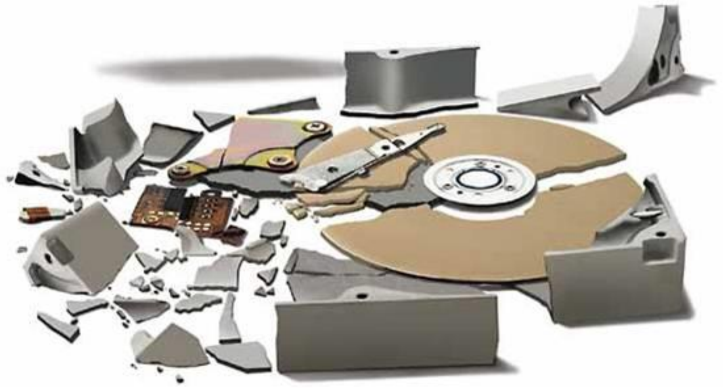
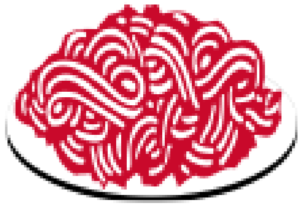


Architectures des Systèmes de Bases de Données

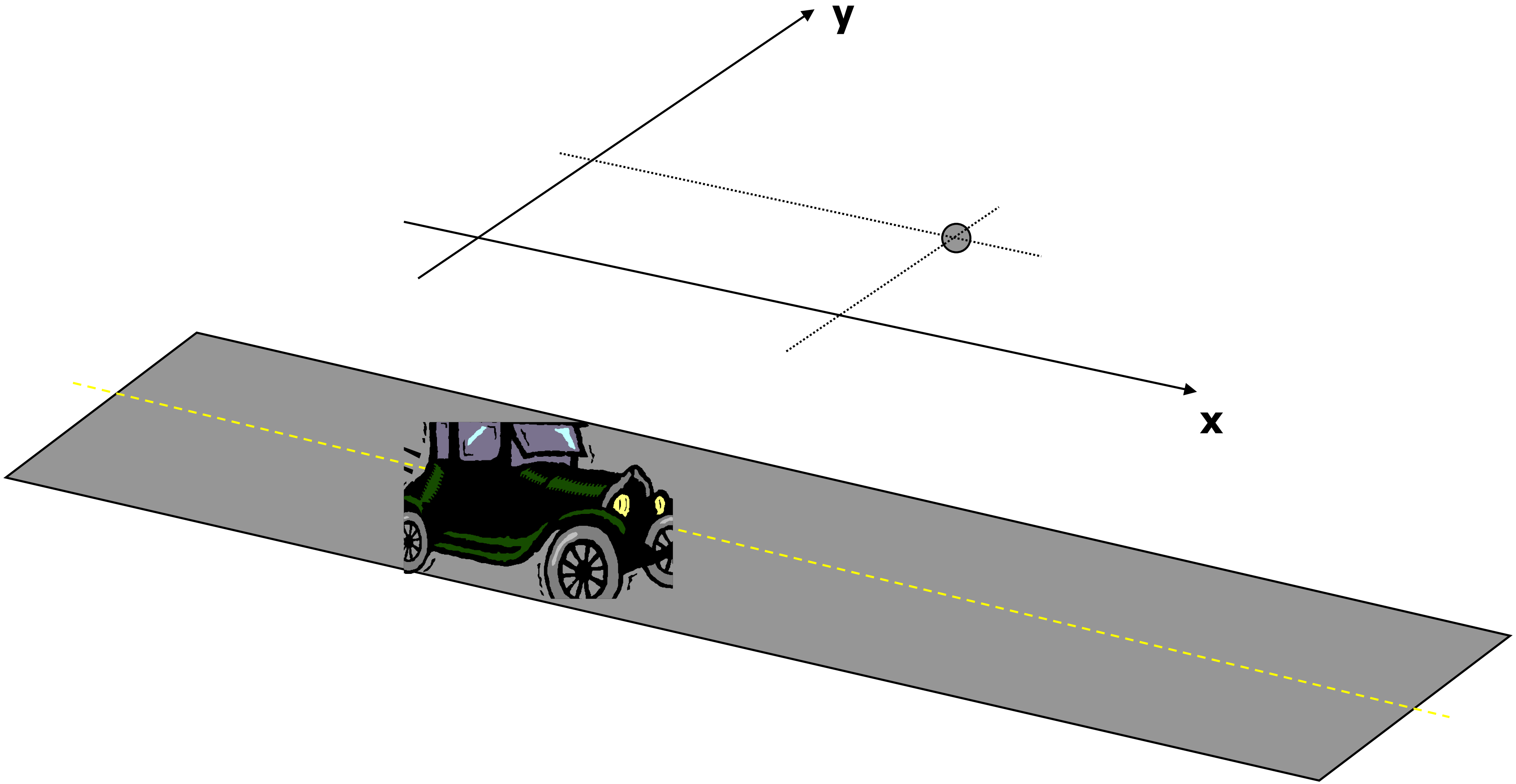
Encapsulation and Interface



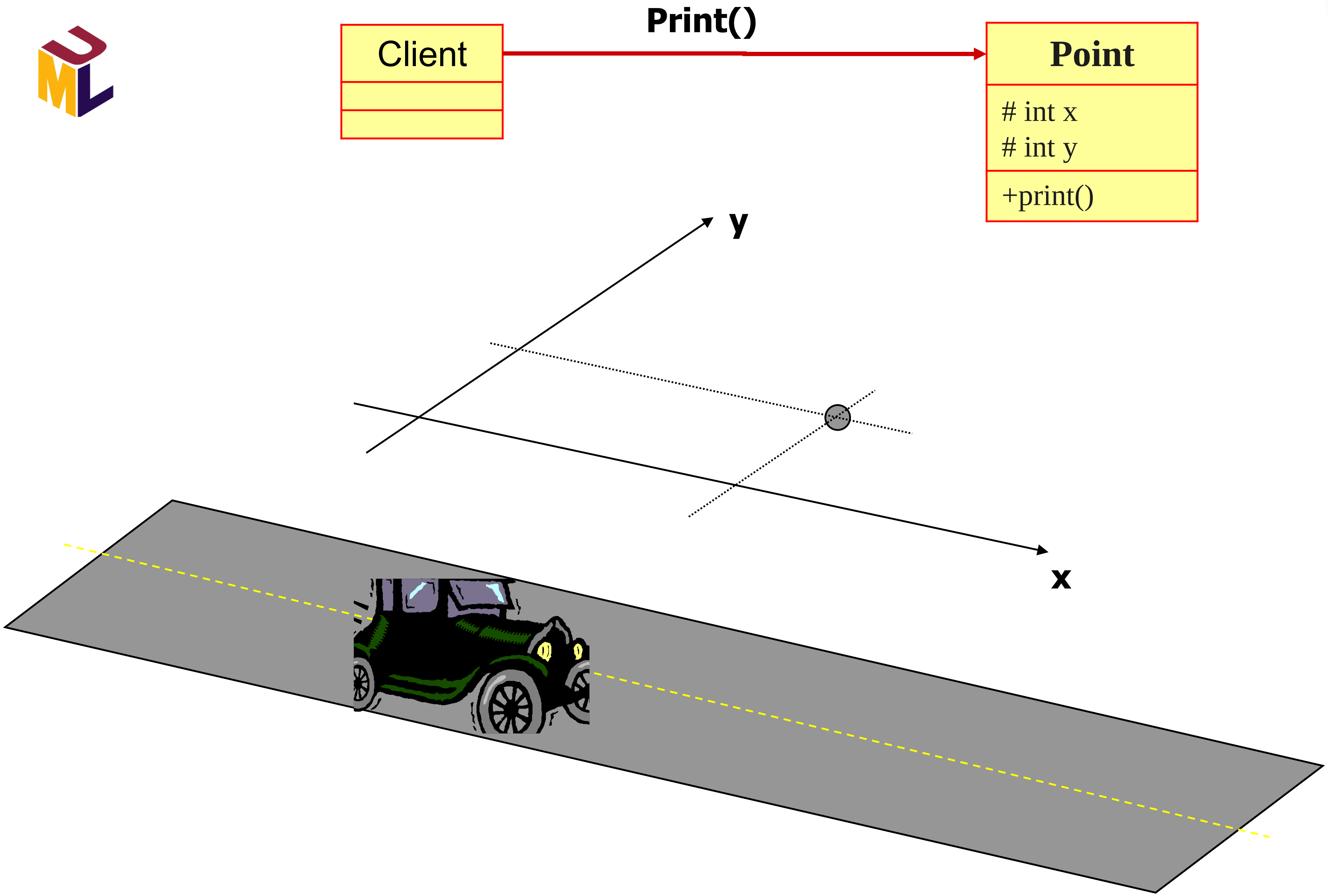
Spaghetti Plate

Traduction en cours

Programming in “present” tense



Programming in “present” tense



Print()

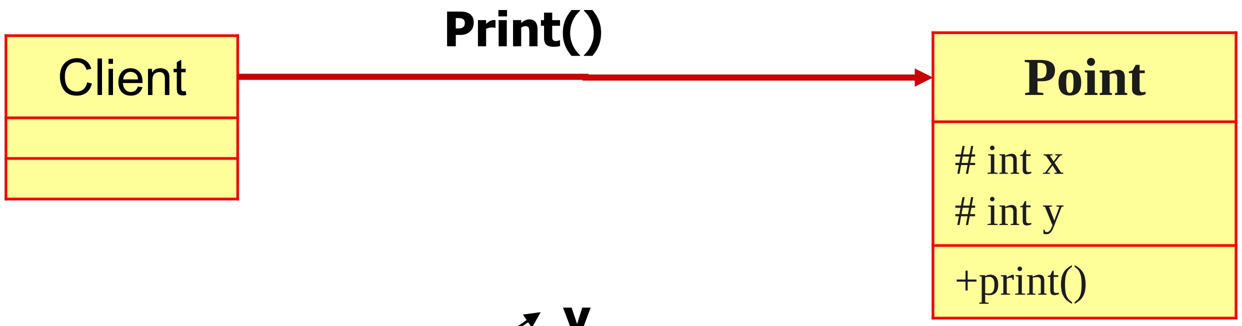
Client

Point

# int x
# int y

+print()

Future



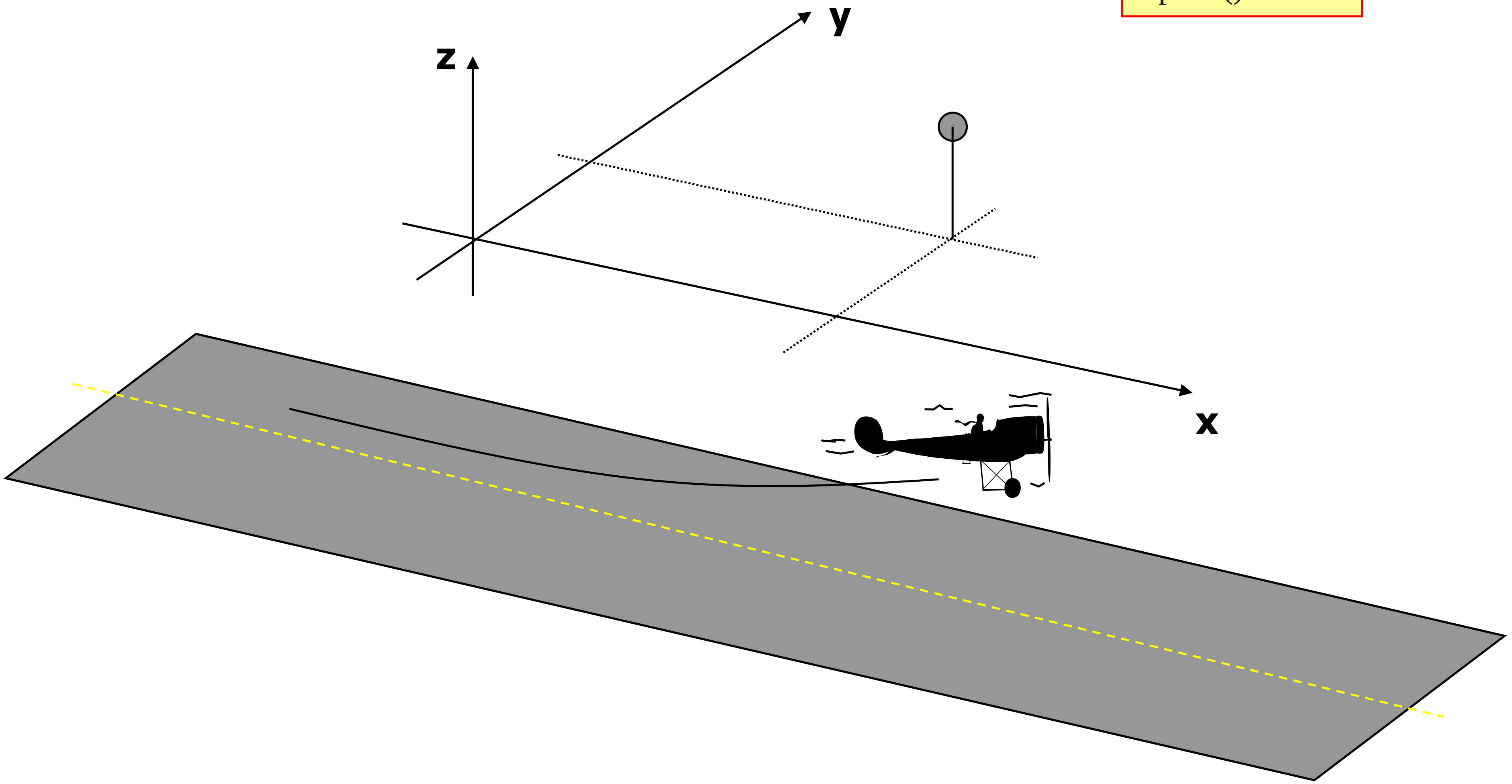
Client

Print()

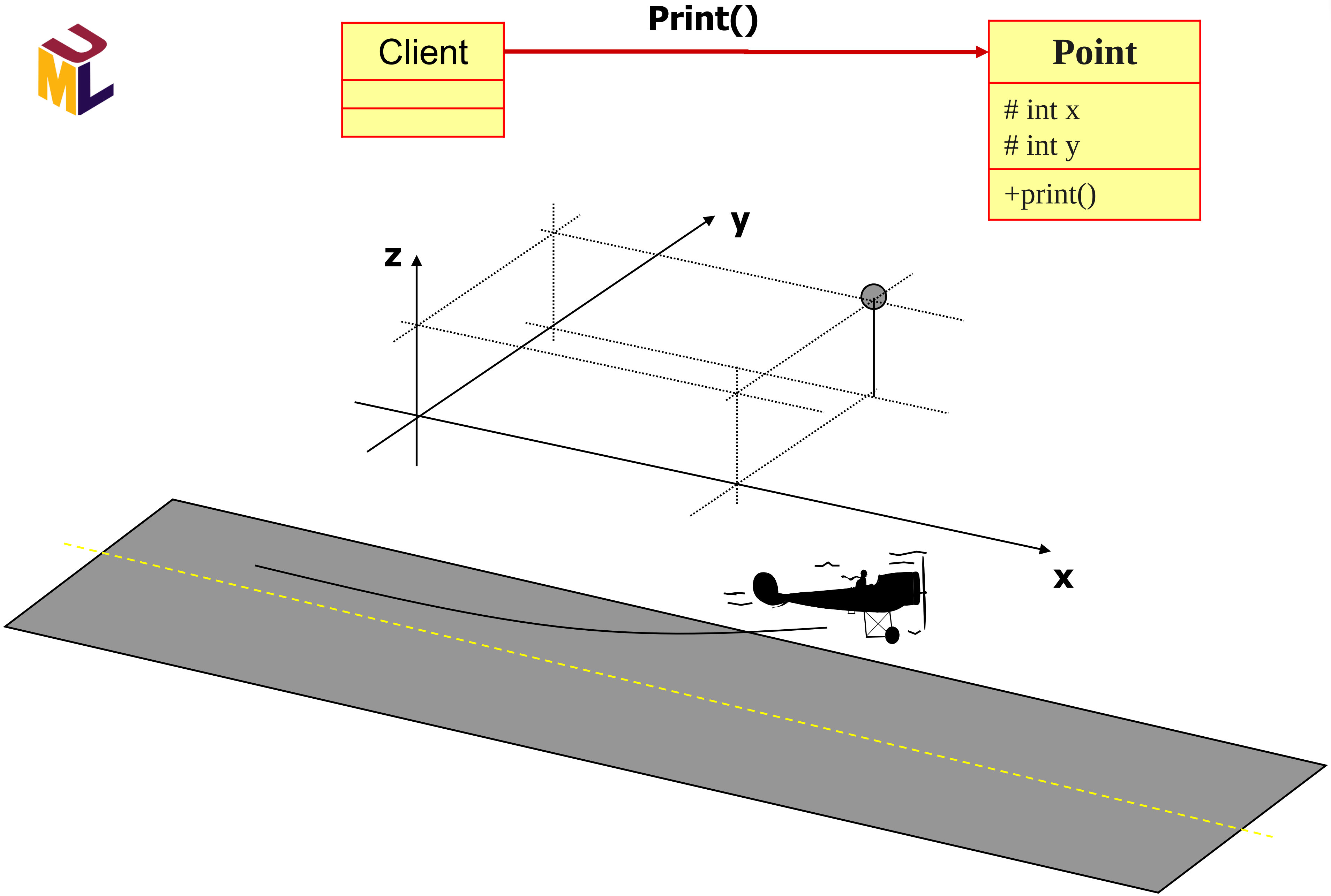
Point

# int x
# int y

+print()



Future



Client

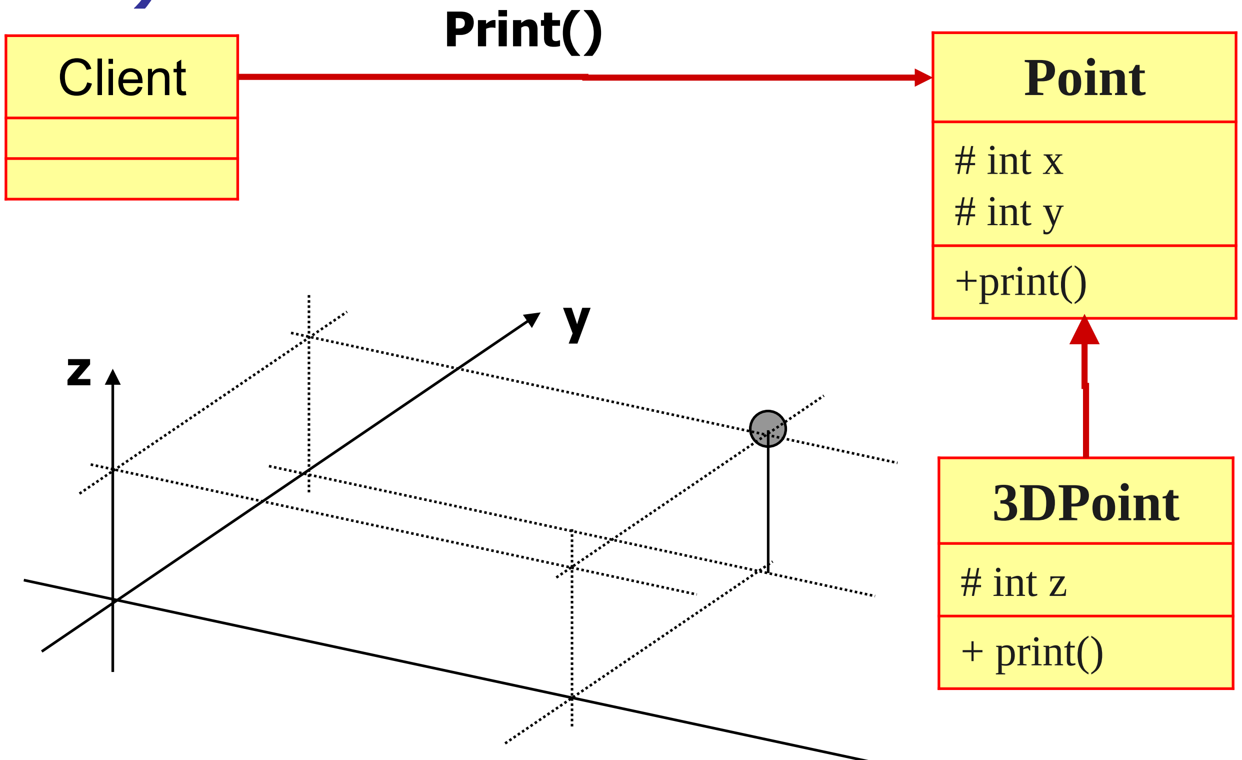
Print()

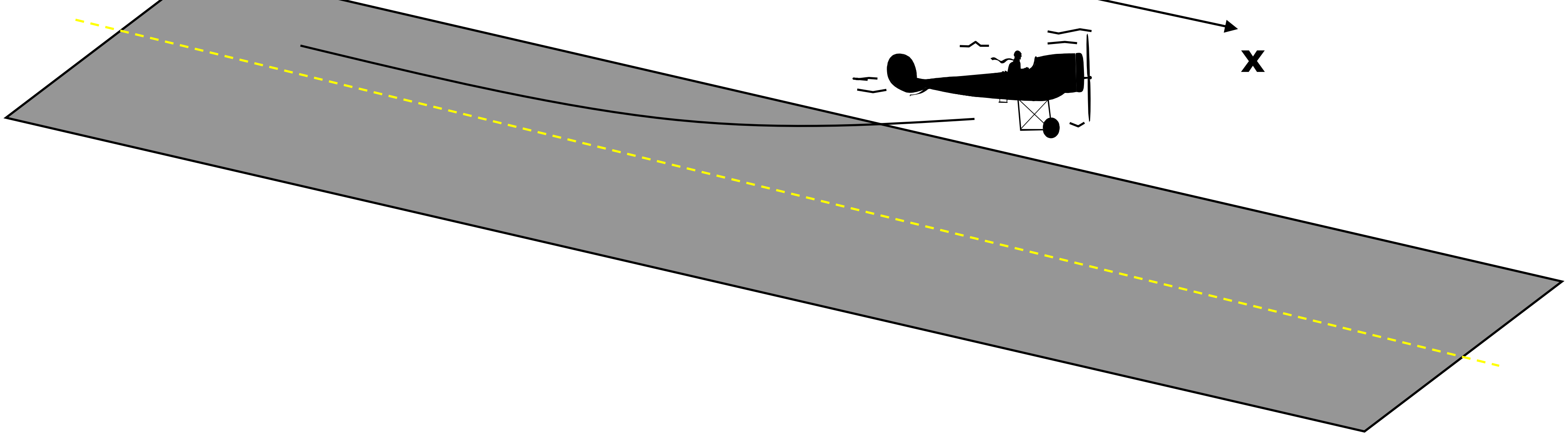
Point

# int x
# int y

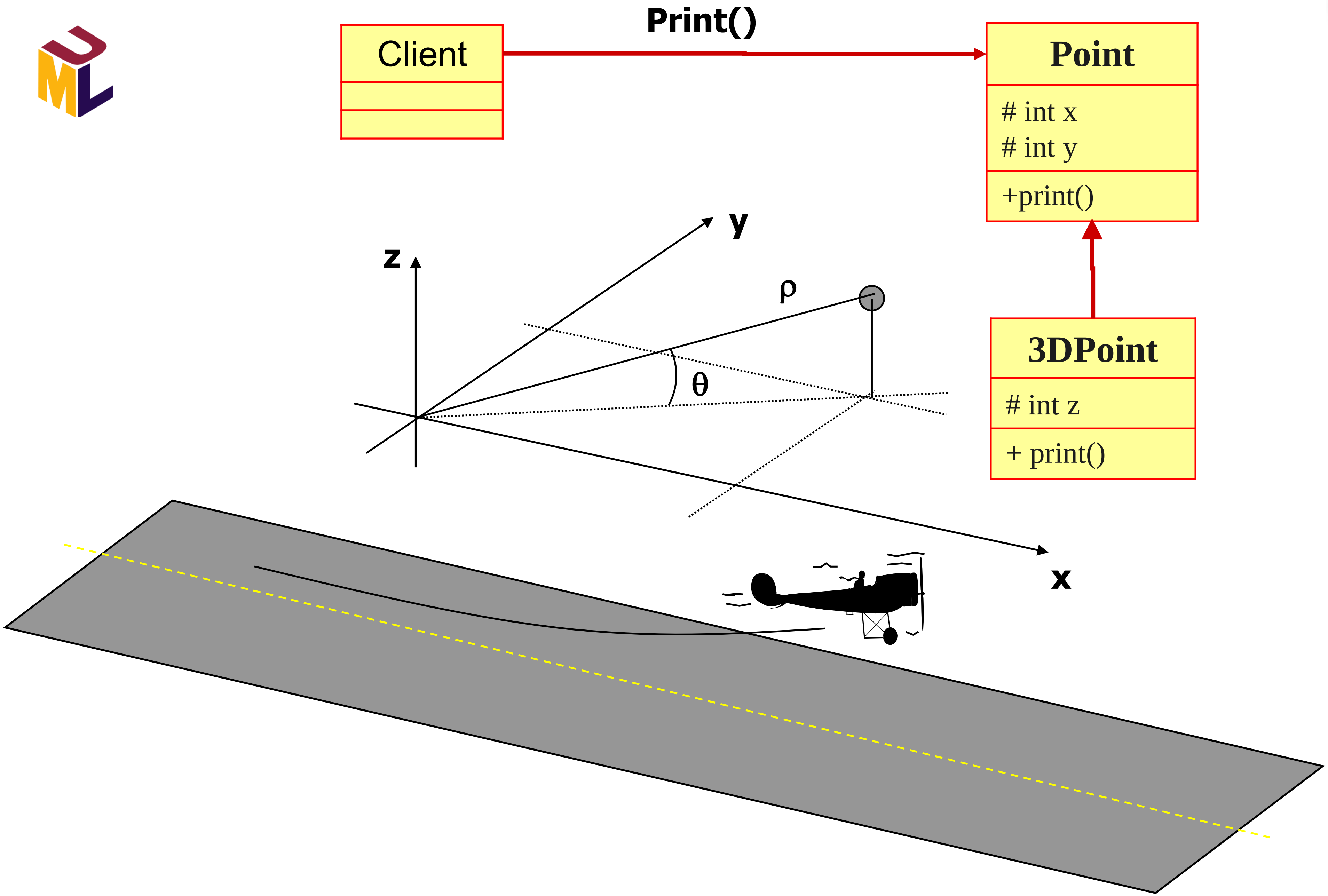
+print()

