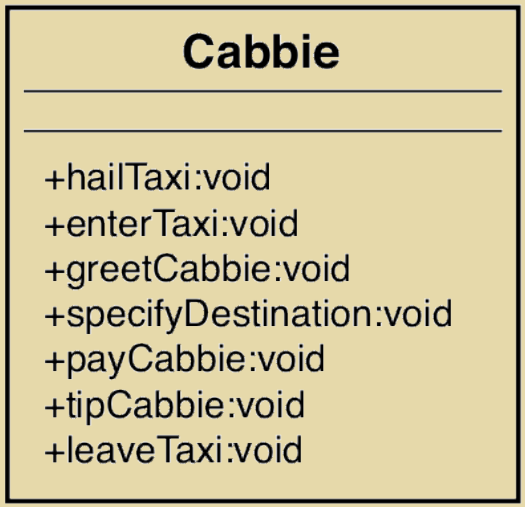

Figure: The methods in a cabbie class (figure 2.7 of [1])

[13] Pag. 48-50 of [1]

References

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