Object-Oriented Though Process

Covered topics

1. Interface vs Implementation
2. Abstract thinking
3. Minimal Interface

Object-Oriented Design

- A desired OO design result

A **robust** and functional object model

- Don’t aim to reach a perfect design in the first time
- Before starting to design, think the problem through

Structured and Object-Oriented Design

Structured and object-oriented development coexist

- Wrappers
- Structured constructs are everywhere
  - Loops, if-statements, …

But OO design requires a new though process

Object-Oriented Though Process

- Knowing the difference between the interface and the implementation
- Thinking more abstractly
- Giving the user the minimal interface possible

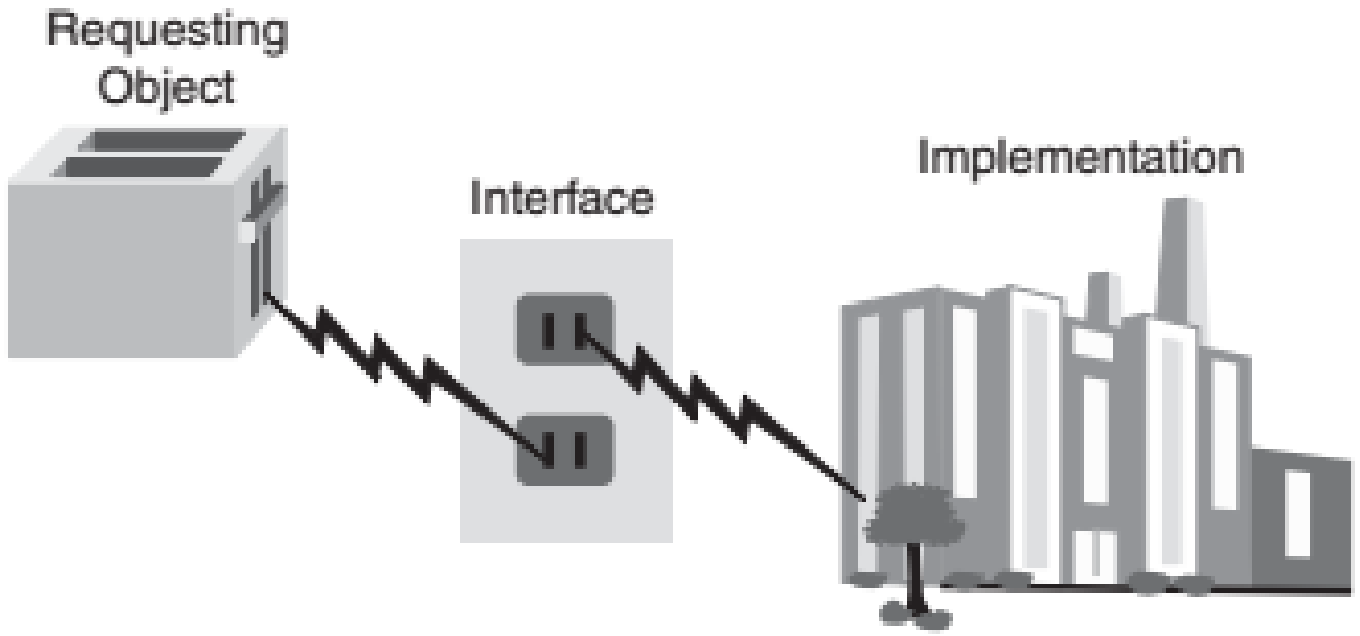
Interface vs Implementation

What the user needs to know

vs

What the user does not need to know

Interface vs Implementation



Motorbike

- What are the components of the interface between you and the motorbike?
  - The handlebars, the brake, the pedal
- The implementation of your motorbike?
  For average users, it is of little concern.
- However, any driver would recognize and know how to use the handlebars – the common interface

Changing the implementation

- What if the manufacturer replaced

  Mechanical fuel injection by

  Electronic fuel injection

→ Average driver would not notice

Implementations should be/are interchangeable

Interchangeable implementation

The interchangeable implementation must be

identical in every way

with respect to the interface

Otherwise, it would be a big problem like

Changing the current from AC to DC

Interface vs Implementation: Changing

- A change in the implementation should have no impact on the users

- A change in the interface might impact on the users

Changing the Interface

- What if the manufacturer replaced

End users vs Programmers

- End users see GUI – users of an application
- Programmers see class interfaces – users of classes