Programming in "Future" tense (Scott Meyers)

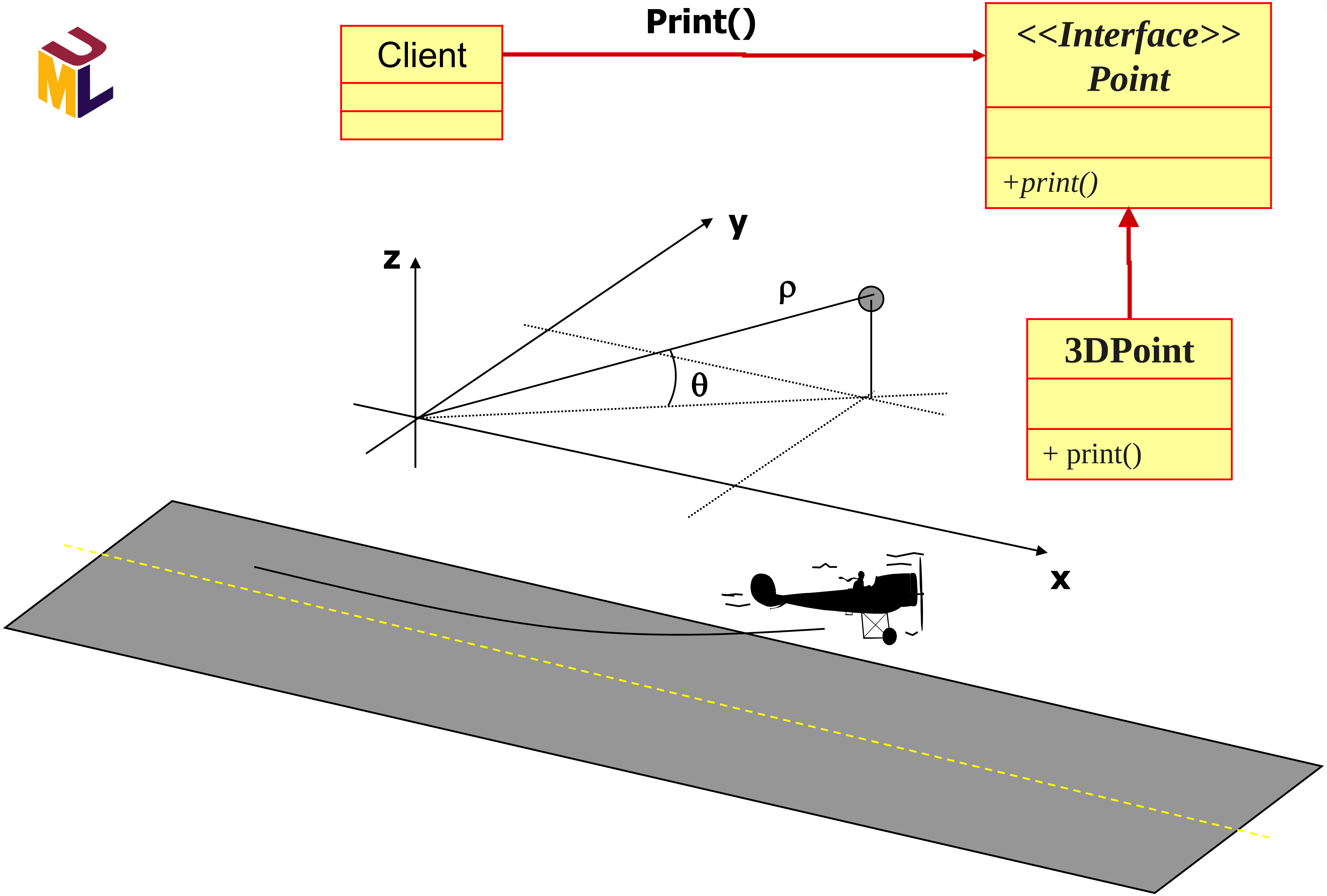




Programming in “Future” tense



Programming in “Future” tense



Client

Print()

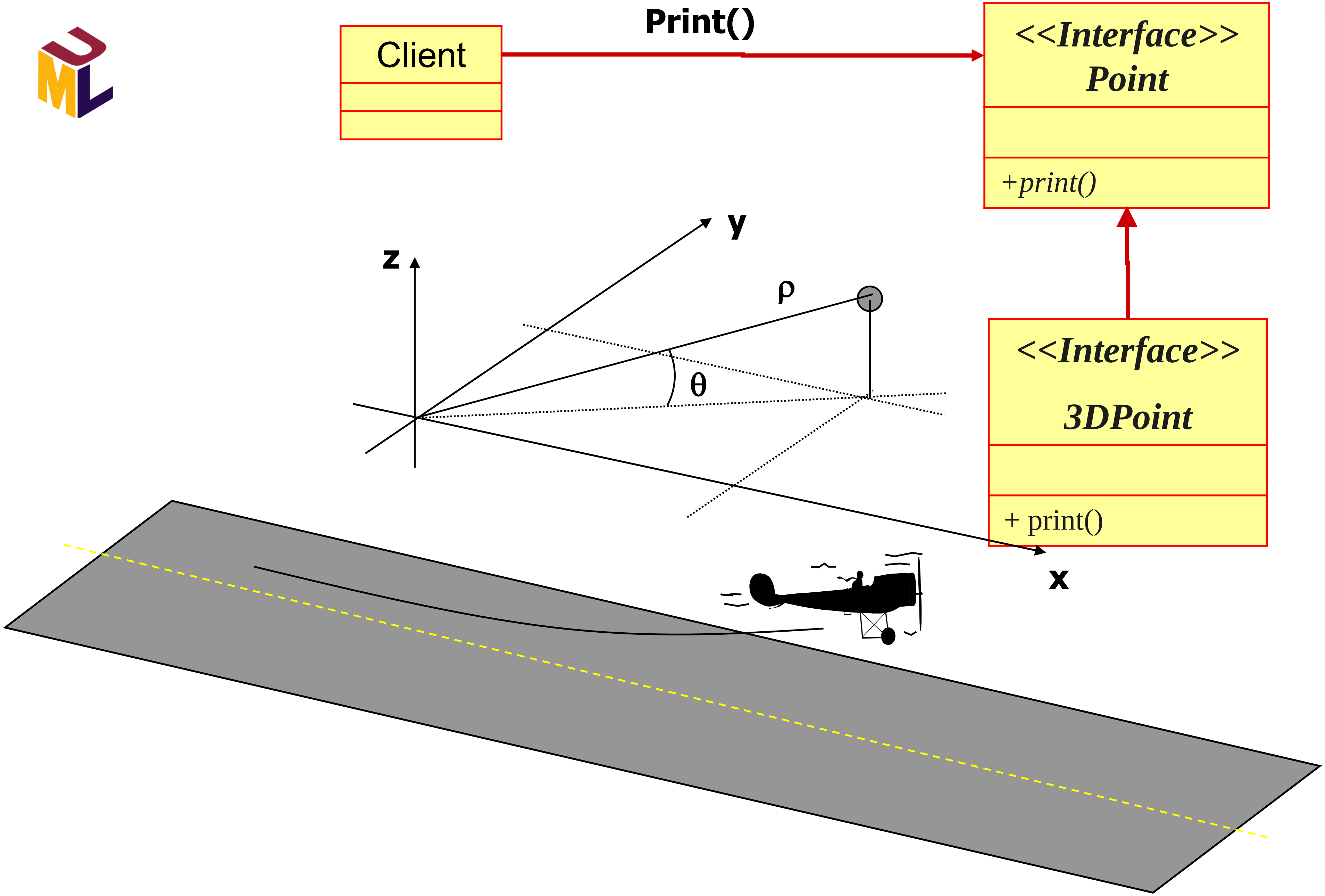
<<Interface>>
Point

+print()

3DPoint

+ print()

Programming in “Future” tense



Client

Print()

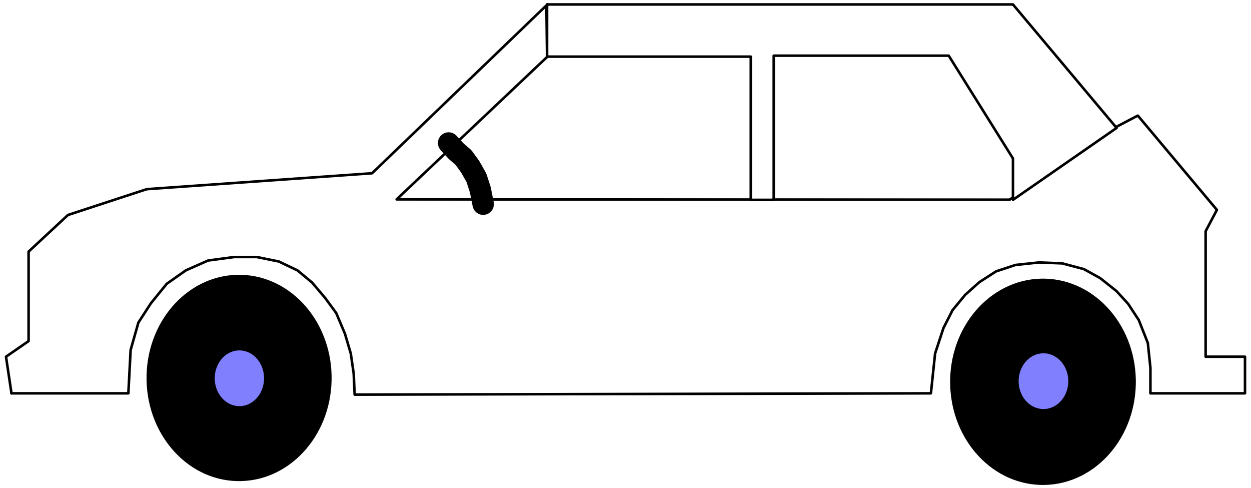
<<Interface>>
Point

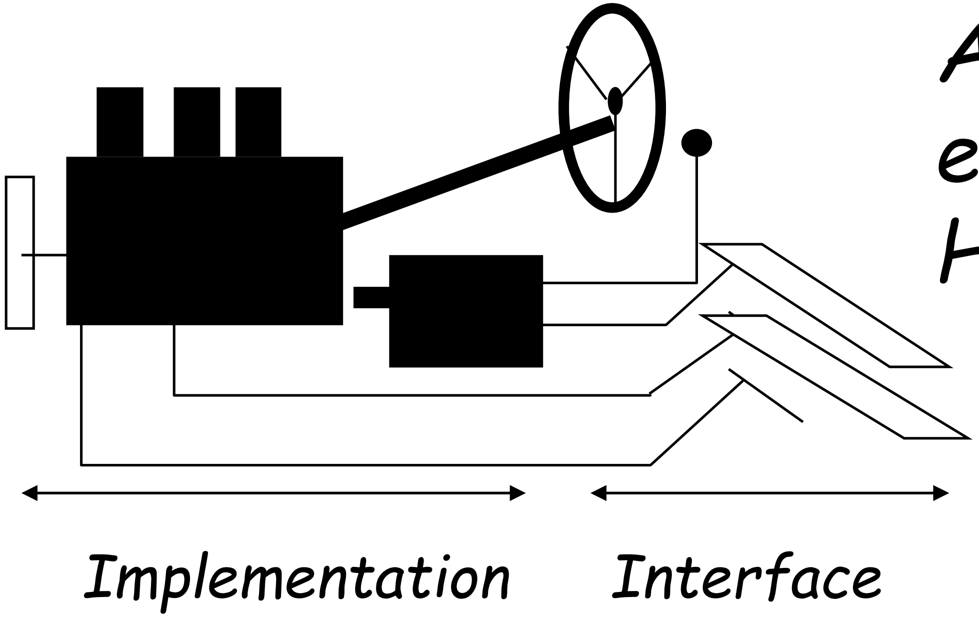
+print()

<<Interface>>
3DPoint

+ print()

Design By and For Interfaces (GOF)





A driver doesn't care of engine's internal working. He only knows the interface

Interface

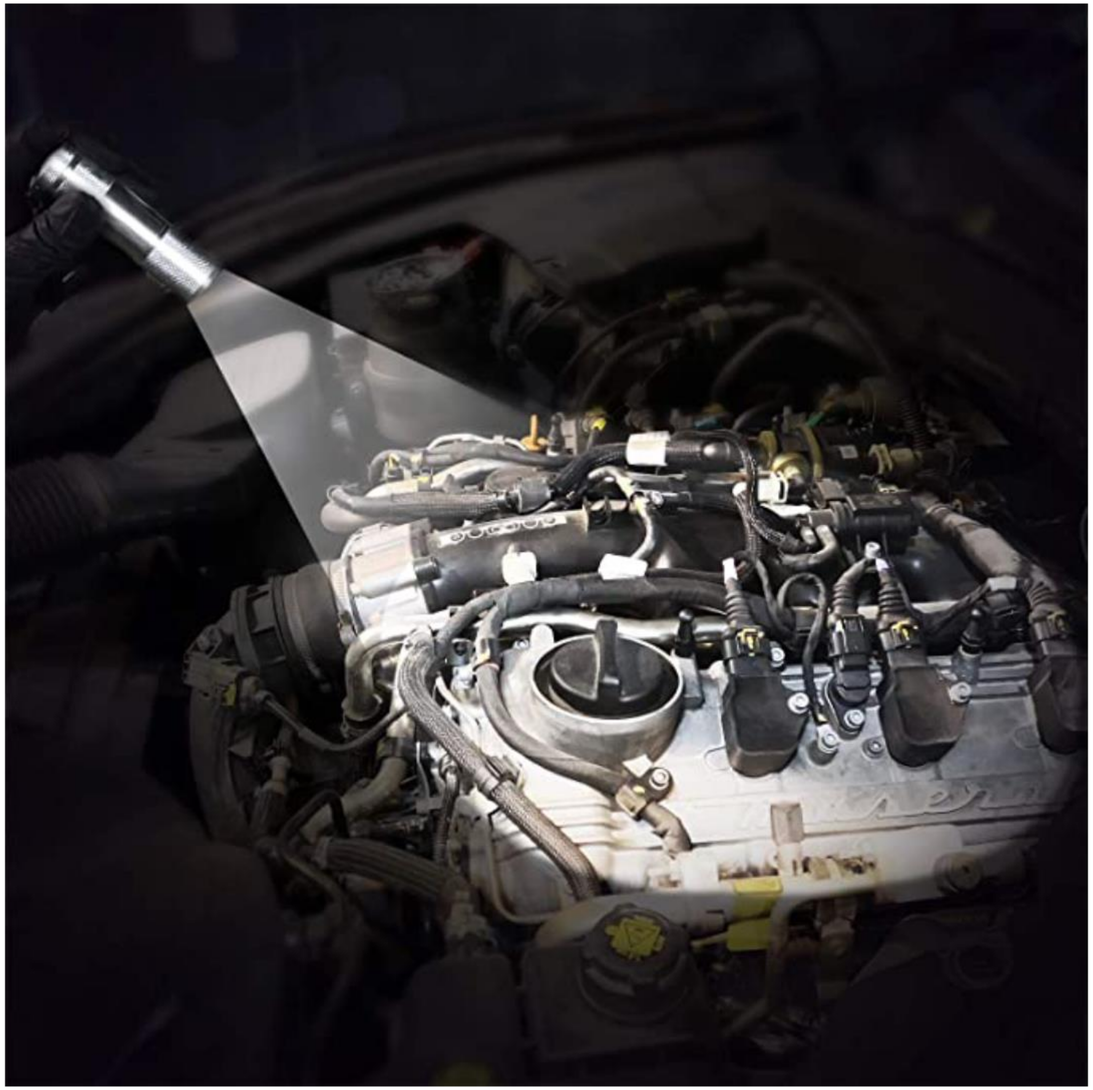
```
interface car {

        start()

        accelerate()

        stop()

}
```

car → start()

Design By and For Interfaces

A driver doesn't care of engine's internal working. He only knows the interface

Design By and For Interfaces

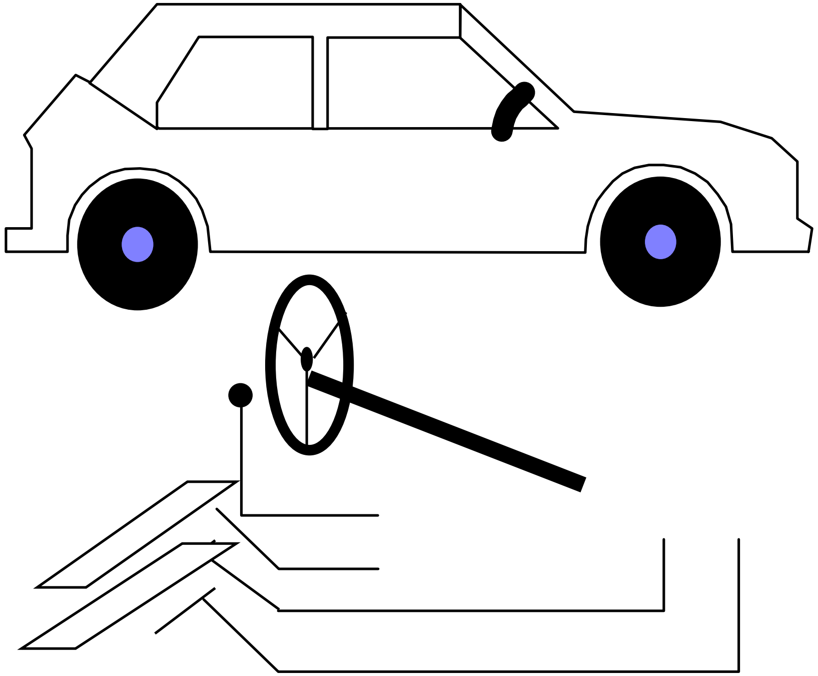


A driver doesn't care of engine's internal working. He only knows the interface

Design By and For Interfaces

A driver doesn't care of engine's internal working. He only knows the interface

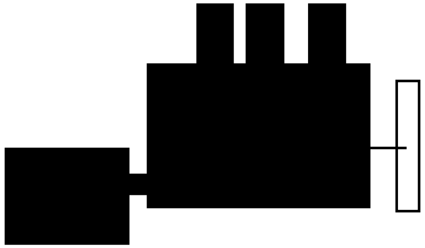
Interface Versus Implementation



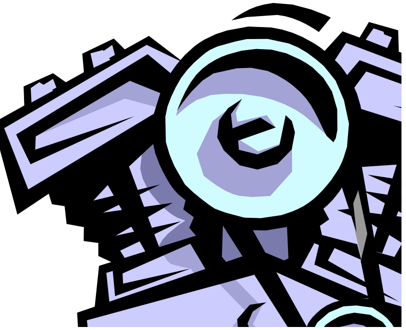
Interface

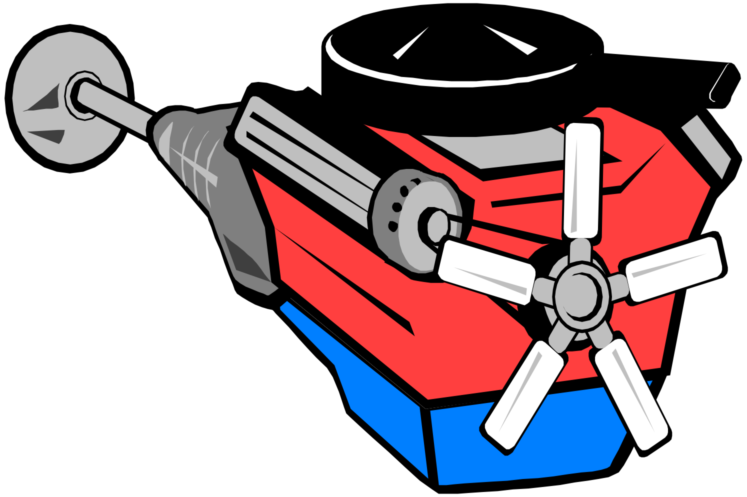
(specification)

```
Struct {
       Int field1
       double field2
       abc field3
       xxx field4
}
```



```
00101010001101001
00101111010101010
01001010010101010
01001110110010101
01010001010111111
00001001001001000
```

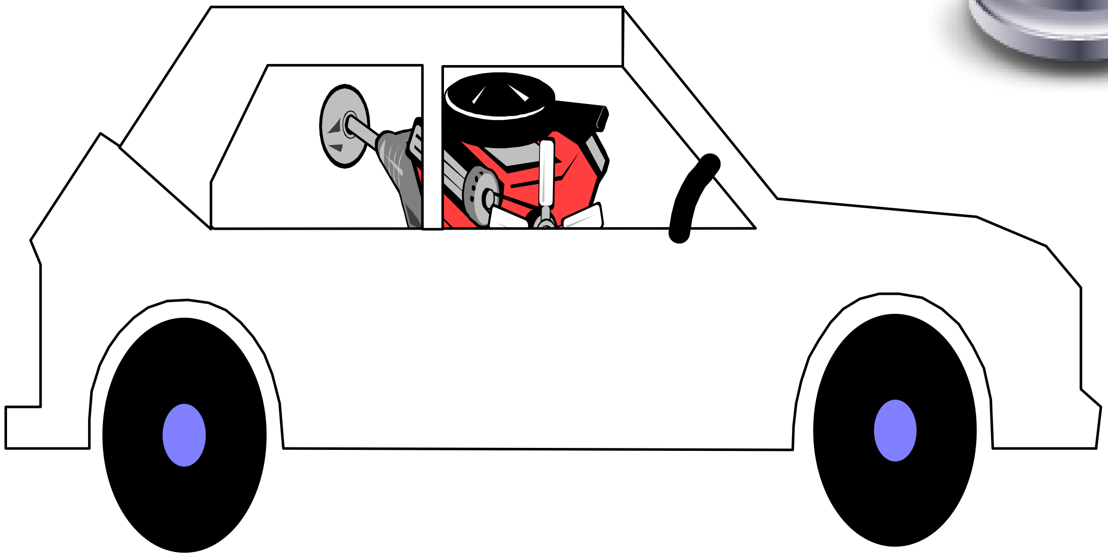




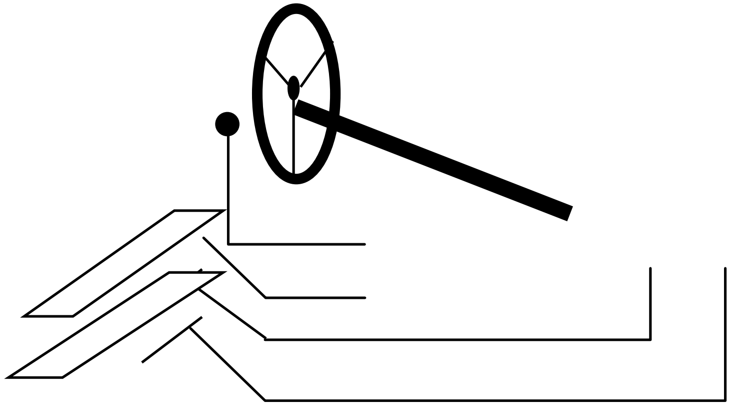
Implementation

(body)

Interface Versus Implementation

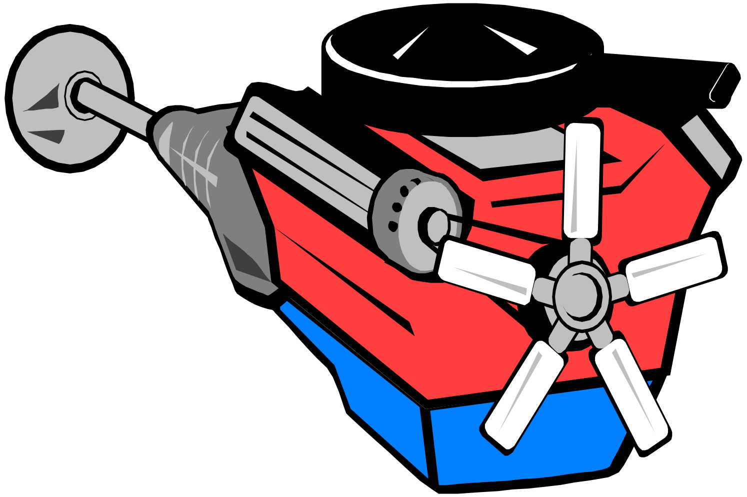


No Implementation details In the Interface



Interface

(specification)

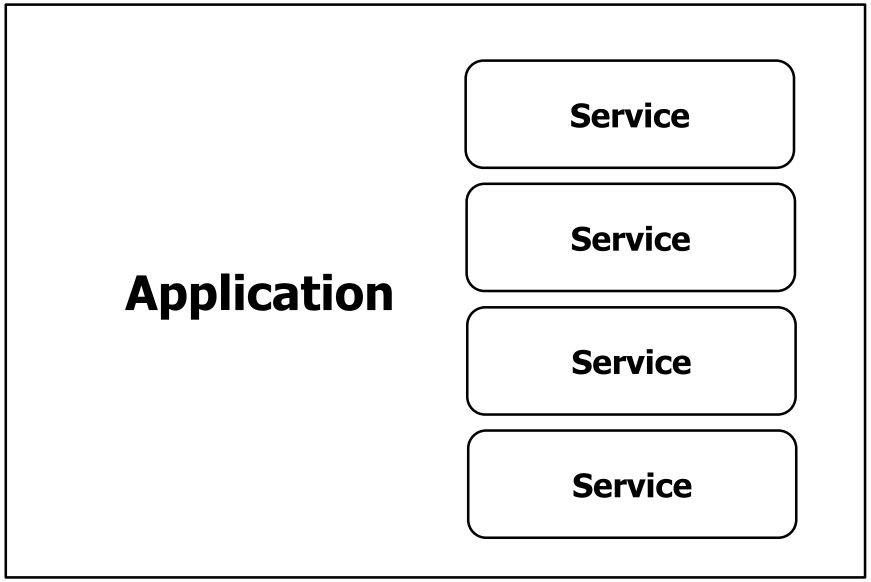


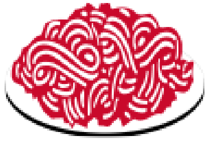
Implementation

(body)

```
Struct {
        Int field1
        double field2
        abc field3
        xxx field4
}
```

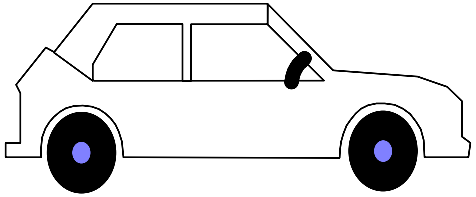
Software Application Architecture





Spaghetti Plate

Software Application Architecture

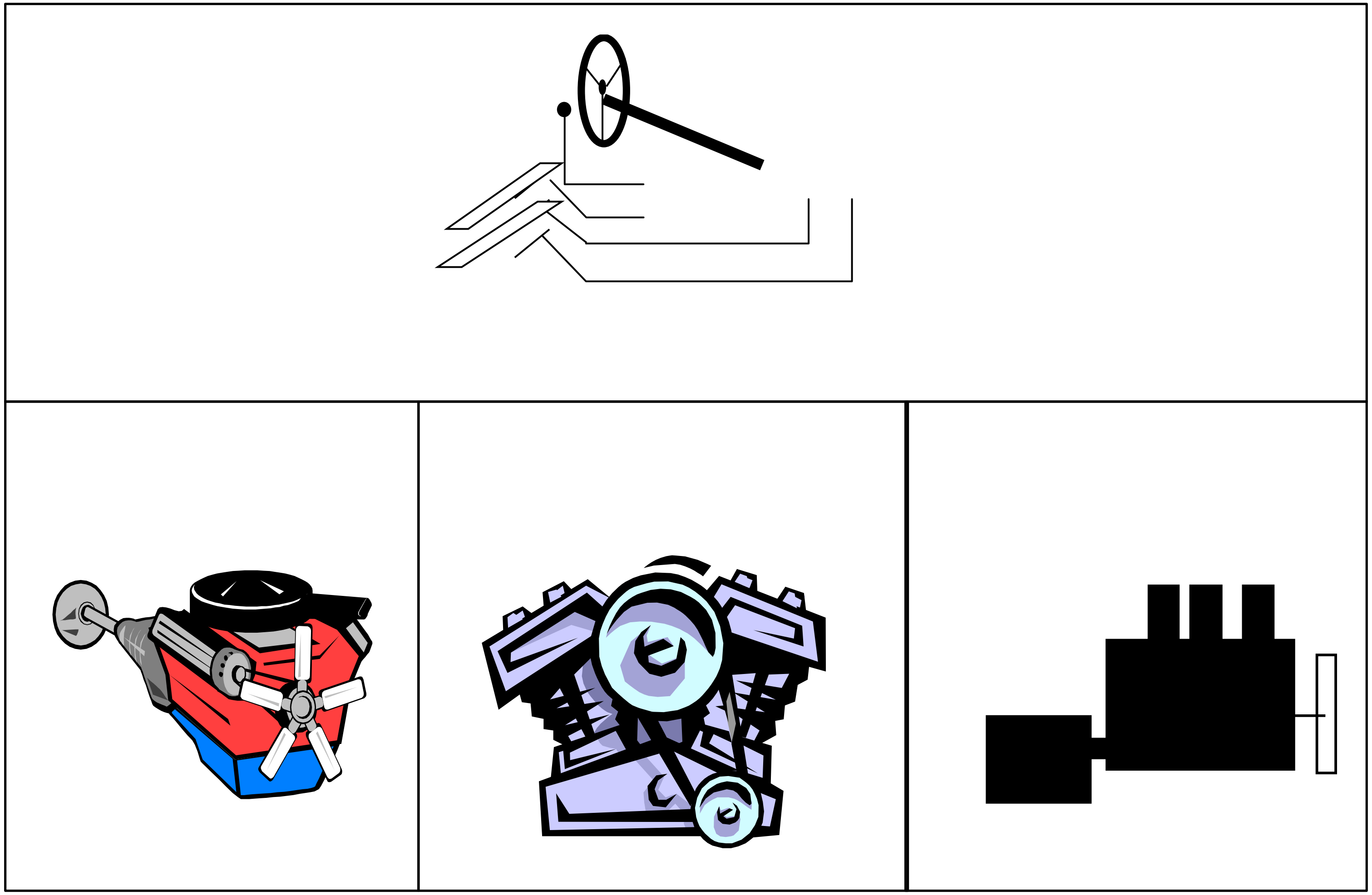


Interface

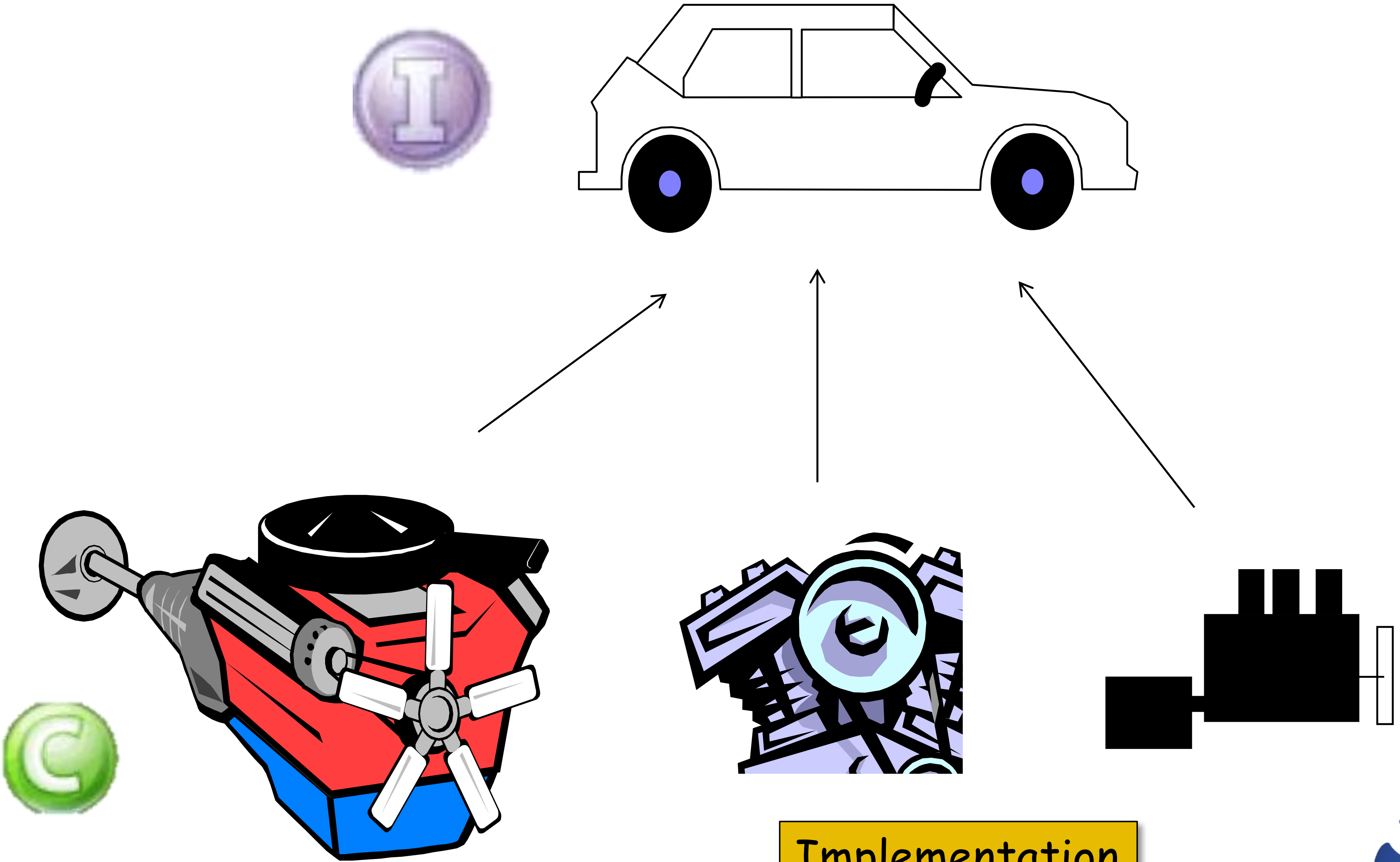
```
Struct {
        Int field1
        double field2
        abc field3
        xxx field4
}
```

Implementation

```
00101010001101001
00101111010101010
01001010010101010
01001110110010101
01010001010111111
00001001001001000
```

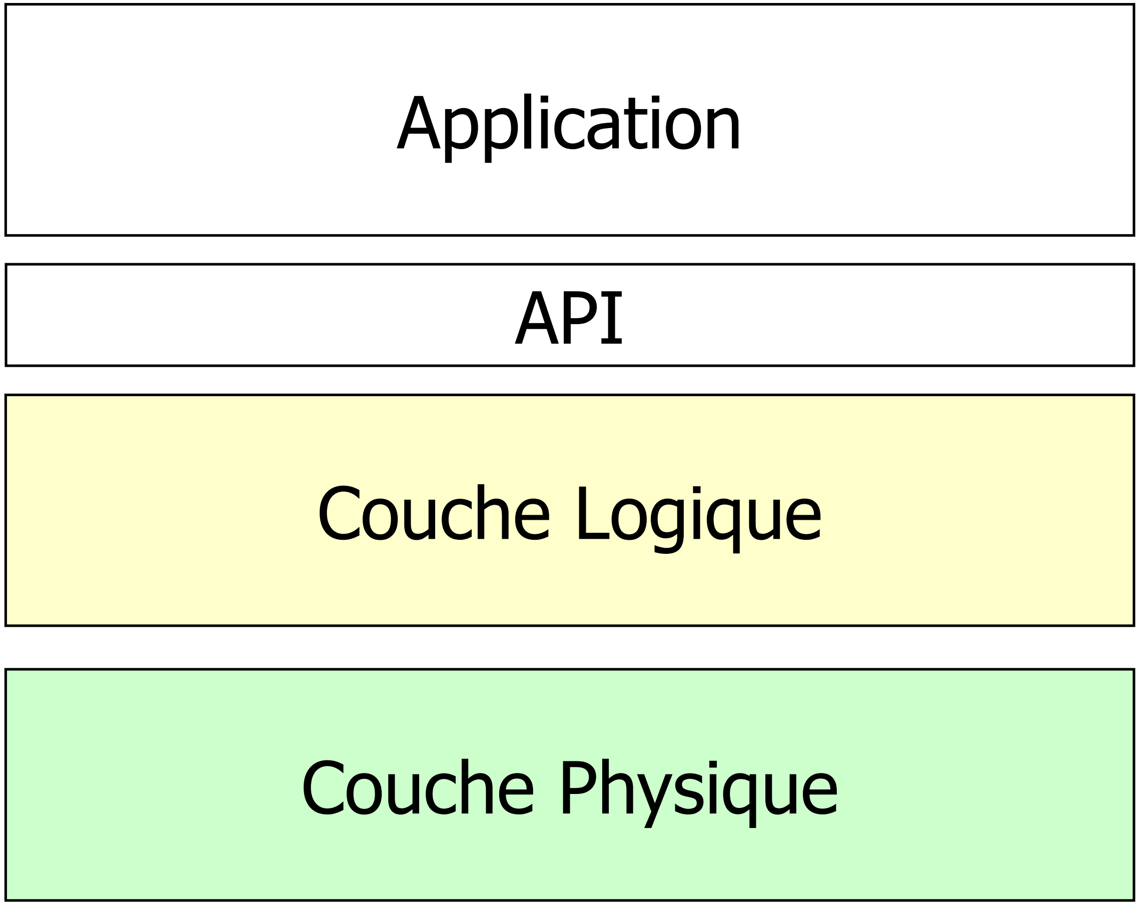


Interface Versus Implementation

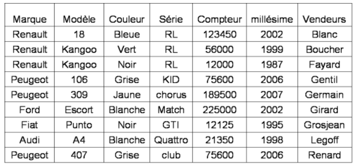


Implementation

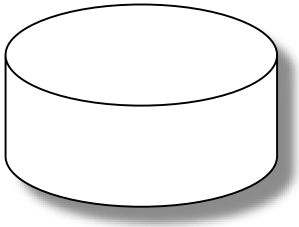
Abstraction en couches (vues)



SQL



Marque	Modèle	Couleur	Série	Compteur	millésime	Vendeurs
Renault	18	Bleue	RL	123450	2002	Blanc
Renault	Kangoo	Vert	RL	56000	1999	Boucher
Renault	Kangoo	Noir	RL	12000	1987	Fayard
Peugeot	106	Grise	KID	75600	2006	Gentil
Peugeot	309	Jaune	chorus	189500	2007	Germain
Ford	Escort	Blanche	Match	225000	2002	Girard
Fiat	Punto	Noir	GTI	12125	1995	Grosjean
Audi	A4	Blanche	Quattro	21350	1998	Legoff
Peugeot	407	Grise	club	75600	2006	Renard



API : Application Programming Interface

Traduction en cours

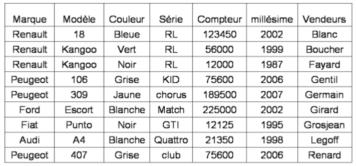
Abstraction en couches (vues)

No Implementation details
In the Interface

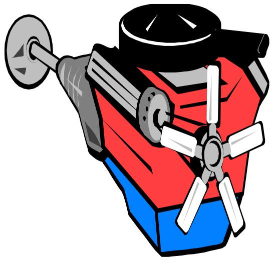
Application

API

SQL

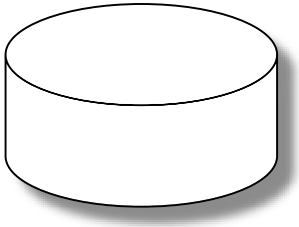


Marque	Modèle	Couleur	Série	Compteur	millésime	Vendeurs
Renault	18	Bleue	RL	123450	2002	Blanc
Renault	Kangoo	Vert	RL	56000	1999	Boucher
Renault	Kangoo	Noir	RL	12000	1987	Fayard
Peugeot	106	Grise	KID	75600	2006	Gentil
Peugeot	309	Jaune	chorus	189500	2007	Germain
Ford	Escort	Blanche	Match	225000	2002	Girard
Fiat	Punto	Noir	GTI	12125	1995	Grosjean
Audi	A4	Blanche	Quattro	21350	1998	Legoff
Peugeot	407	Grise	club	75600	2006	Renard



Couche Logique

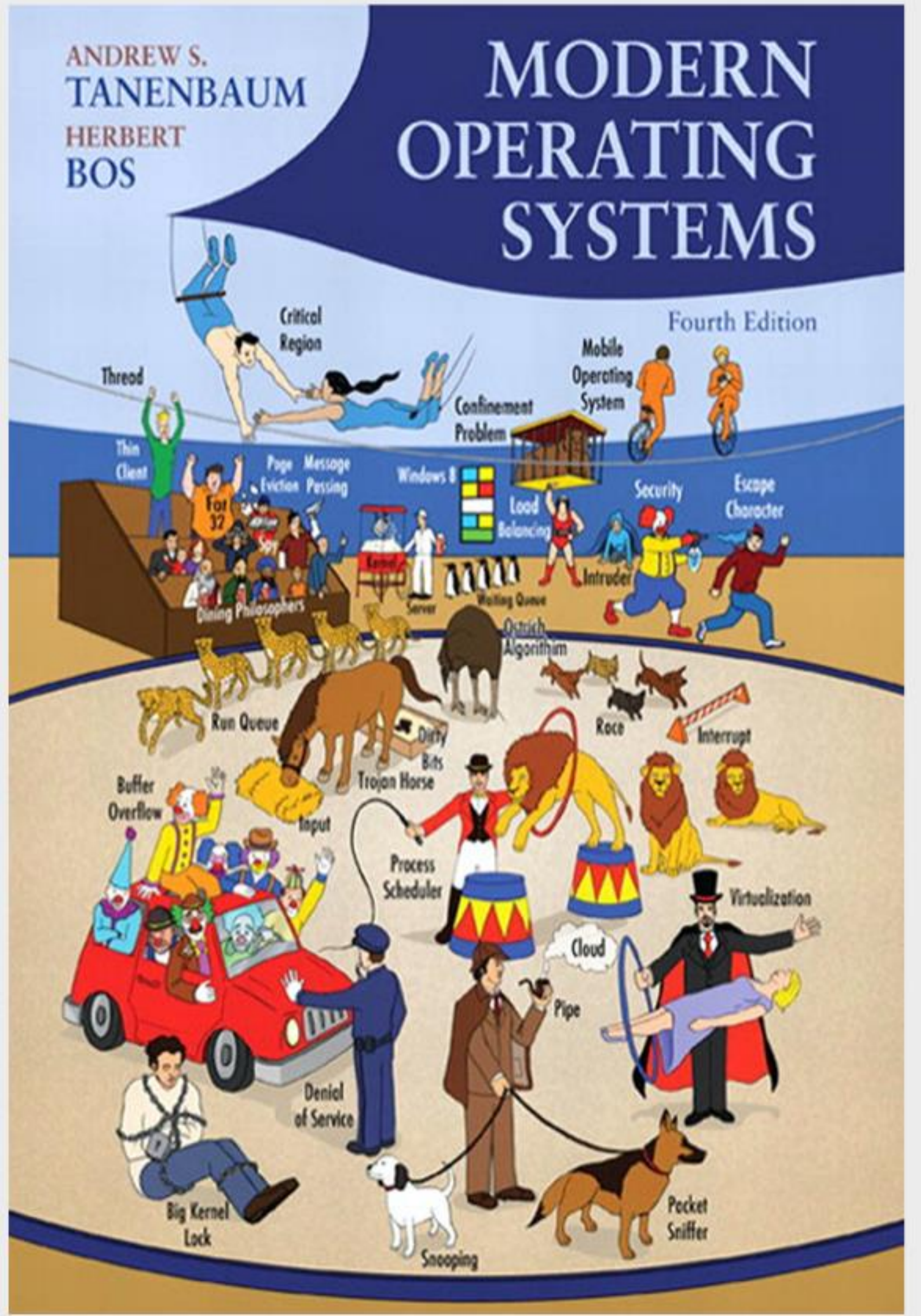
Couche Physique



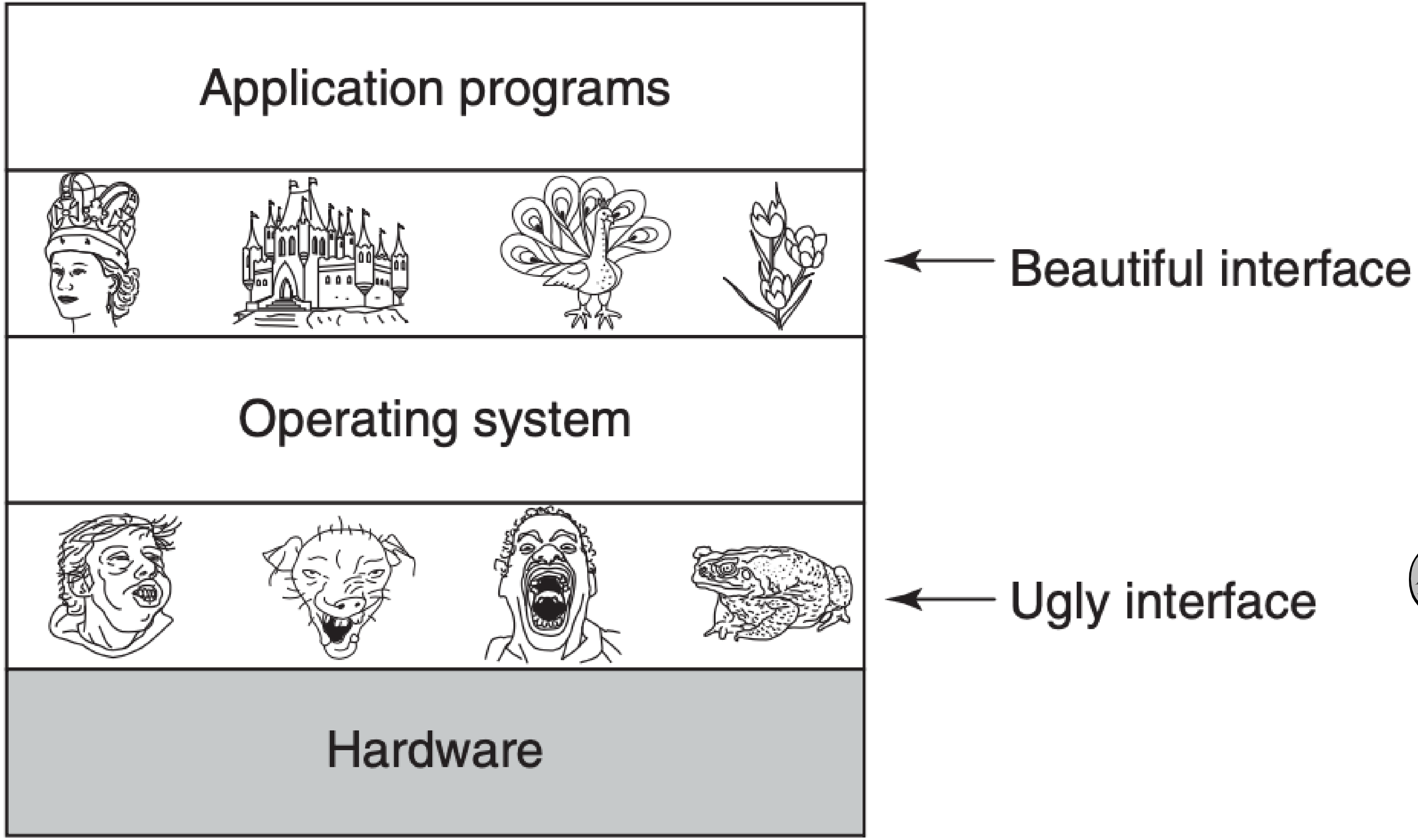
API : Application Programming Interface

Traduction en cours

A.S.T


ANDREW S. TANENBAUM
HERBERT BOS
MODERN OPERATING SYSTEMS
Fourth Edition
Critical Region
Thread
Thin Client
Page Eviction
Message Passing
Fat 32
Spy
Dining Philosophers
Windows 8
Confinement Problem
Mobile Operating System
Load Balancing
Security
Escape Character
Kernel
Server
Waiting Queue
Intruder
Ostrich Algorithm
Run Queue
Dirty Bits
Trojan Horse
Race
Interrupt
Buffer Overflow
Input
Process Scheduler
Virtualization
Cloud
Pipe
Denial of Service
Big Kernel Lock
Snooping
Packet Sniffer