The interface

- The interface: The services presented to user

- The interface to a class should contain ONLY what the user needs to know

The implementation

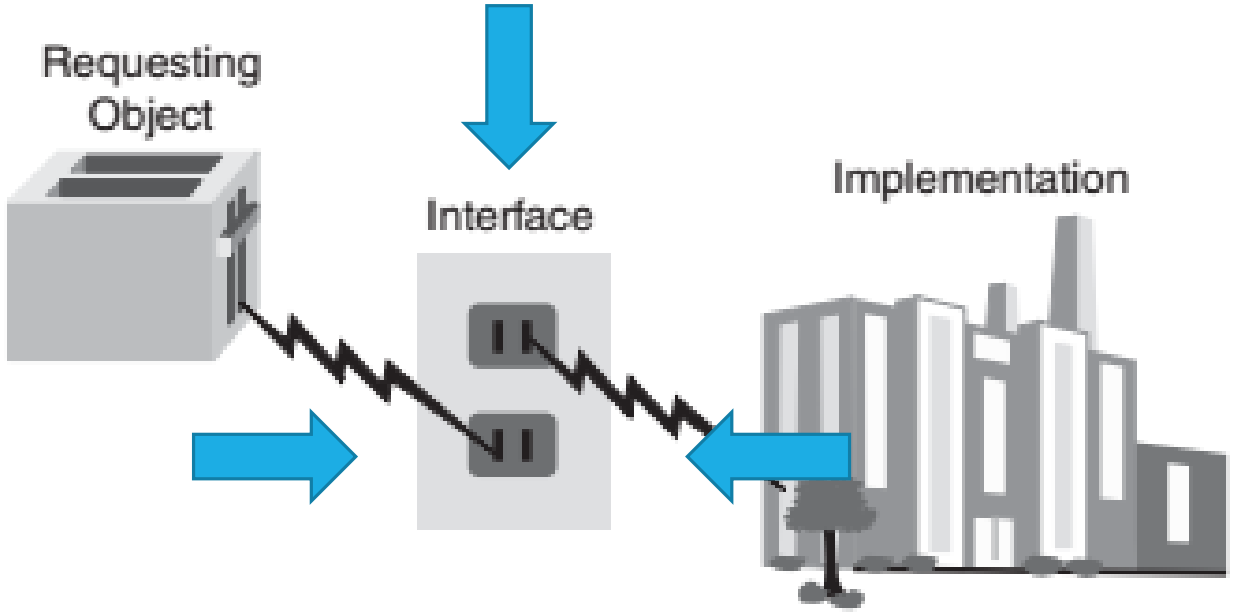
The implementation are hidden from the user

The ultimate goal of the design

A change to the implementation
should not require
a change to the user's code

The interface specification

Both the users and the implementation must conform to the interface



Interface/Implementation example

- In groups, make an initial attempt to

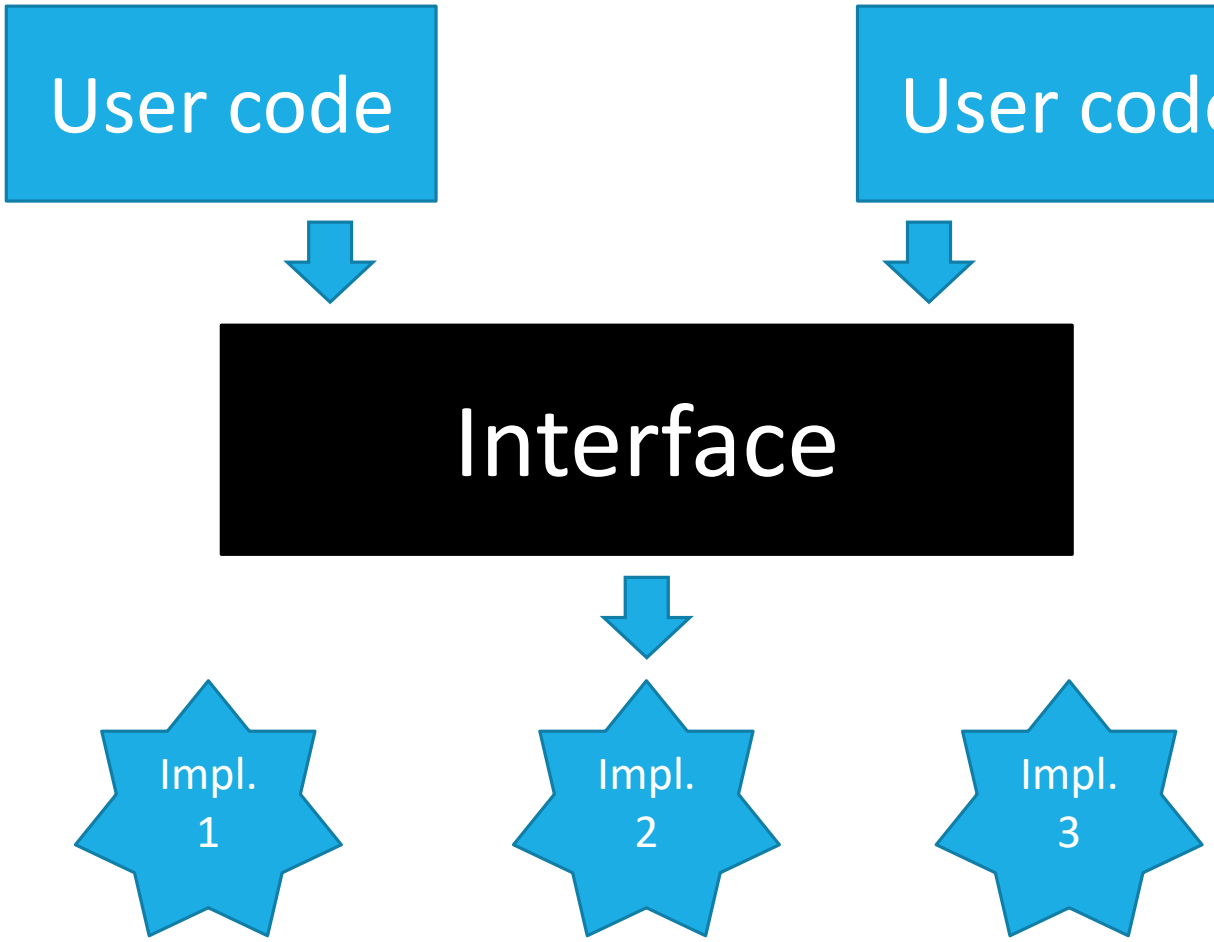
  design an interface(s) for part of your game project