Abstraction en couches (vues)



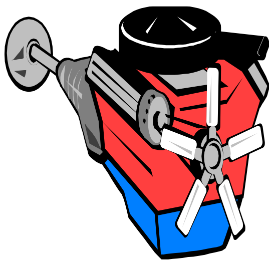
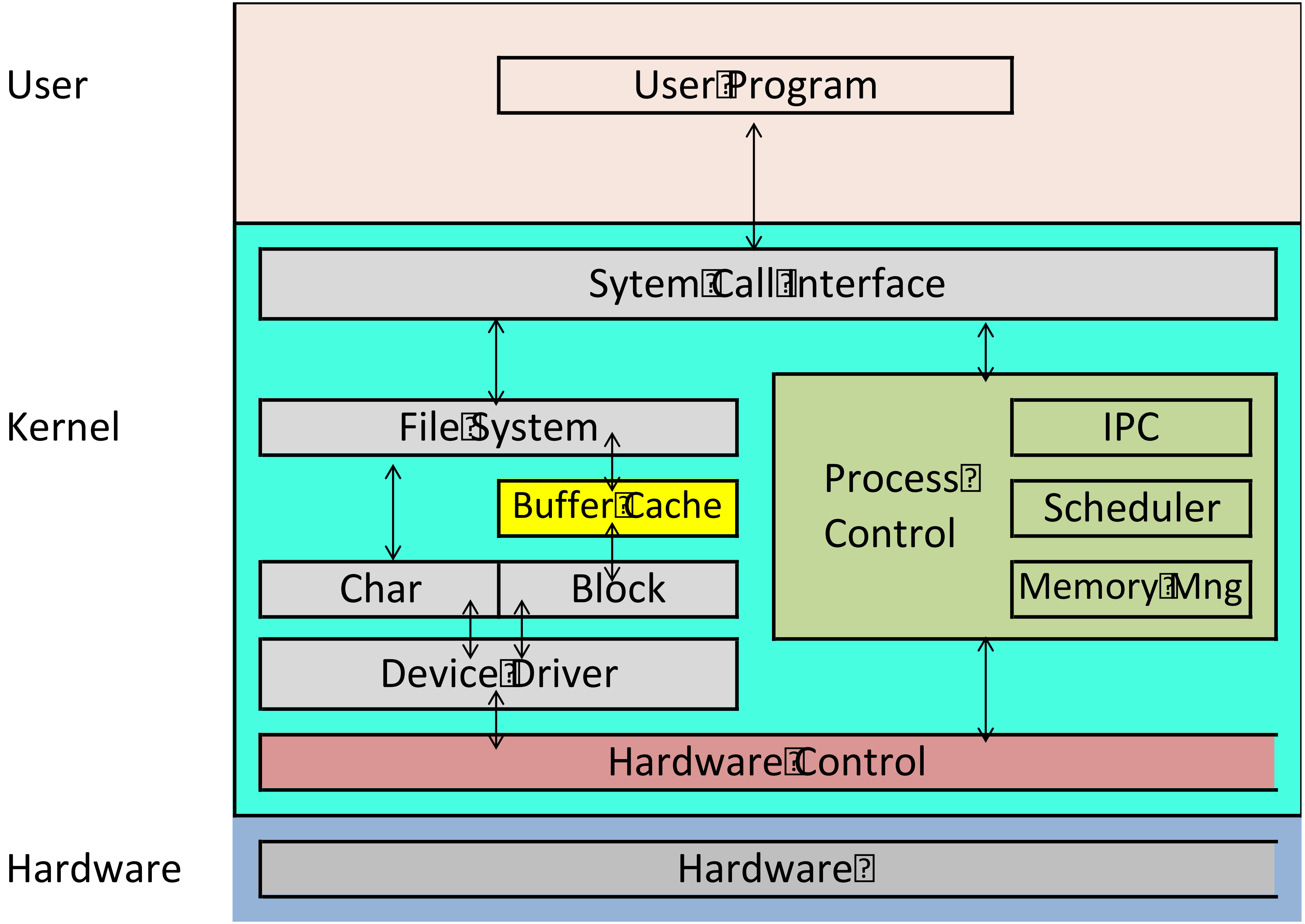


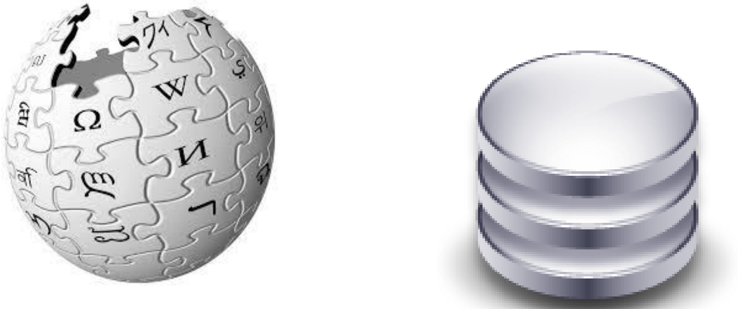
Figure 1-2. Operating systems turn ugly hardware into beautiful abstractions.

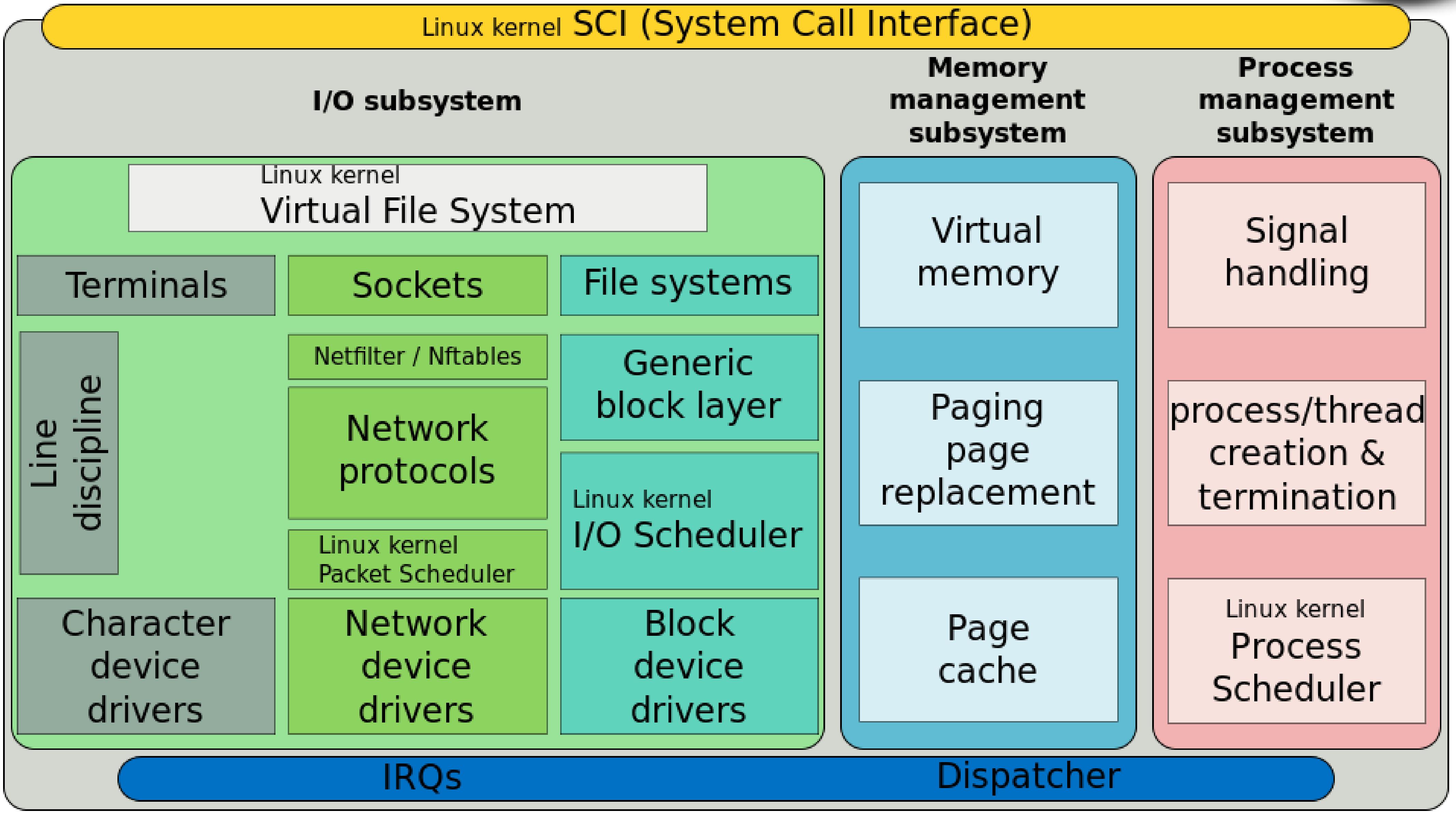
Traduction en cours

Operating System



Linux



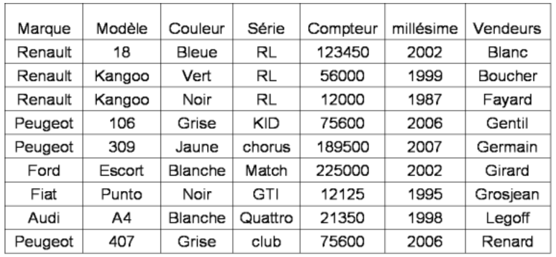


Linux kernel SCI (System Call Interface)
I/O subsystem
Memory management subsystem
Process management subsystem
Linux kernel Virtual File System
Terminals
Sockets
File systems
Line discipline
Netfilter / Nftables
Network protocols
Generic block layer
Linux kernel Packet Scheduler
Linux kernel I/O Scheduler
Character device drivers
Network device drivers
Block device drivers
Virtual memory
Paging page replacement
Page cache
Signal handling
process/thread creation & termination
Linux kernel Process Scheduler
IRQs
Dispatcher

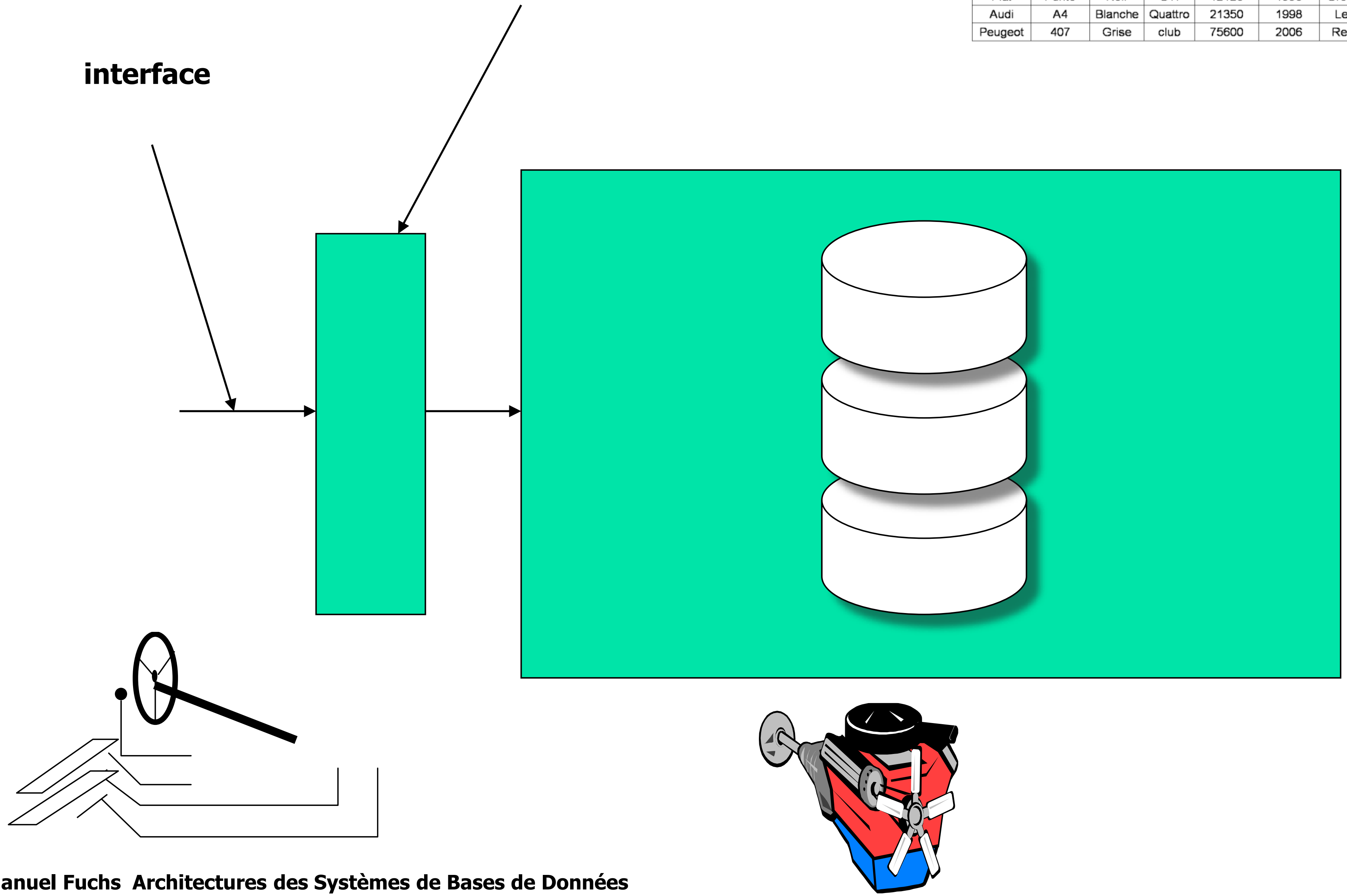
Encapsulation

Tables

Modèle = vue simplifiée →



Marque	Modèle	Couleur	Série	Compteur	millésime	Vendeurs
Renault	18	Bleue	RL	123450	2002	Blanc
Renault	Kangoo	Vert	RL	56000	1999	Boucher
Renault	Kangoo	Noir	RL	12000	1987	Fayard
Peugeot	106	Grise	KID	75600	2006	Gentil
Peugeot	309	Jaune	chorus	189500	2007	Germain
Ford	Escort	Blanche	Match	225000	2002	Girard
Fiat	Punto	Noir	GTI	12125	1995	Grosjean
Audi	A4	Blanche	Quattro	21350	1998	Legoff
Peugeot	407	Grise	club	75600	2006	Renard

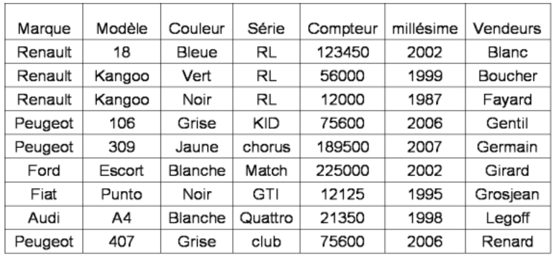


Fichiers

Traduction en cours

Encapsulation

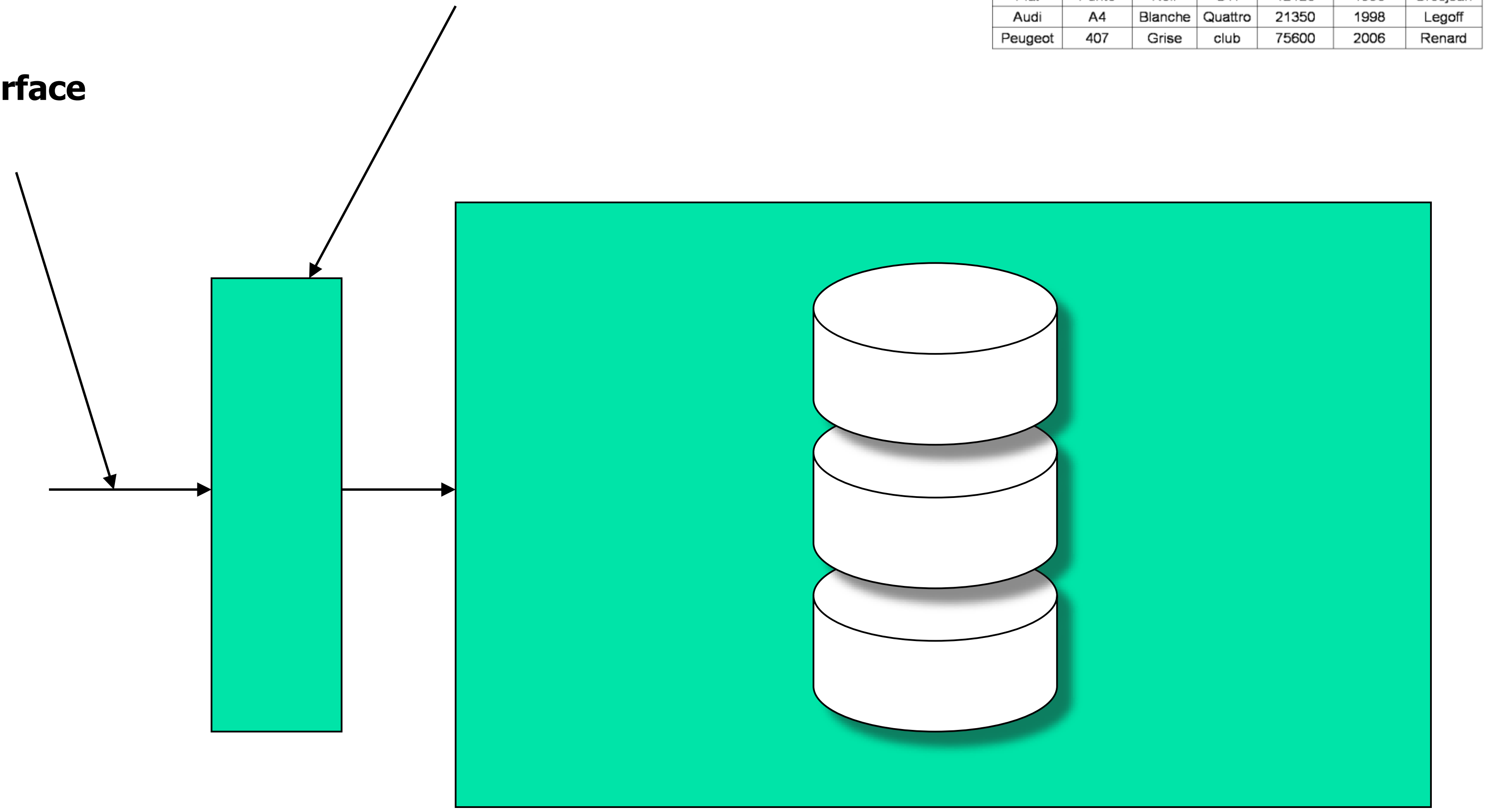
Tables



Marque	Modèle	Couleur	Série	Compteur	millésime	Vendeurs
Renault	18	Bleue	RL	123450	2002	Blanc
Renault	Kangoo	Vert	RL	56000	1999	Boucher
Renault	Kangoo	Noir	RL	12000	1987	Fayard
Peugeot	106	Grise	KID	75600	2006	Gentil
Peugeot	309	Jaune	chorus	189500	2007	Germain
Ford	Escort	Blanche	Match	225000	2002	Girard
Fiat	Punto	Noir	GTI	12125	1995	Grosjean
Audi	A4	Blanche	Quattro	21350	1998	Legoff
Peugeot	407	Grise	club	75600	2006	Renard

Modèle = vue simplifiée

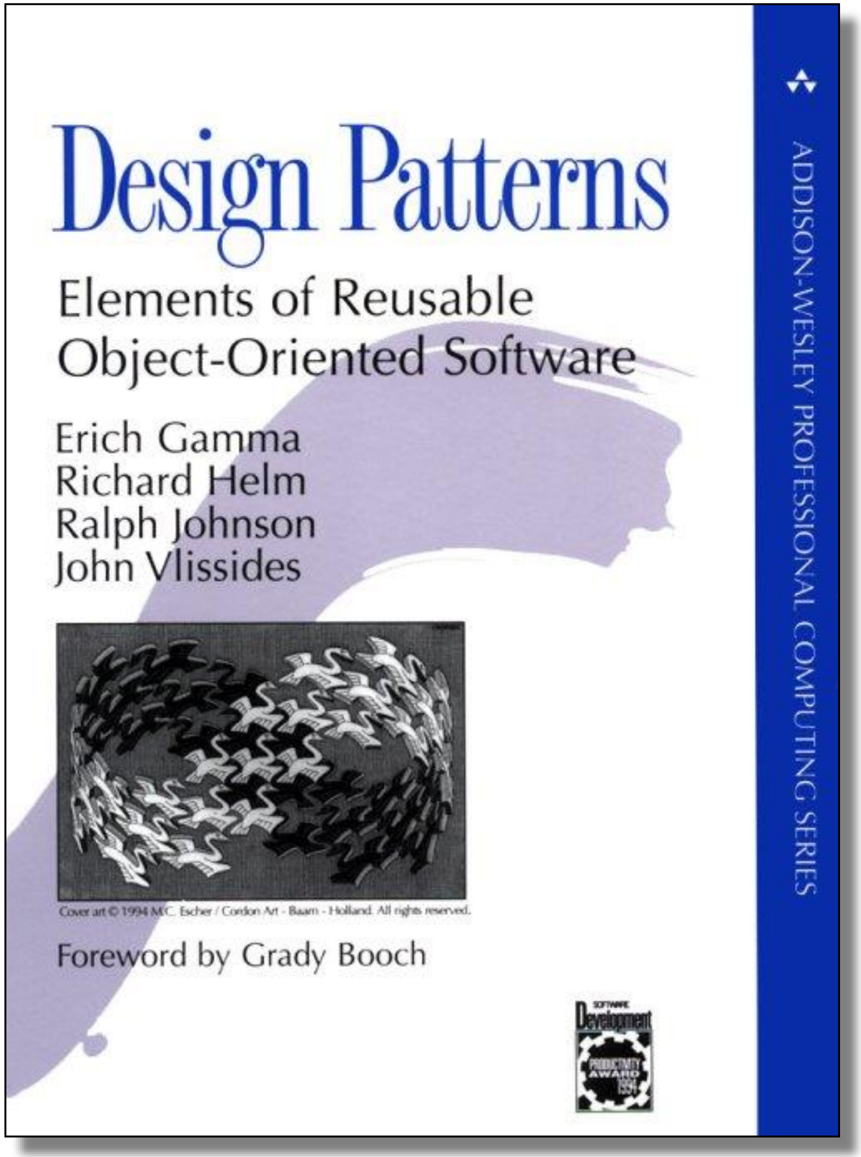
interface



Fichiers

Traduction en cours

The design pattern seminal book (1994)



Design Patterns

Elements of Reusable Object-Oriented Software

Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides

Cover art © 1994 M.C. Escher / Cordon Art - Baarn - Holland. All rights reserved.