Start from the requirements for the game

Design the interface that programmers will use

  - Application-programming interface – API

Separation of interfaces from implementation



Code recompilation

+ Dynamically loaded classes: like in JAVA, C#

+ Statically linked classes: C++

Usefulness, reusability and levels of abstraction

In OOP classes can be reused

- ***Reusable*** classes tend to have interfaces

  more abstract then concrete

- Highly abstract interface are often more ***useful*** than highly concrete interface, but not always the case
- It is possible to have

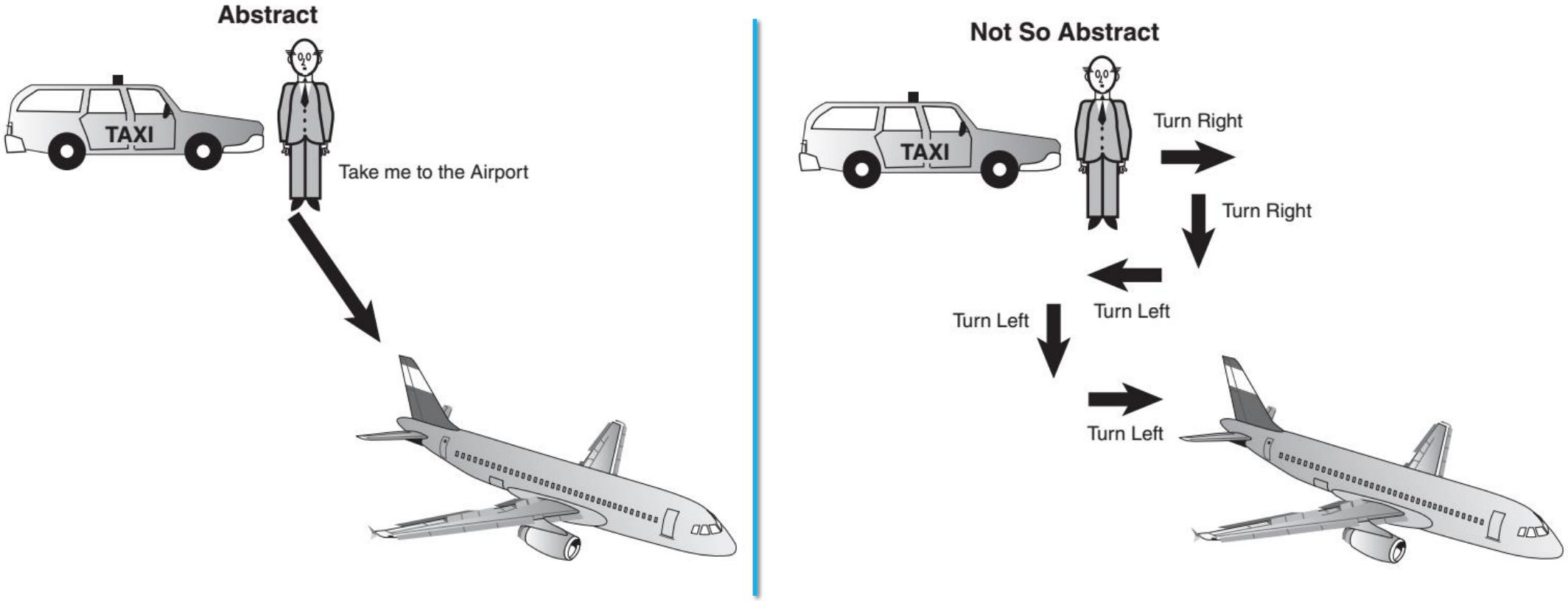
  a very useful, concrete class but not at all reusable

Design goal

To design abstract, highly reusable classes should

Design highly abstract user interfaces

Abstract vs concrete interface



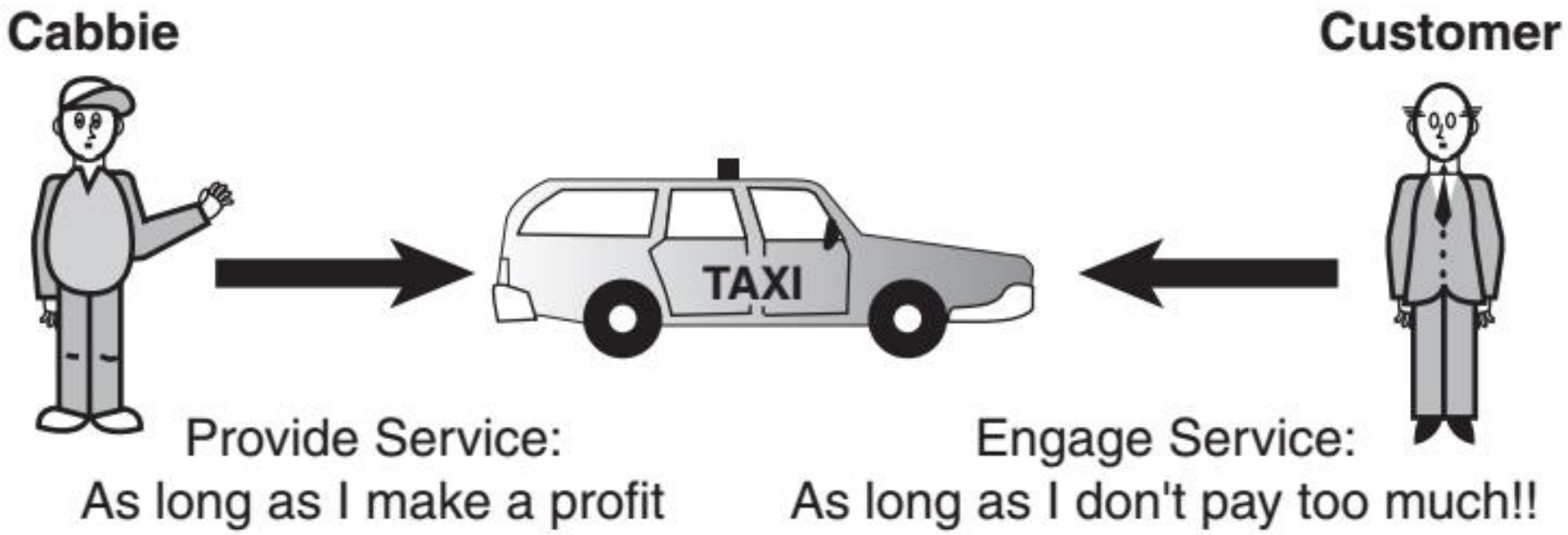
Which one is more reusable ?

Minimal interface possible

Provide as little knowledge of the inner workings of
as possible

- Give only what users need
  Better to have to add than to give more than needed
- Starting from private interfaces
- Design from a user's perspective (not from information systems viewpoint)
- Design a class from requirements
- Design with users of the class

Determining the users



Identify object behaviors

Identify the purpose of each objects and
What it must do to perform properly?

By

Using various methods to gather requirements

Note:
Many of the initial choices will remain in the final public interface

Identify the public interfaces

Think about

- How you would use the object?
- What do you need to do to use the object?

Recommendation

- Each interface (method) models only one behavior

Identify the implementation

Think about

- How to get the class work?

Implementation consists of

- Private methods that are used internally
- The codes within the public methods

Second attempt to design your game project

- Check and rework on the design for your game project

Conclusion

A desired OO design result is

A **robust** and functional object model

And

Designing OO classes is more of an art than a science