Foreword by Grady Booch

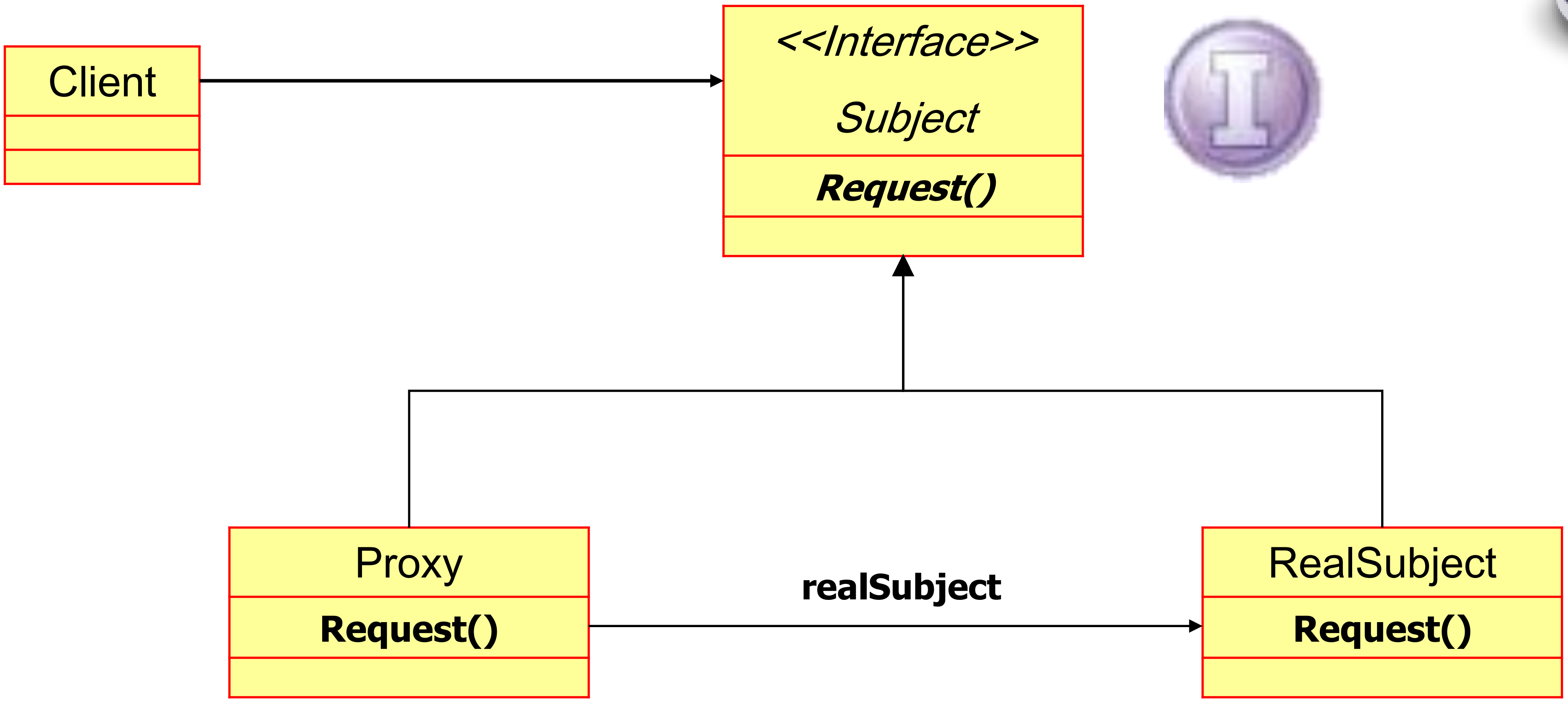
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

Ralph, Erich, Richard, and John at OOPSLA 1994

The Gang of Four

GOF

Proxy (UML)



Client

<<Interface>>

Subject

Request()

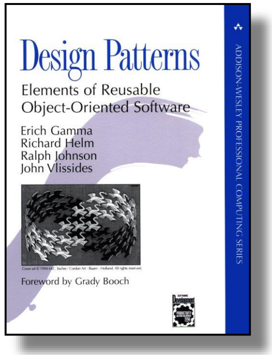
Proxy

Request()

realSubject

RealSubject

Request()



Design Patterns

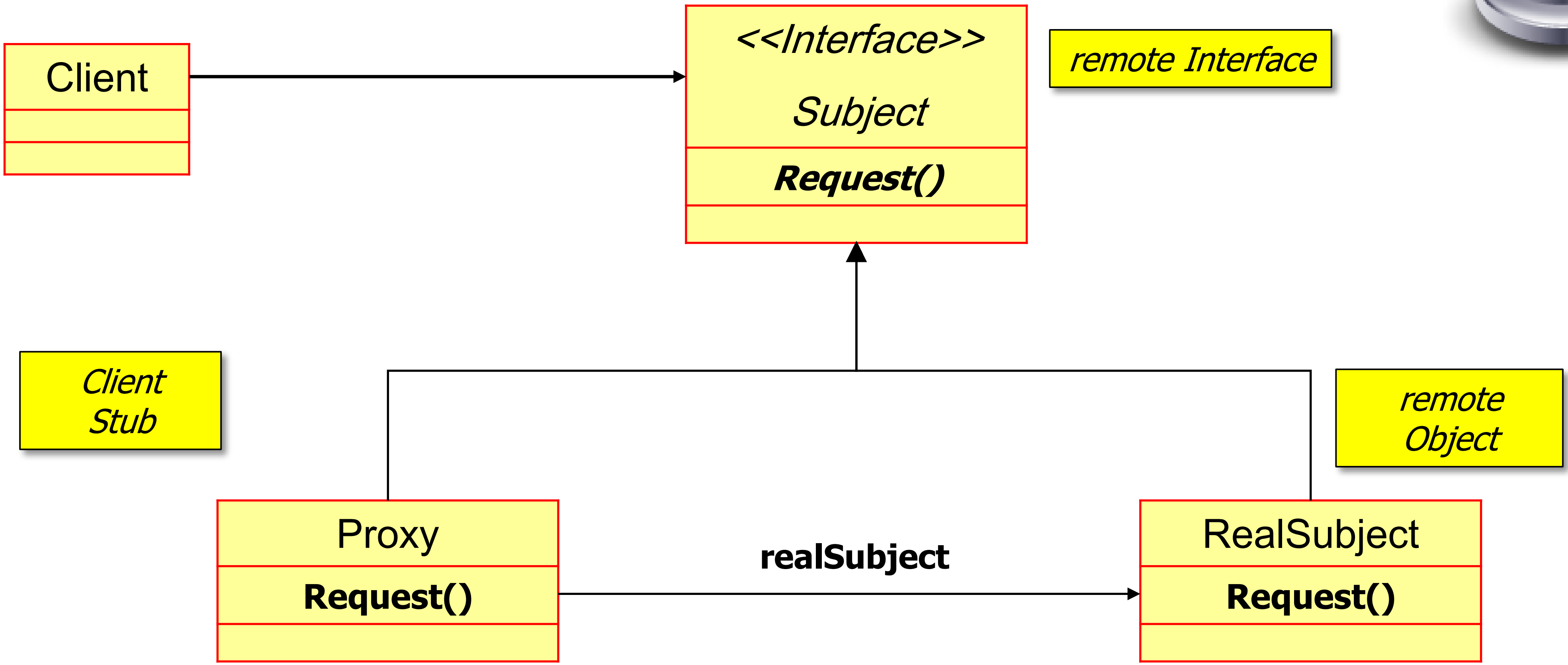
Elements of Reusable Object-Oriented Software

Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides

Foreword by Grady Booch

ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

Proxy (UML)



Client

<<Interface>>

Subject

Request()

remote Interface

Client Stub

remote Object

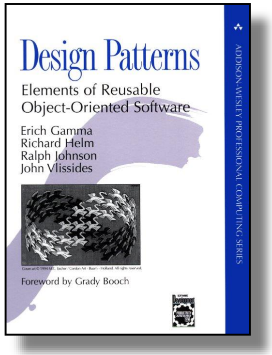
Proxy

Request()

realSubject

RealSubject

Request()



Design Patterns

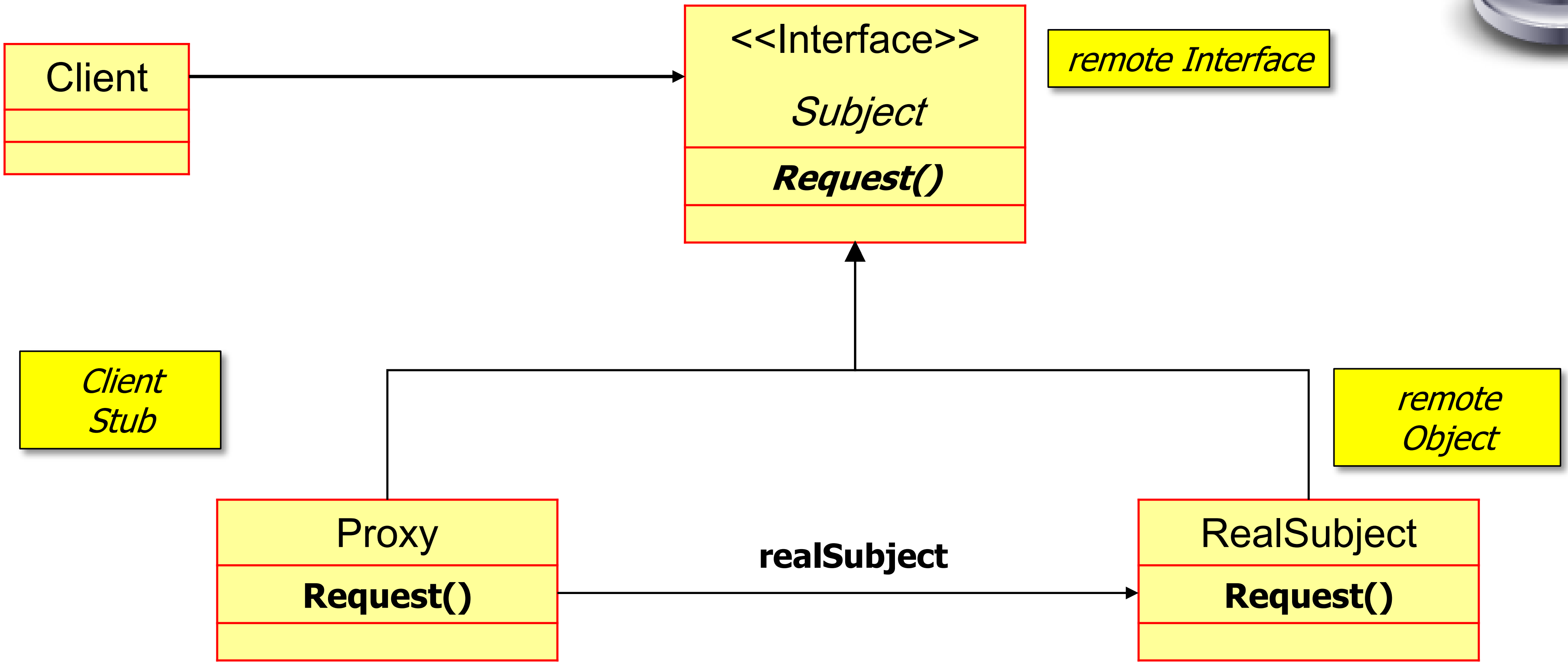
Elements of Reusable Object-Oriented Software

Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides

Foreword by Grady Booch

ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

Proxy (UML)



Client

<<Interface>>

Subject

Request()

remote Interface

Client Stub

remote Object

Proxy

Request()

realSubject

RealSubject

Request()

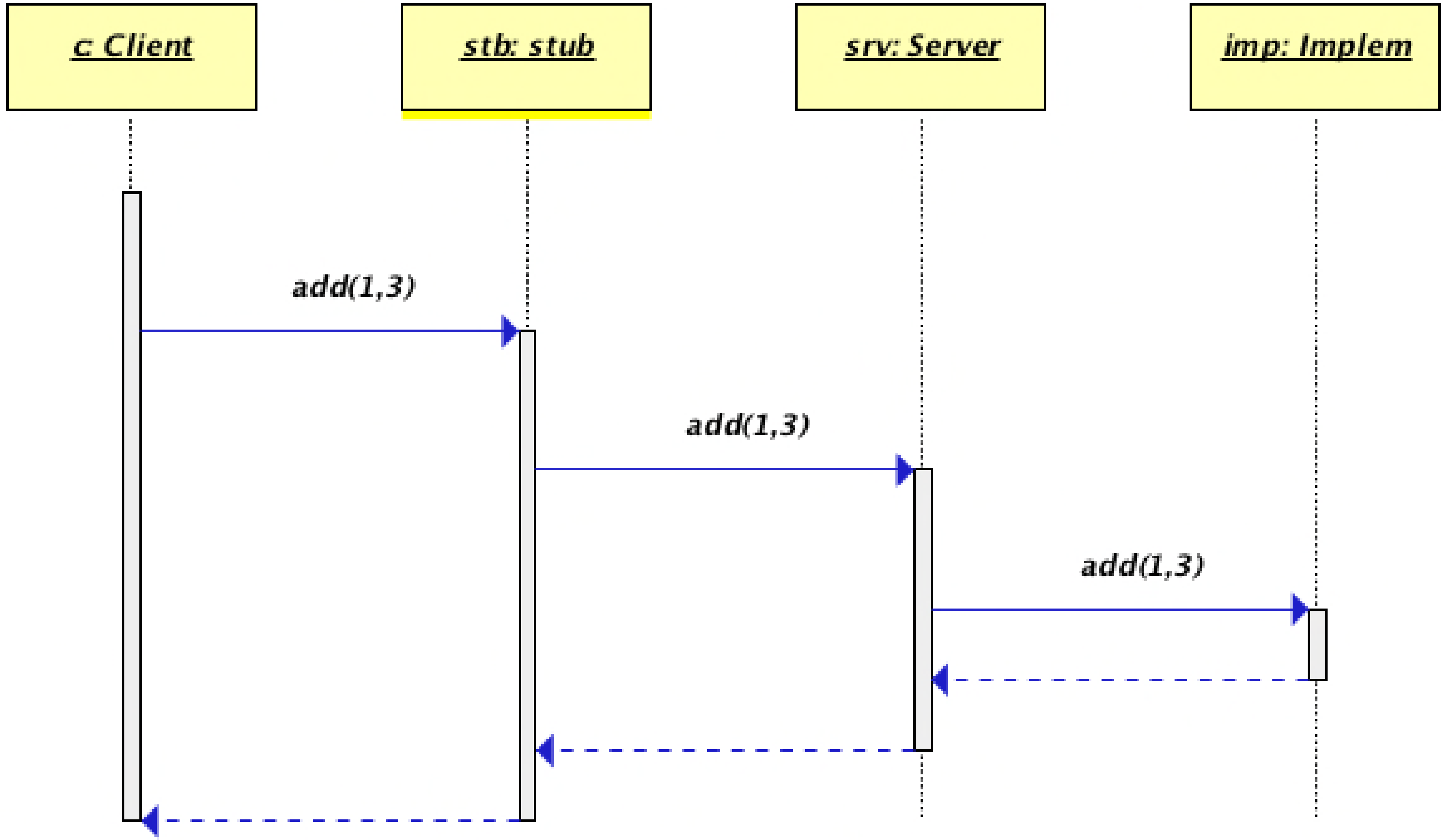
Client side Stub

Client JVM

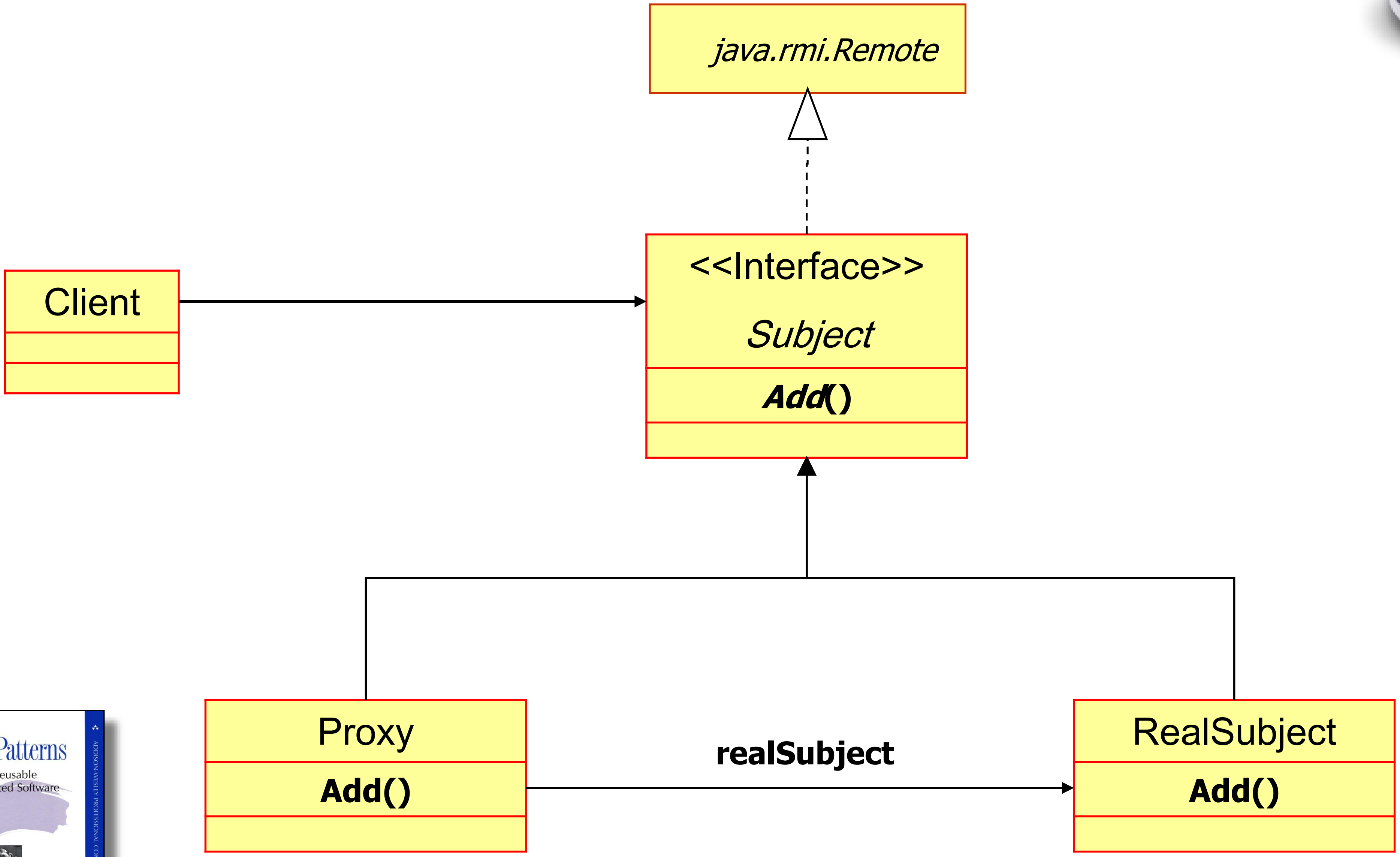
Server side Skeleton

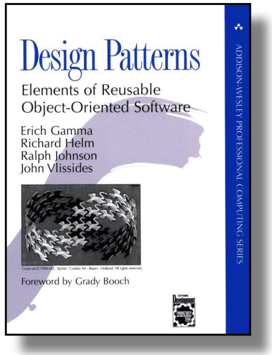
Server JVM

Proxy (UML) : Add server



Proxy (UML) : Add server





Design Patterns

Elements of Reusable Object-Oriented Software

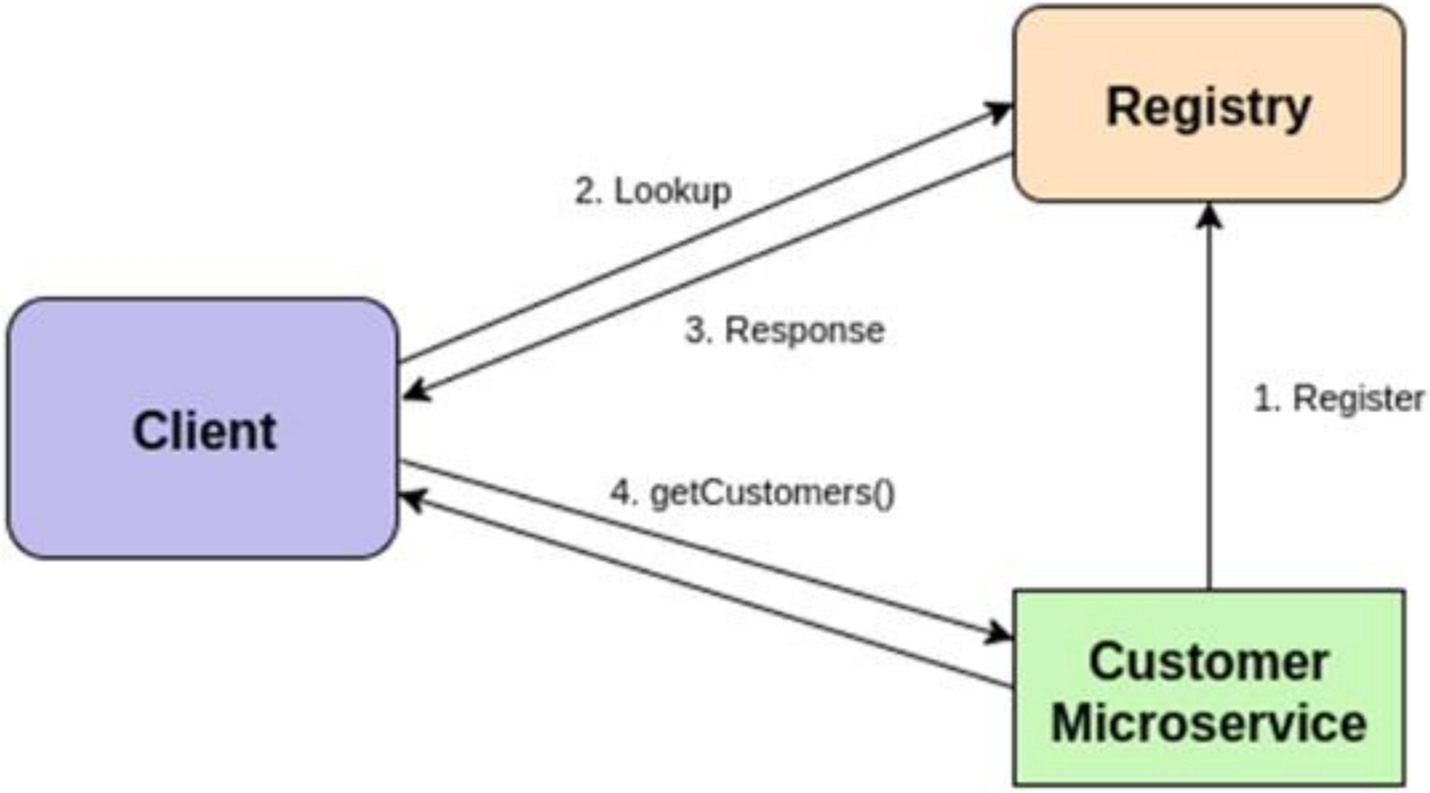
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides

Foreword by Grady Booch

ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

Services Interface Registry

Microservices



Registry

2. Lookup

3. Response

1. Register

Client

4. getCustomers()

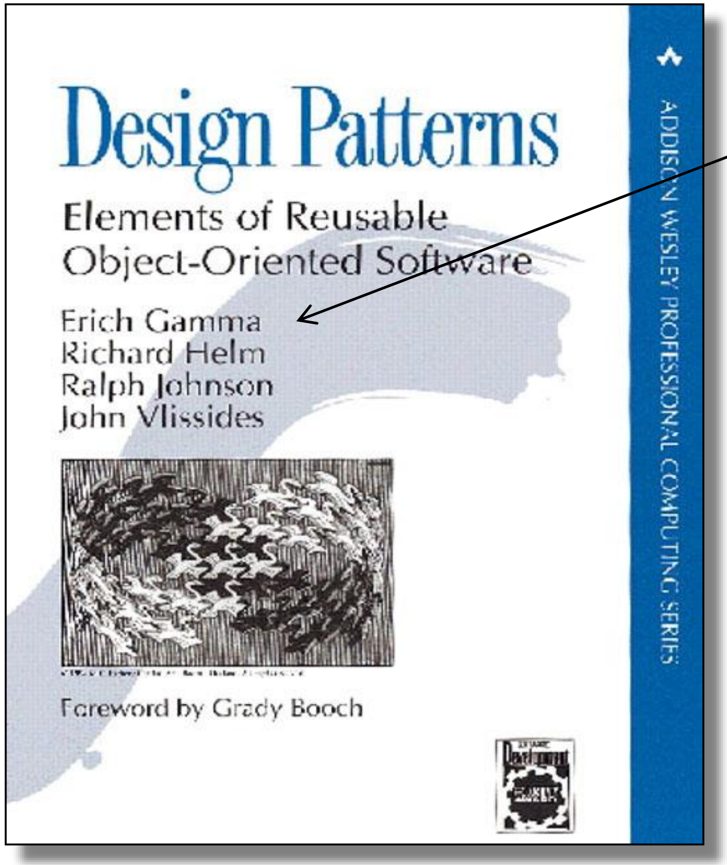
Customer Microservice

https://jaxenter.com/eclipse-microservices-service-discovery-138577.html

GOF

“program to an interface, not an implementation”

http://en.wikipedia.org/wiki/Design_Patterns



Design Patterns

Elements of Reusable Object-Oriented Software

Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides

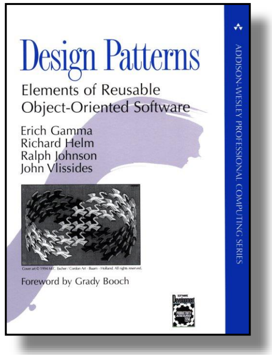
Foreword by Grady Booch

ADDISON WESLEY PROFESSIONAL COMPUTING SERIES

Moodel

GOF rules

- "Program to an 'interface', not an 'implementation'."
- Composition over inheritance:
  - "Favor 'object composition' over 'class inheritance'."
- Advantages of interfaces over implementation:
  - Clients remain unaware of the specific types of objects they use, as long as the object adheres to the interface
  - Clients remain unaware of the classes that implement these objects;



Design Patterns

Elements of Reusable Object-Oriented Software

Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides

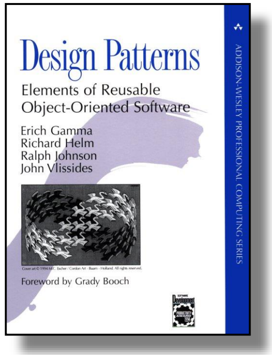
Foreword by Grady Booch

ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

Dynamic binding and polymorphism,

- Use of an interface leads to dynamic binding and polymorphism, which are central features of object-oriented programming

- Object Interaction Model



Design Patterns

Elements of Reusable Object-Oriented Software

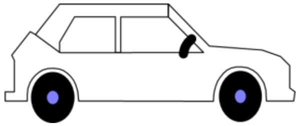
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides

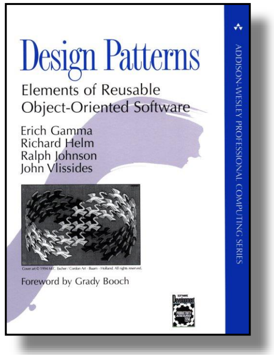
Foreword by Grady Booch

ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

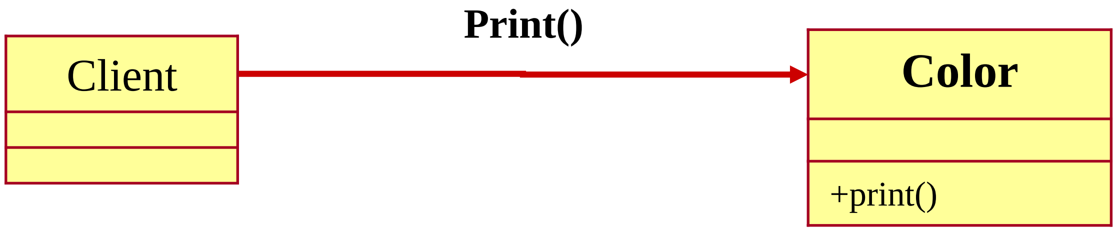
The object solution

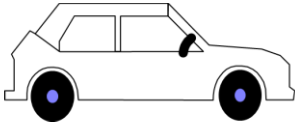
- Polymorphism




Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Foreword by Grady Booch
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

Polymorphism

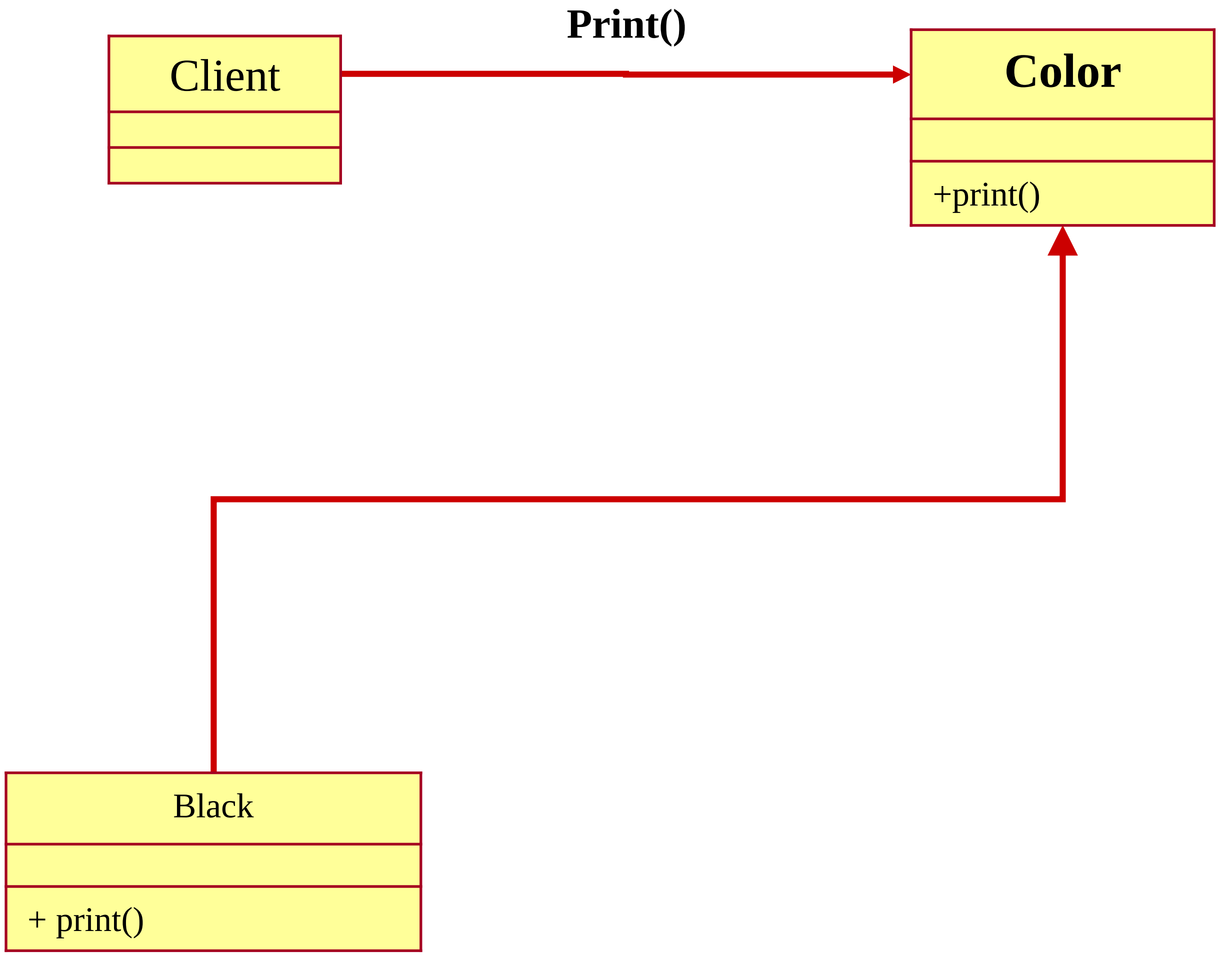




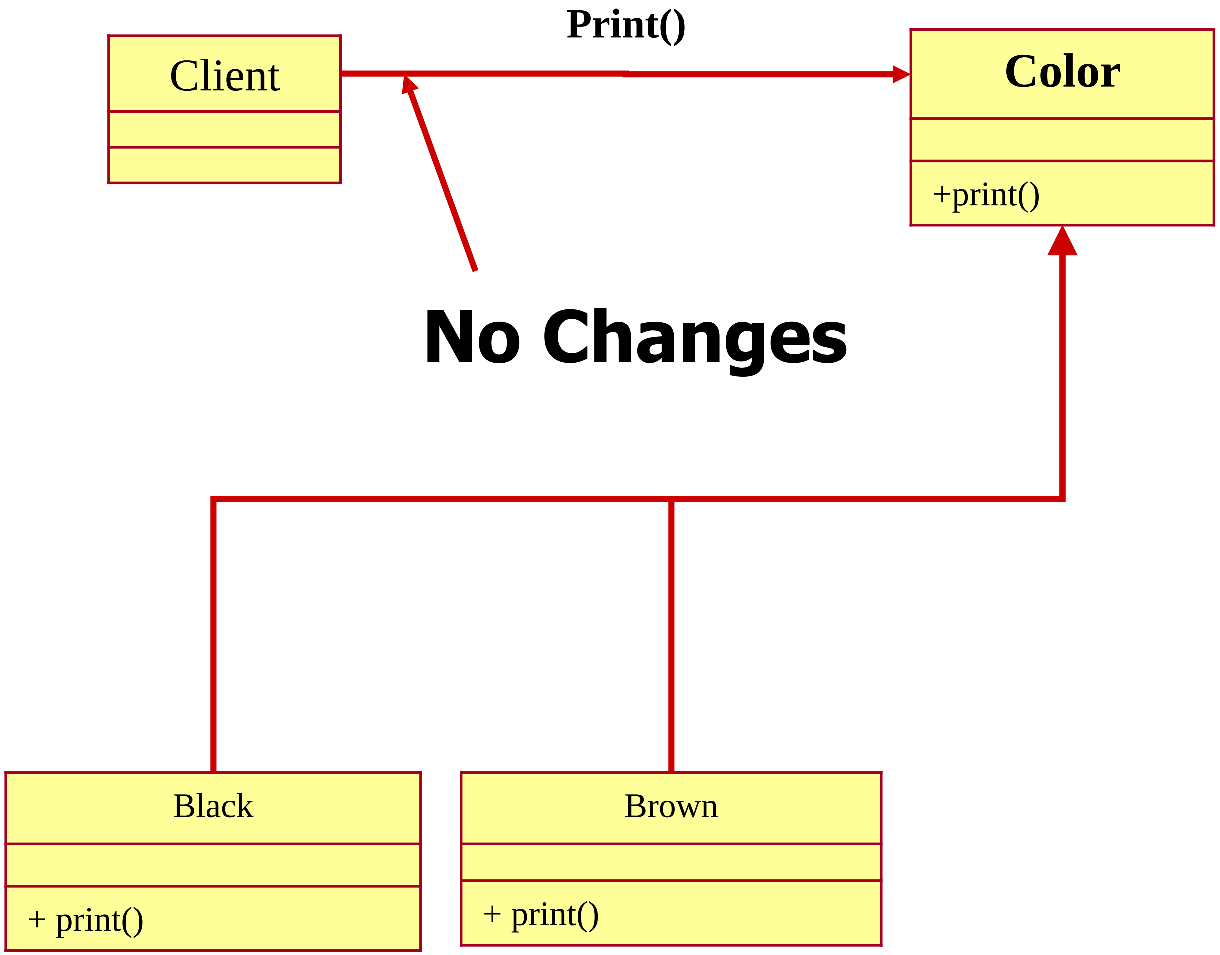
Client Object sends a message to Server Object

Message = Method

Polymorphism



Polymorphism



Print()

Client

Color

+print()

No Changes

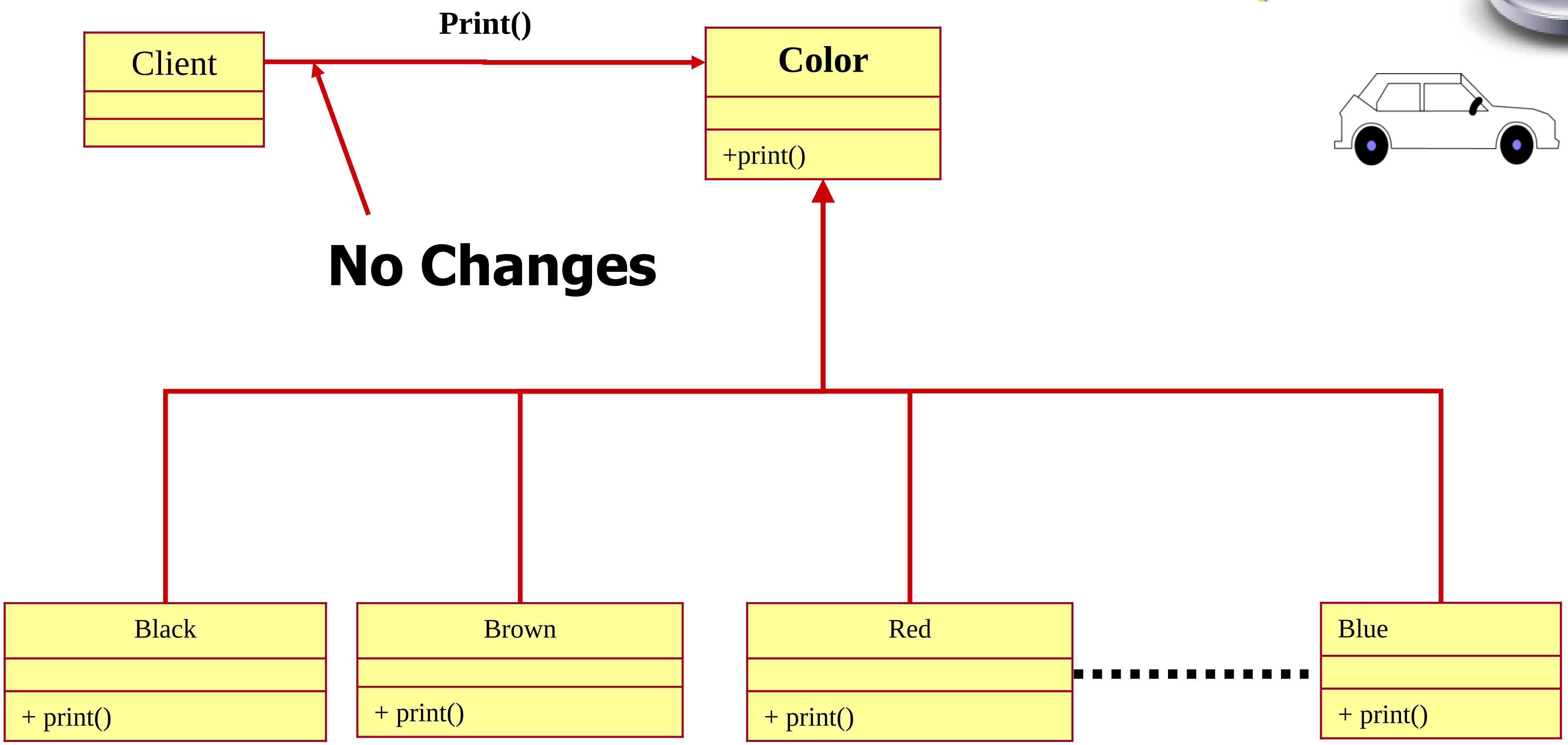
Black

+ print()

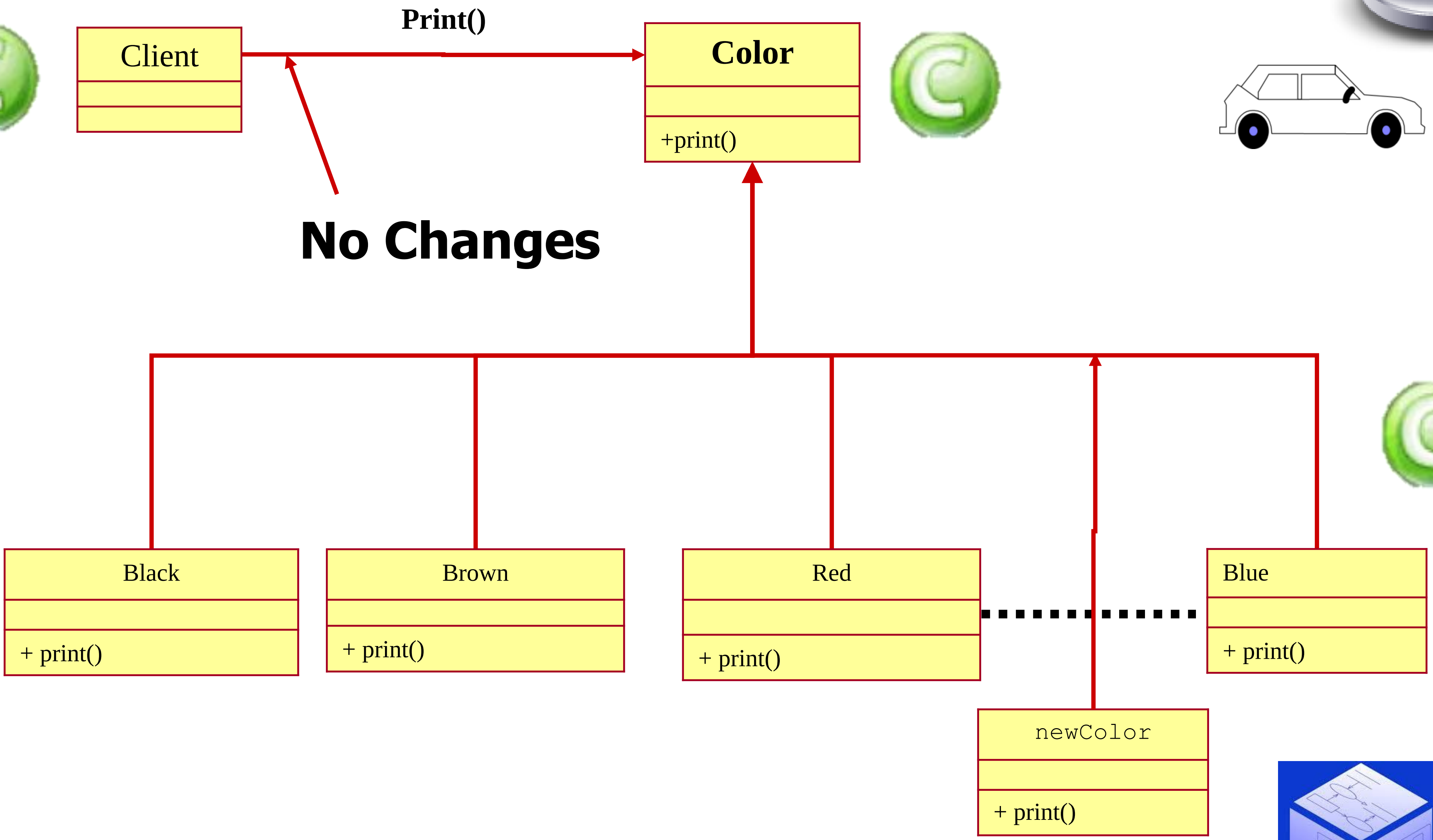
Brown

+ print()

Polymorphism



Polymorphism



Print()

Client

Color

+print()

No Changes

Black

+ print()

Brown

+ print()

Red

+ print()

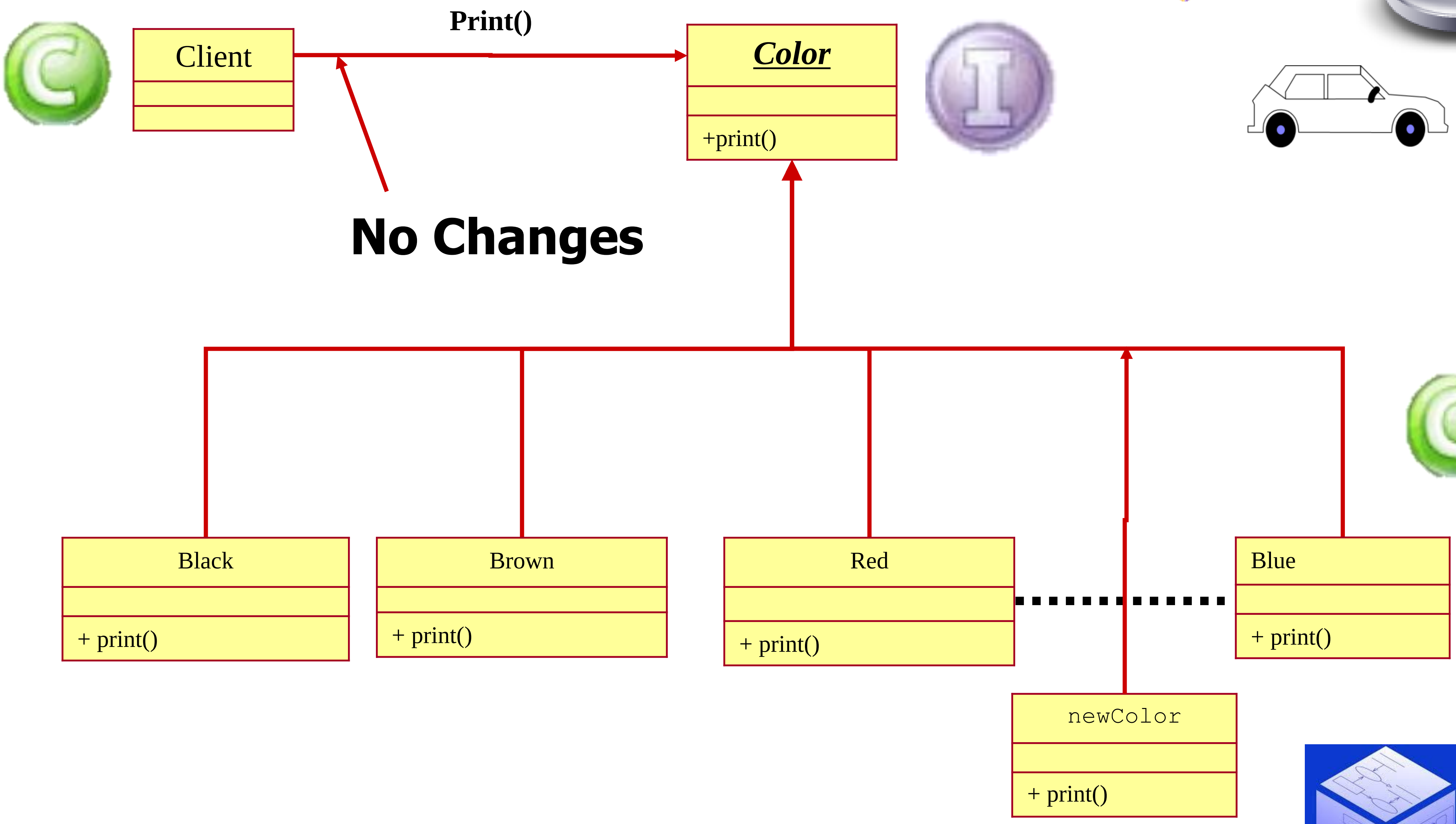
Blue

+ print()

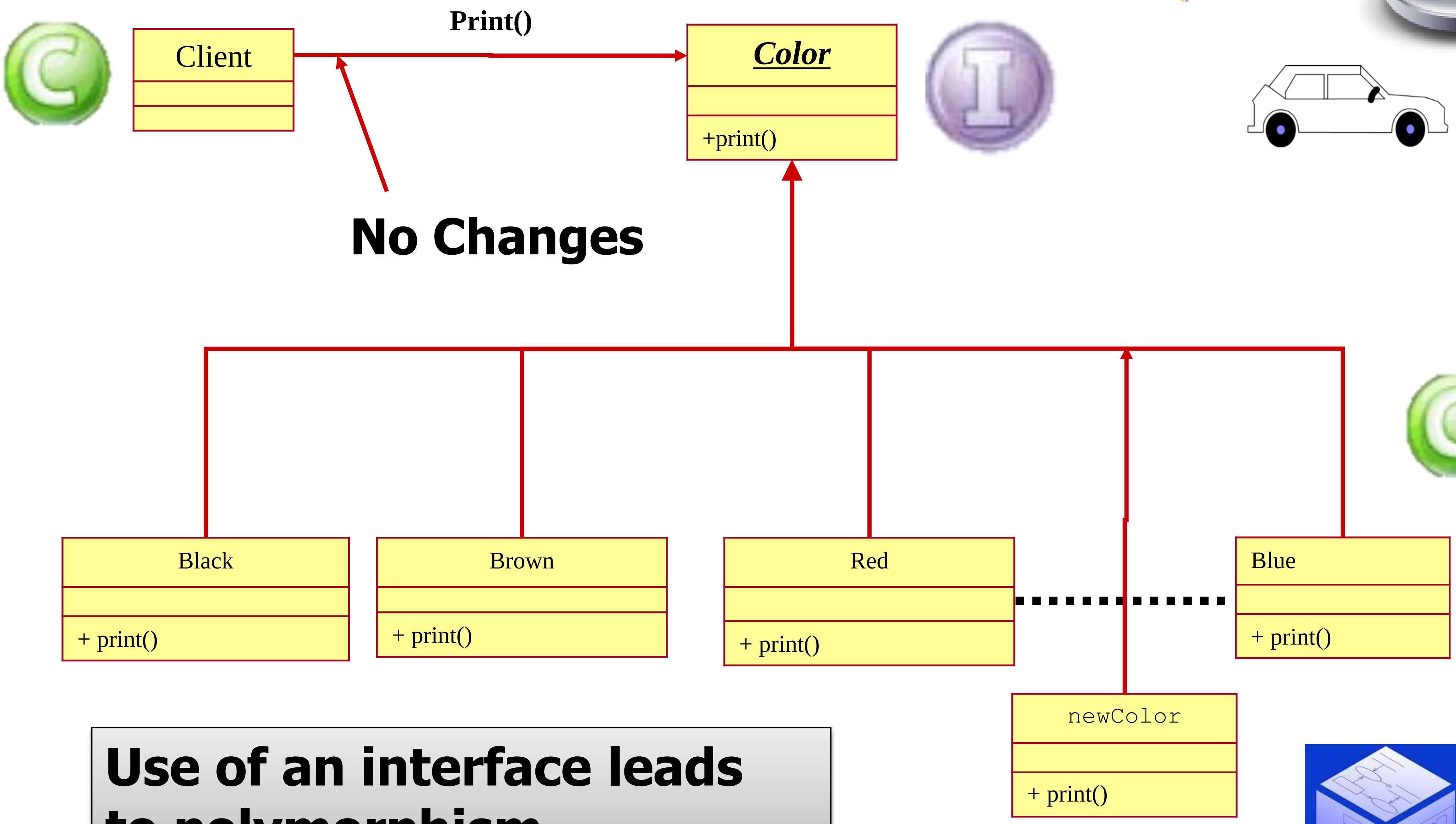
newColor

+ print()

Polymorphism

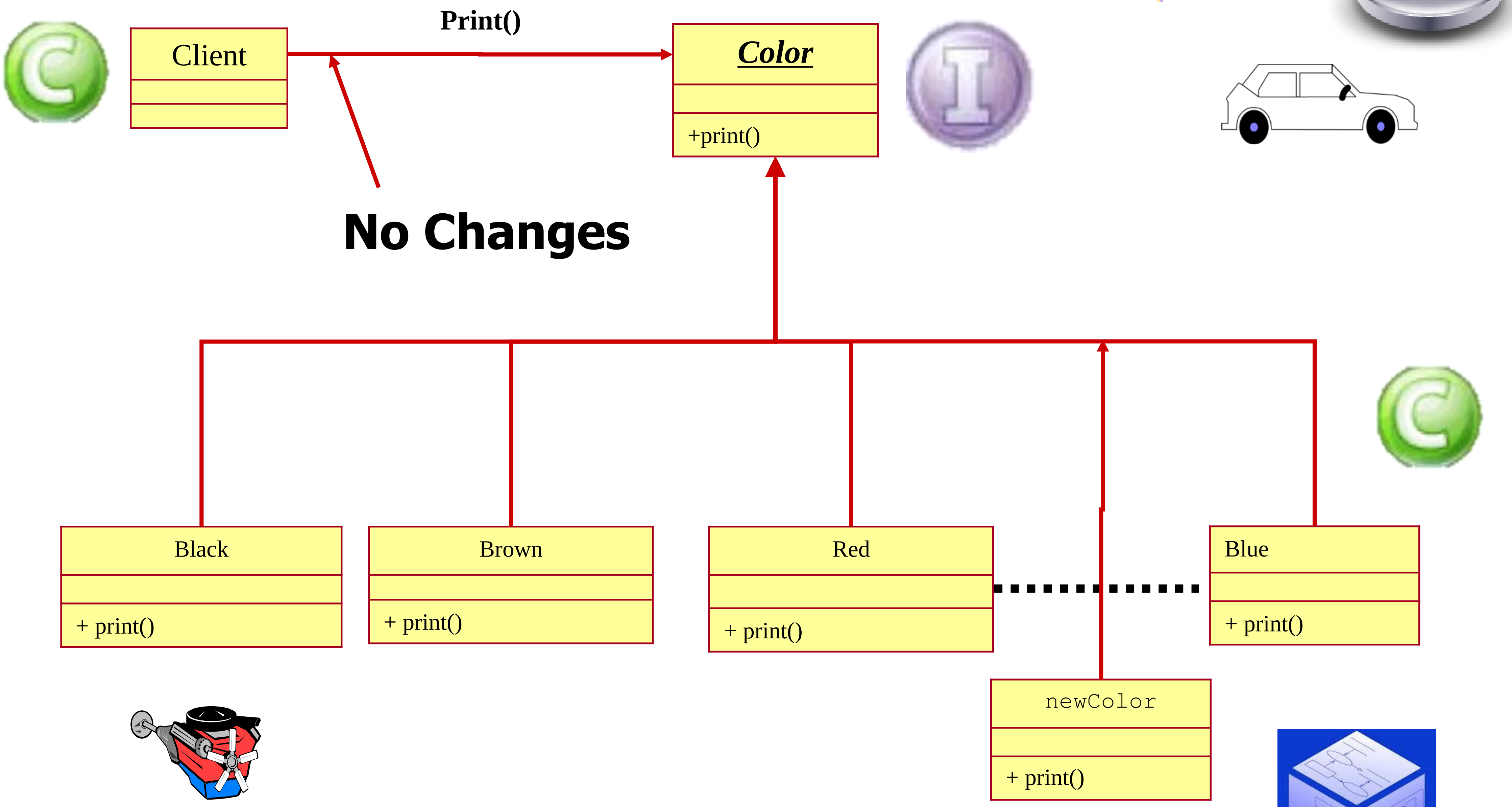


Polymorphism



Use of an interface leads to polymorphism

Polymorphism



Client

Print()

Color

+print()

No Changes

Black

+ print()

Brown

+ print()

Red

+ print()

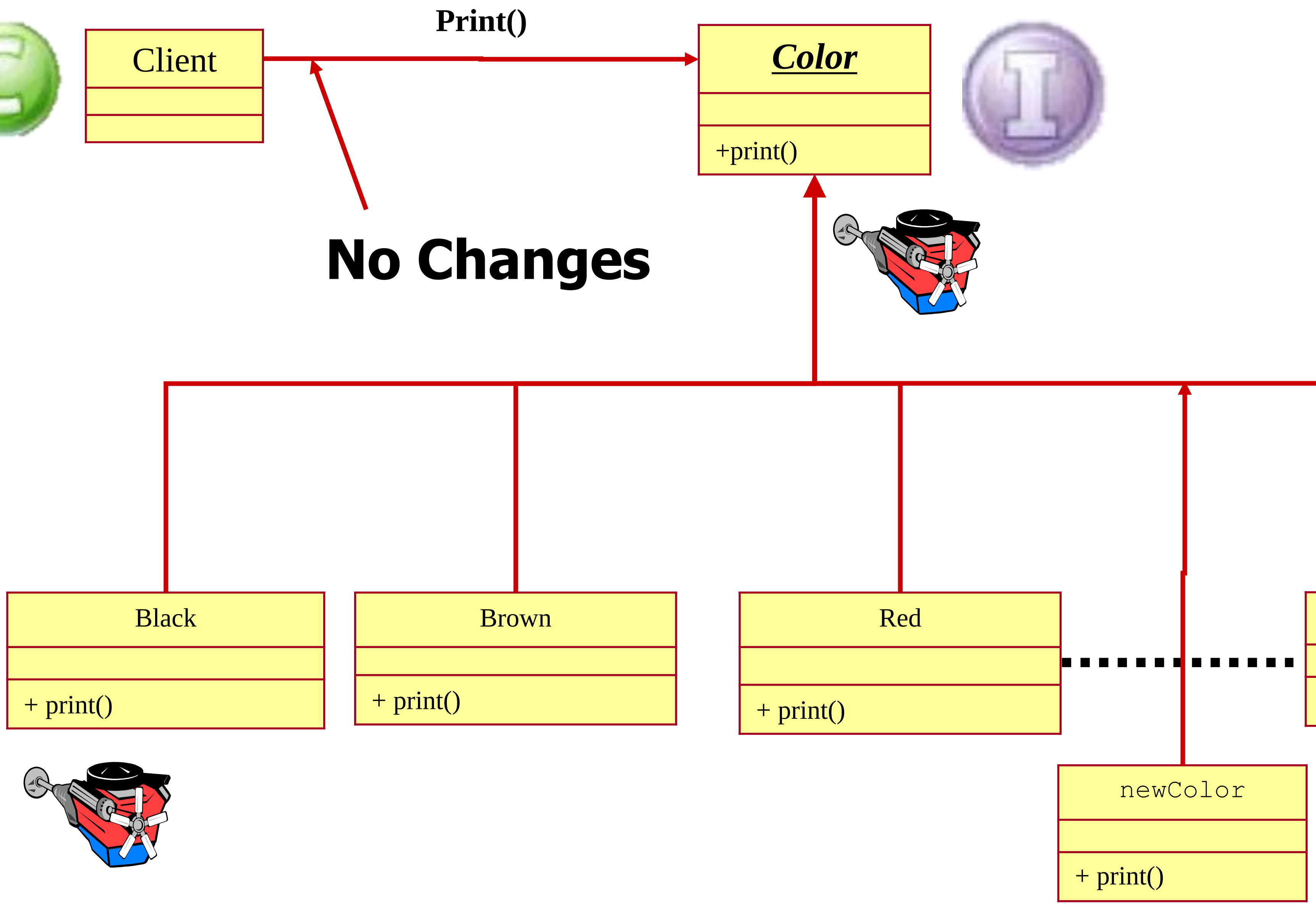
Blue

+ print()

newColor

+ print()

Polymorphism



Print()

Client

Color

+print()

No Changes

Black

+ print()

Brown

+ print()

Red

+ print()

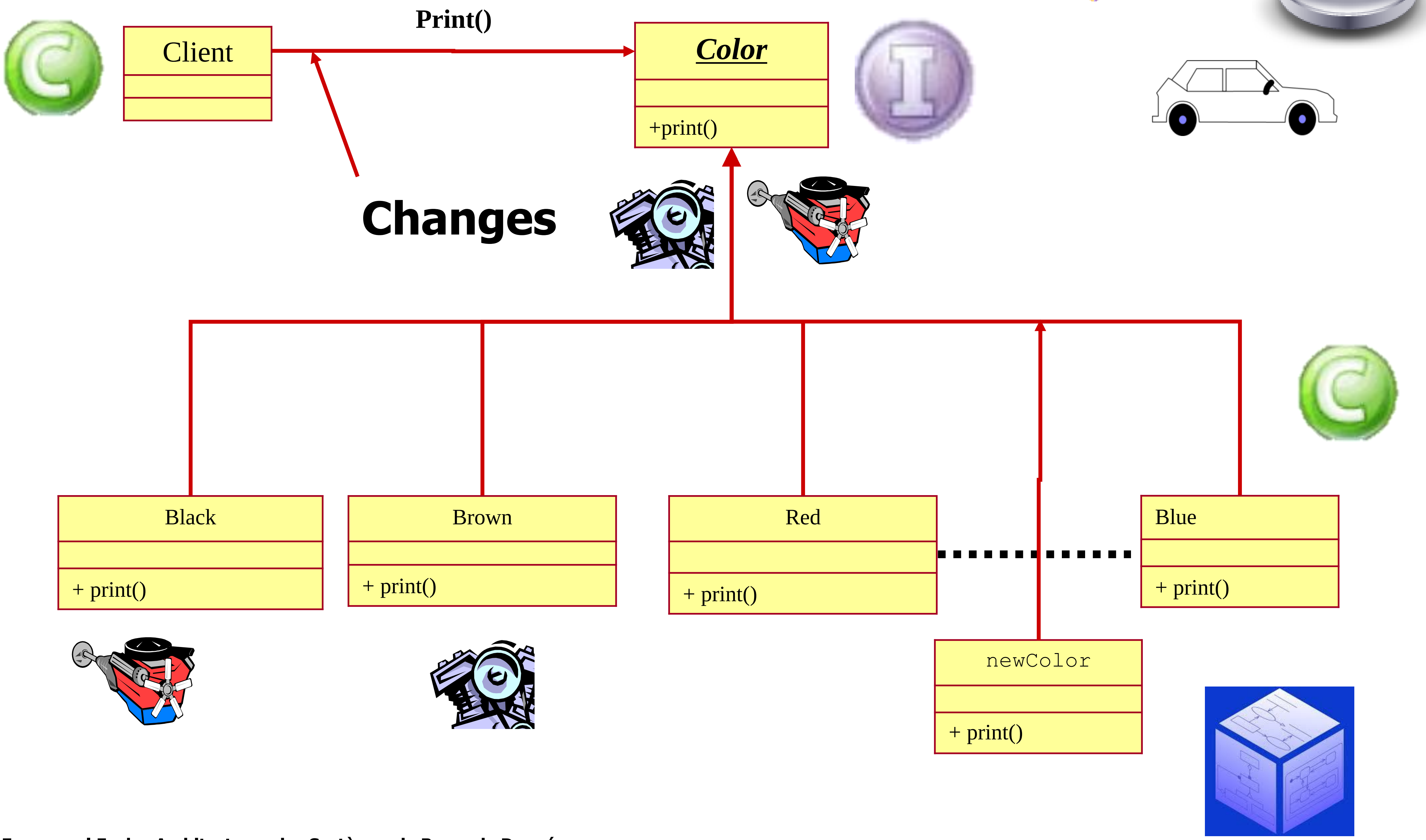
Blue

+ print()

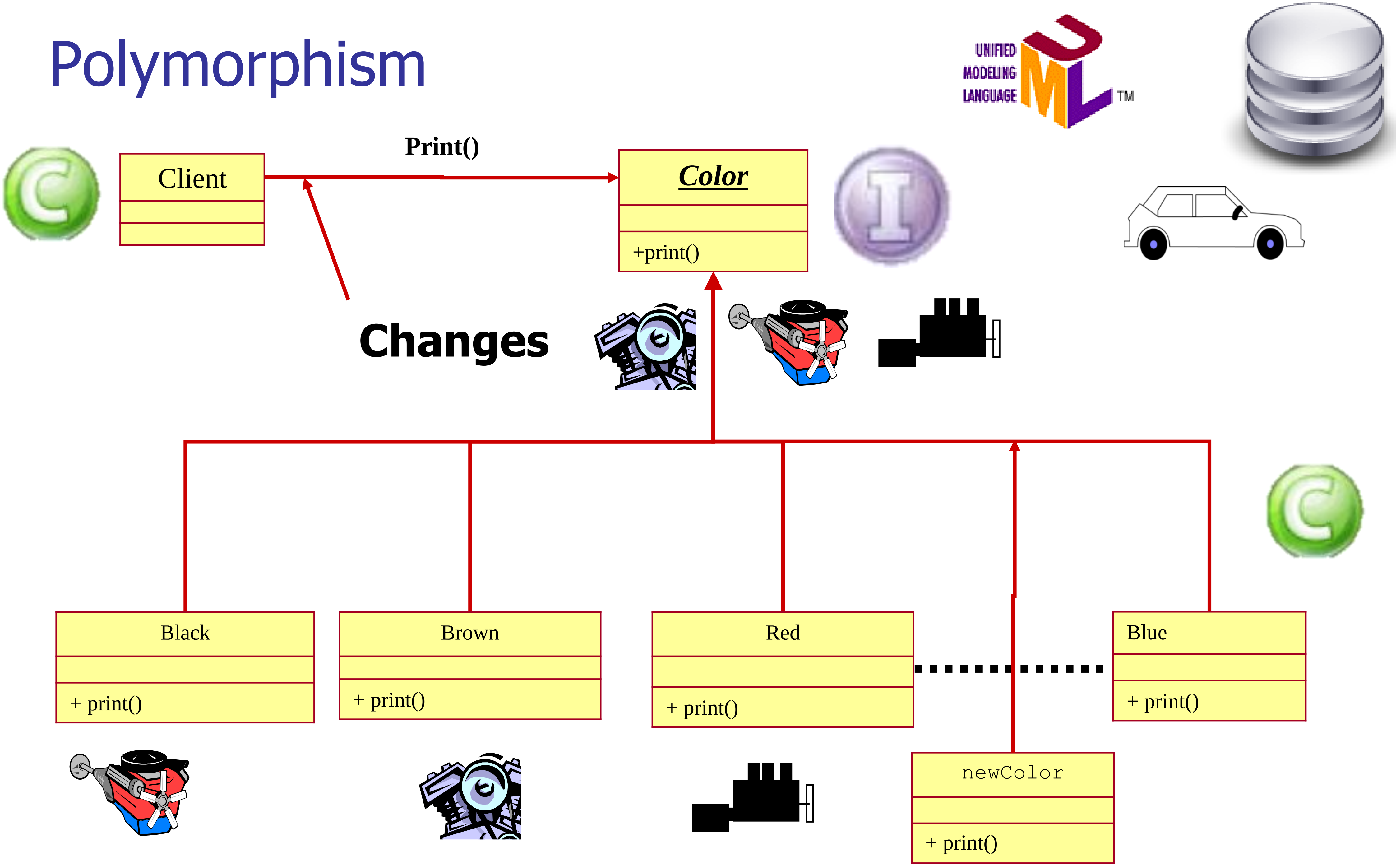
newColor

+ print()

Polymorphism

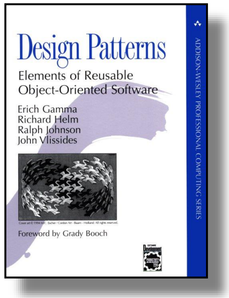


Polymorphism



Inheritance versus Composition

- Inheritance is white-box reuse :
  - White-box referring to visibility,
  - The internals of parent classes are often visible to subclasses.
- Object composition :
  - Well-defined interfaces are used dynamically at runtime by objects obtaining references to other objects.
- Black-box reuse :
  - No internal details of composed objects need be visible in the code using them.



Design Patterns

Elements of Reusable Object-Oriented Software

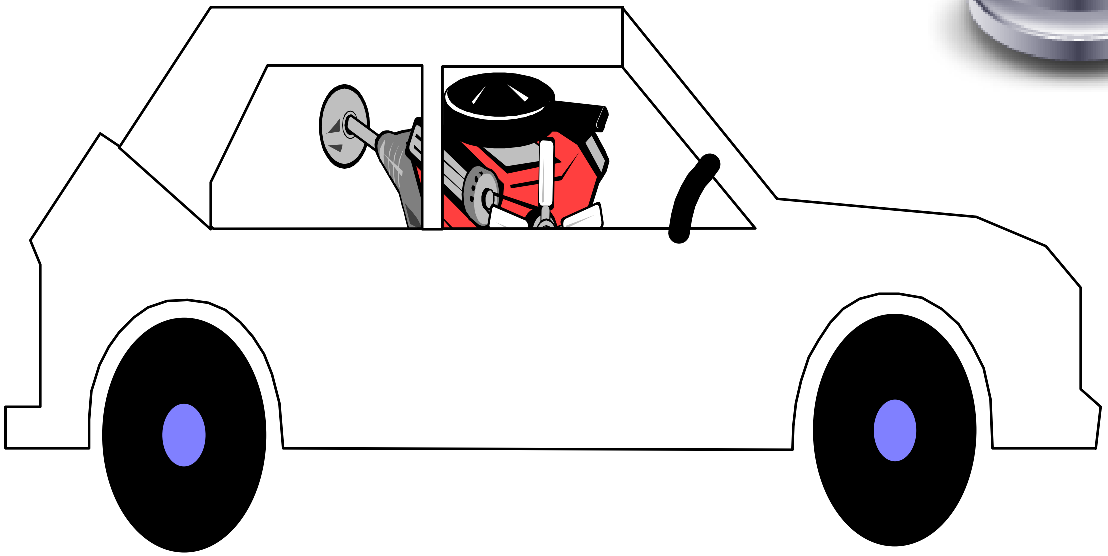
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides

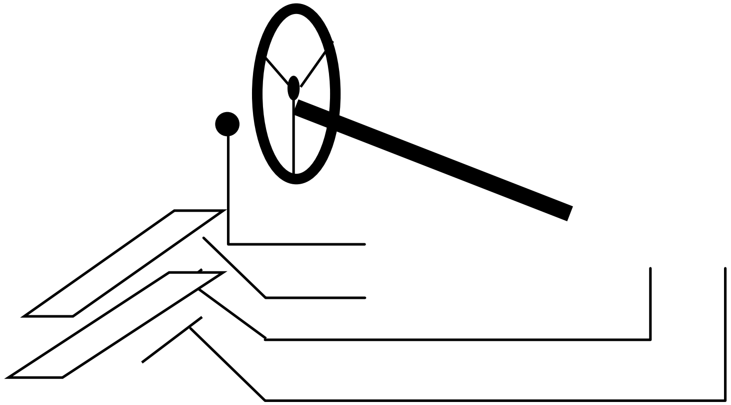
Foreword by Grady Booch

ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

Interface Versus Implementation

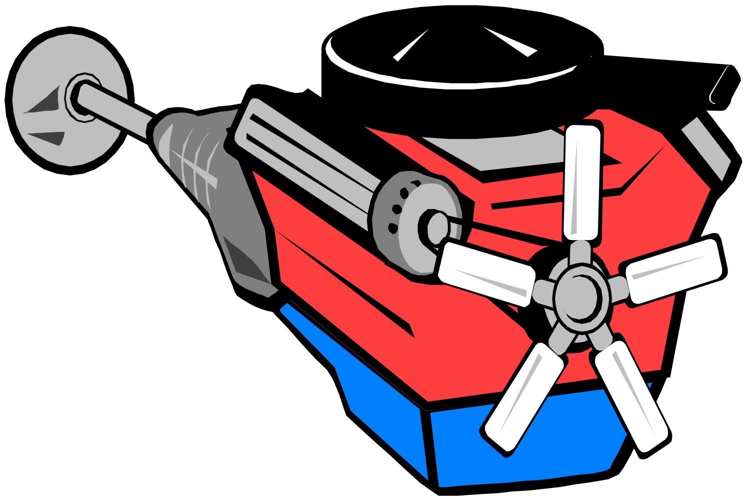
No Implementation details
In the Interface





Interface

(specification)



Implementation

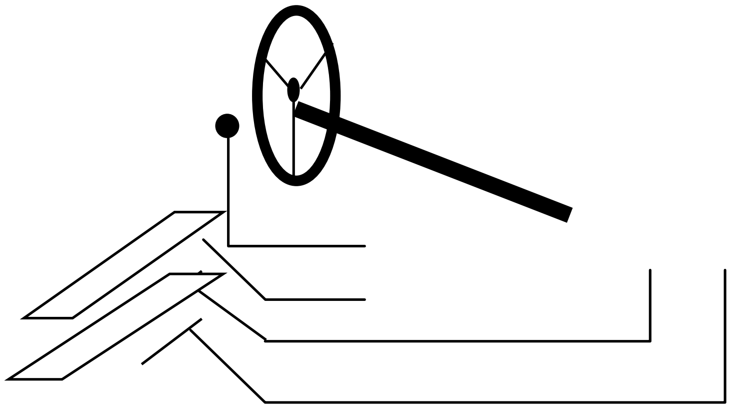
(body)

```
Struct {
        Int field1
        double field2
        abc field3
        xxx field4
}
```

Interface Versus Implementation

No Implementation details
In the Interface

```
public interface HashProbing {

            boolean put(int key, char value);

            char get(int key);

            boolean remove(int key);
}
```

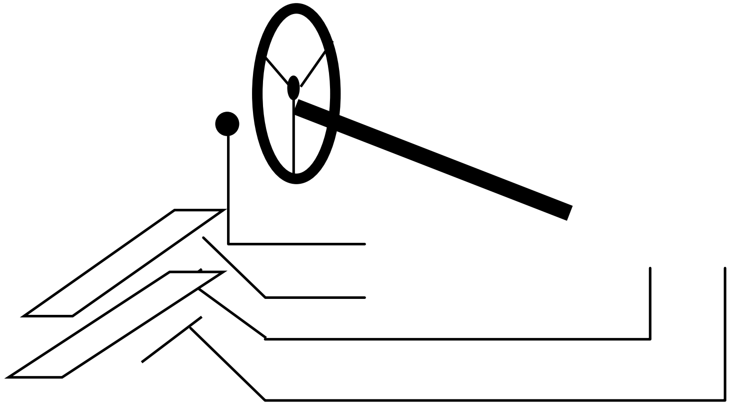


Interface

(specification)

Interface Versus Implementation

No Implementation details
In the Interface



Interface

(specification)

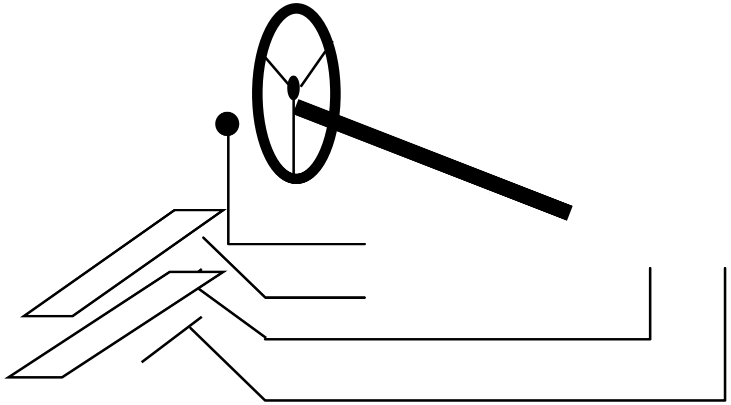
```
Public interface HashProbing {

            public boolean put(int key, int value);

            public int get(int key);

            public void remove(int key);

            public int[] keys();

            public int[] values();

            public double load();
}
```

X

Interface Versus Implementation

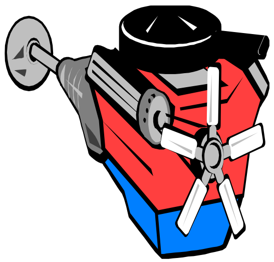
No Implementation details In the Interface

```
Public interface HashProbing {

        public boolean put(int key, int value);

        public int get(int key);

        public void remove(int key);

        public int[] keys();

        public int[] values();

        public double load();
}
```



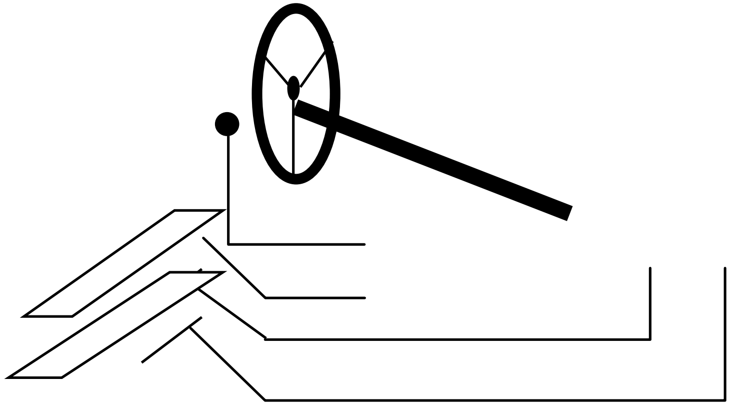
Interface

(specification)



Interface Versus Implementation

No Implementation details
In the Interface



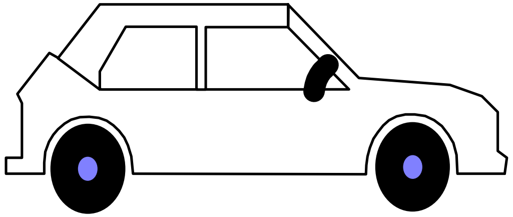
Interface

(specification)

```
public interface HashProbing {

        public int hash(int key);

        public void put(int key, int value);

        public int get(int key);

        public void remove(int key);

        public String toString();

        public boolean isSlotEmpty(int key, int value);
}
```

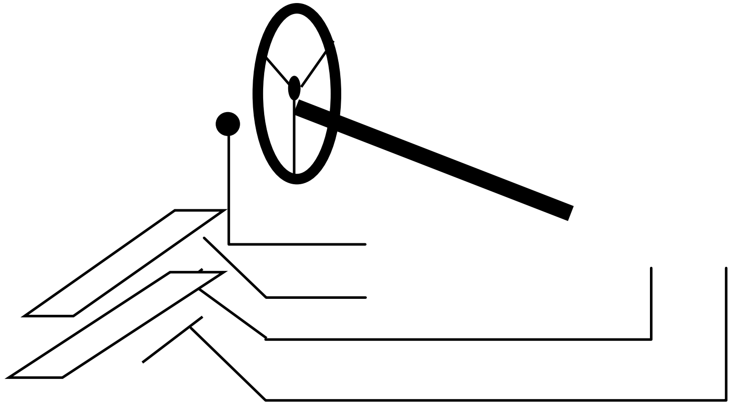
✗

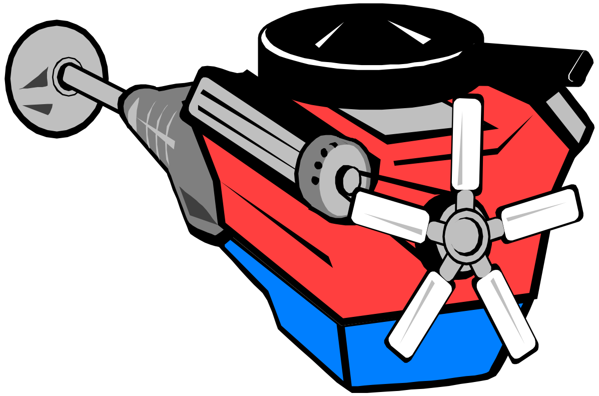
Interface Versus Implementation



Business Interface

Technical Interface





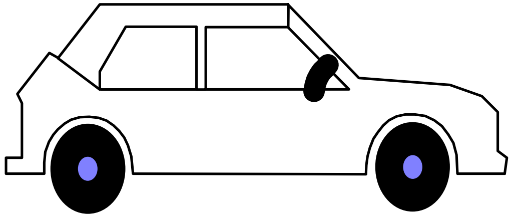
Interface

(specification)

Implementation

(body)

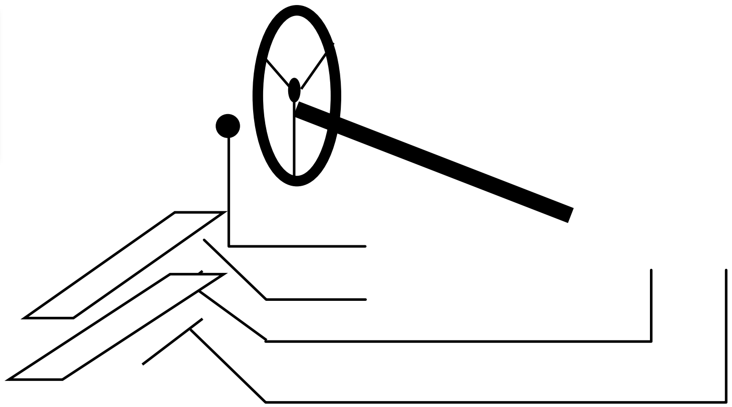
Interface Versus Implementation



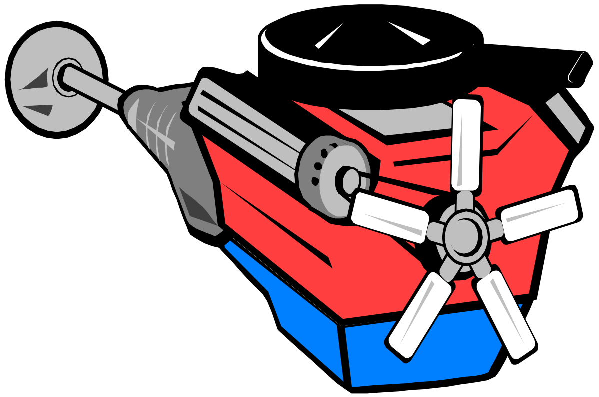
```
Public interface HashProbing {

    public boolean put(int key, int value);

    public int get(int key);

    public void remove(int key);
}
```

```
Public interface HashProbingTest {

    public int[] keys();

    public int[] values();

    public double load();

    public int hash(int key);
}
```

Business Interface



Technical Interface



Interface

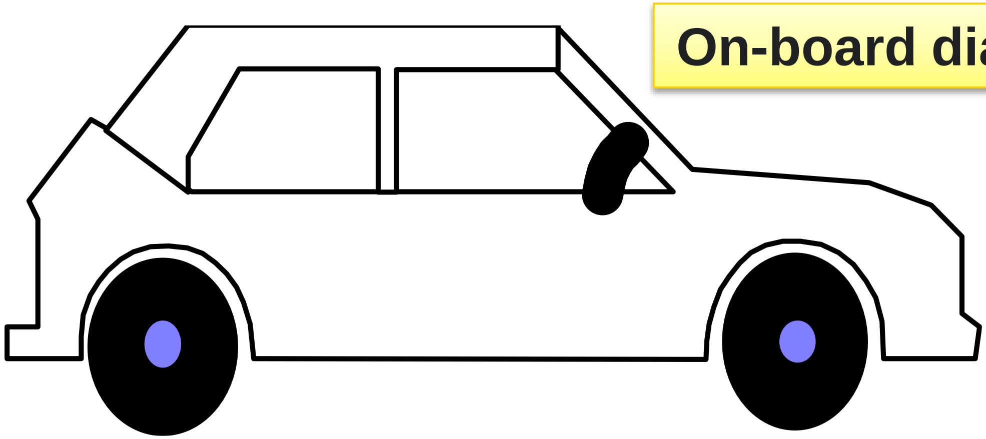
(specification)

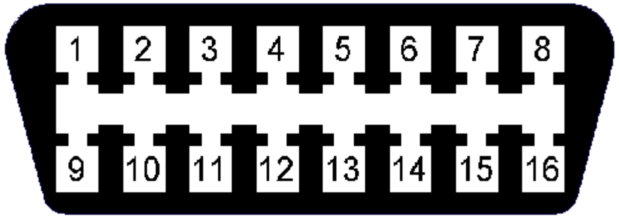
Implementation

(body)

Interface Versus Implementation

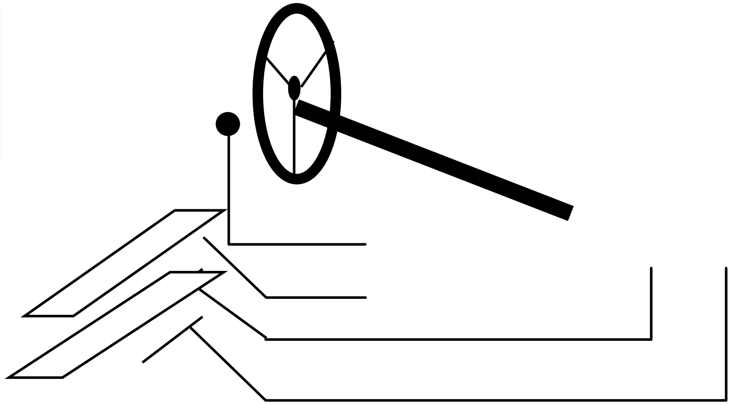
On-board diagnostics (OBD)





OBD

Business Interface

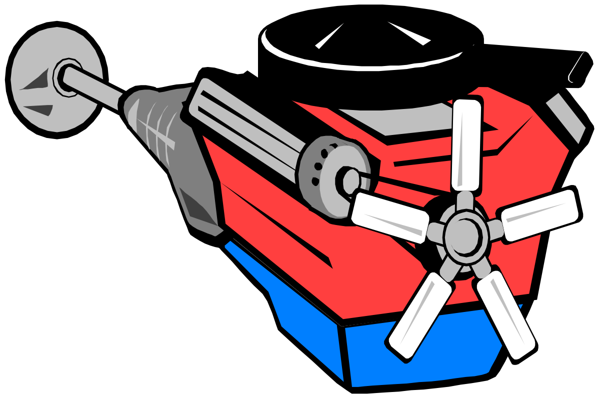


Interface

(specification)

Technical Interface

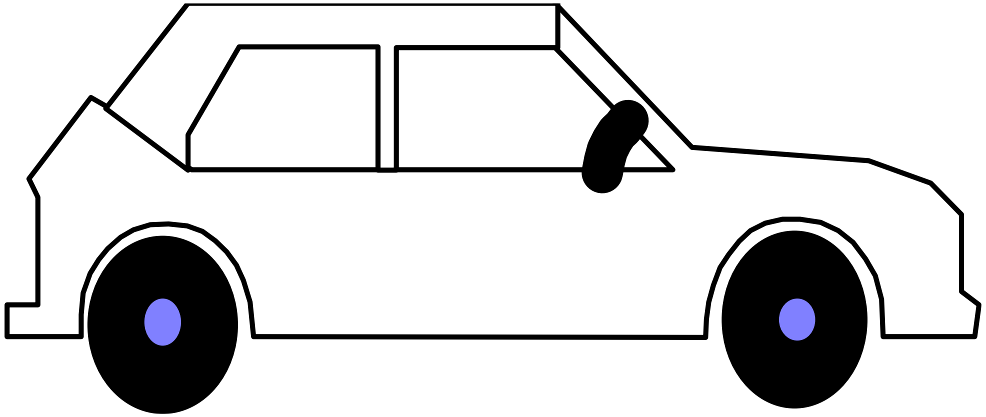
OBD

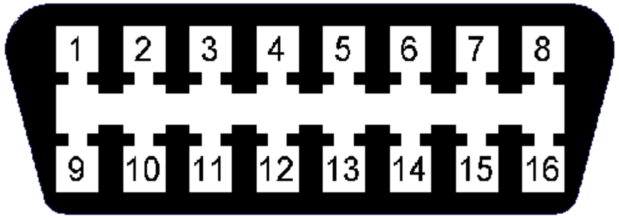


Implementation

(body)

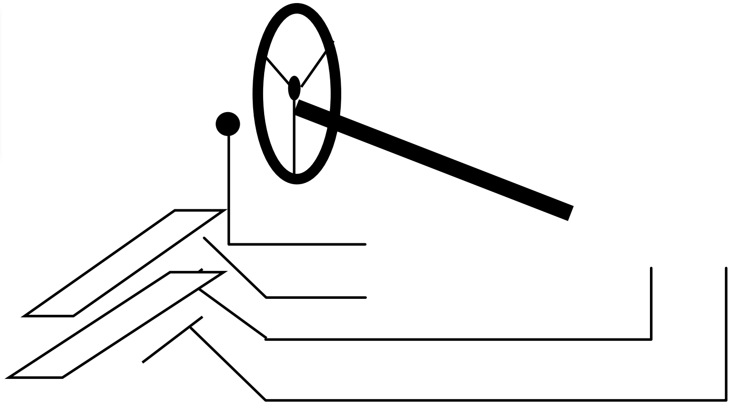
Interface Versus Implementation





OBD

Business Interface

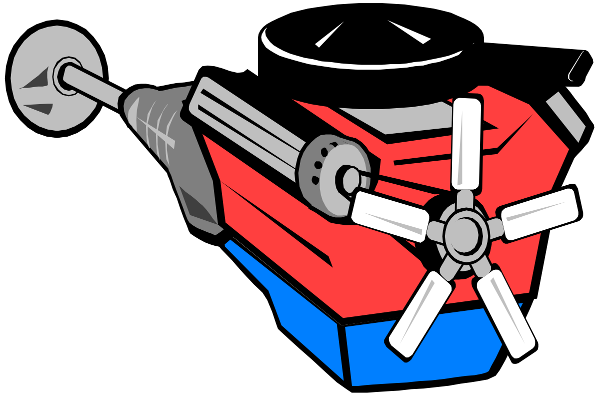


Interface

(specification)

Technical Interface

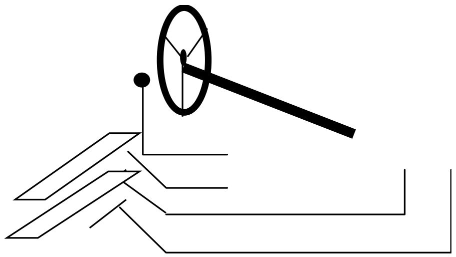
OBD



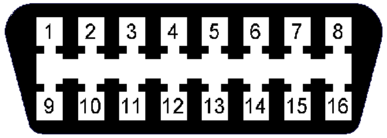
Implementation

(body)

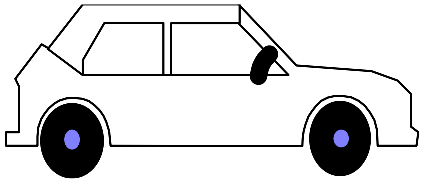
Interface : Separation of concerns



OBD



```
Public interface HashProbing {

    public boolean put(int key, int value);

    public int get(int key);

    public void remove(int key);
}
```

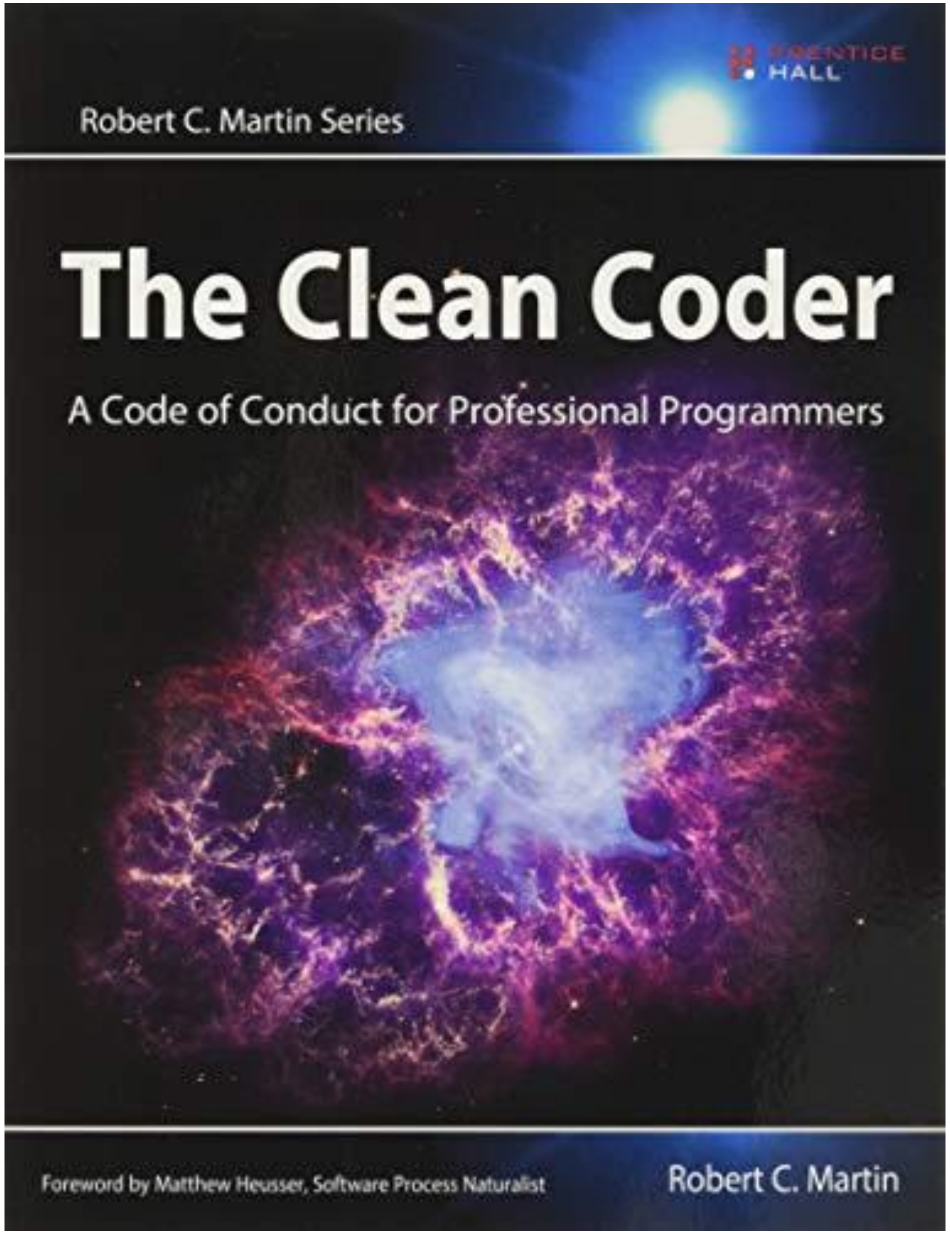


```
Public interface HashProbingTest {

    public int[] keys();

    public int[] values();

    public double load();

    public int hash(int key);
}
```

OBD

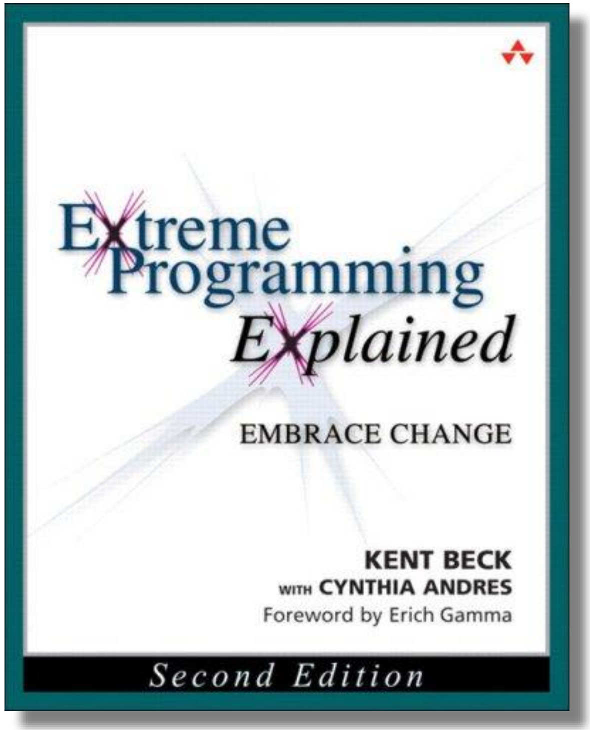
```
Public Class HashProbing {

    ....

                   ....
}
```

Robert Cecil Martin


PRENTICE HALL
Robert C. Martin Series
The Clean Coder
A Code of Conduct for Professional Programmers
Foreword by Matthew Heusser, Software Process Naturalist
Robert C. Martin

Junit

- Créé par Kent Beck (XP) et Erich Gamma (GOF)



Extreme Programming Explained

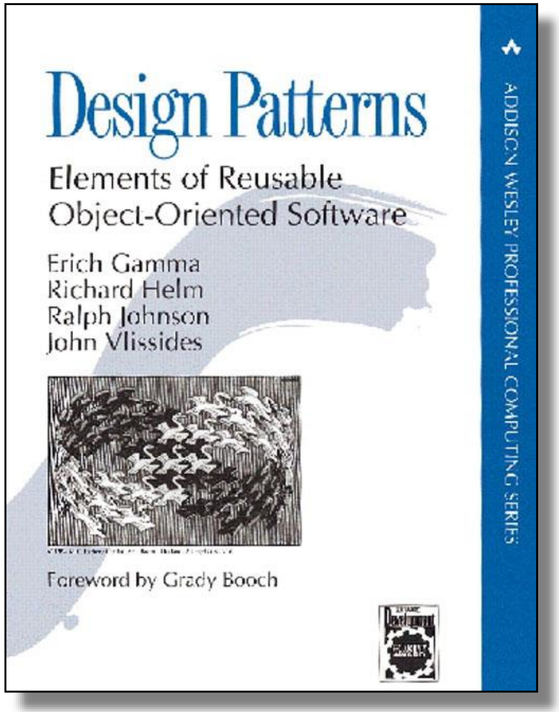
EMBRACE CHANGE

KENT BECK

WITH CYNTHIA ANDRES

Foreword by Erich Gamma

Second Edition



Design Patterns

Elements of Reusable Object-Oriented Software

Erich Gamma

Richard Helm

Ralph Johnson

John Vlissides

Foreword by Grady Booch

ADDISON WESLEY PROFESSIONAL COMPUTING SERIES

SOLID principle

- The Single Responsibility Principle (SRP)
  - Classes should have a single responsibility and thus only a single reason to change.
- The Open/Closed Principle (OCP)
  - Classes and other entities should be open for extension but closed for modification.
- The Liskov Substitution Principle (LSP)
  - Objects should be replaceable by their subtypes.
- The Interface Segregation Principle (ISP)
  - Interfaces should be client specific rather than general.
- The Dependency Inversion Principle (DIP)
  - Depend on abstractions rather than concretions.

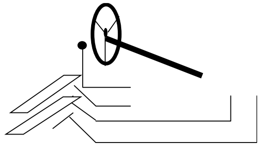
SOLID principle

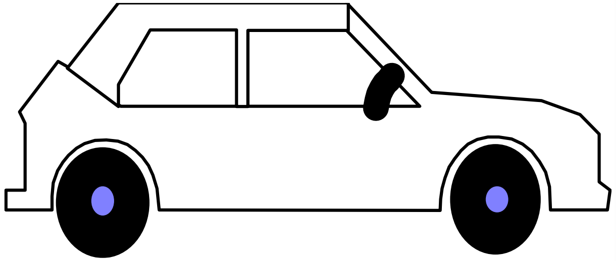
- The Interface Segregation Principle (ISP)
  - Interfaces should be client specific rather than general.

- The Dependency Inversion Principle (DIP)
  - Depend on abstractions rather than concretions

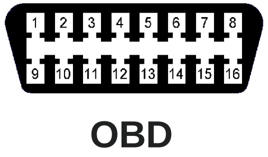
Interface Versus Implementation

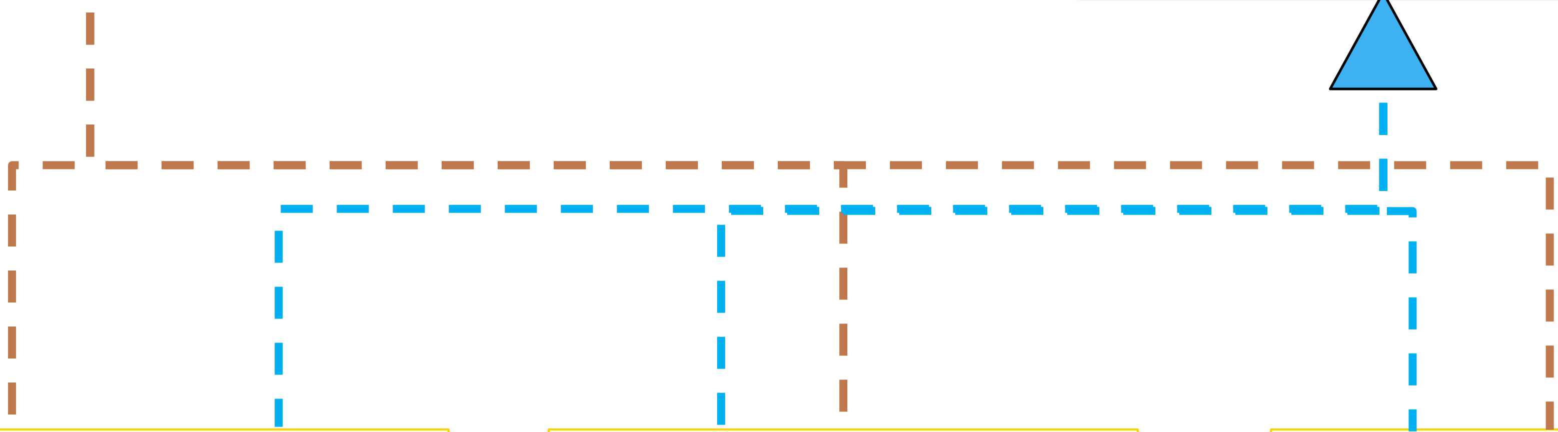
```
Public interface HashProbing {

    public boolean put(int key, int value);

    public int get(int key);

    public void remove(int key);
}
```





```
Public interface HashProbingTest {

    public int[] keys();

    public int[] values();

    public double load();

    public int hash(int key);
}
```



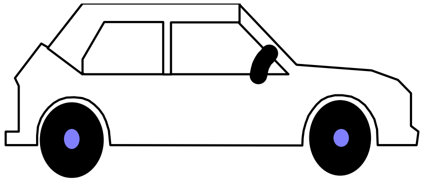


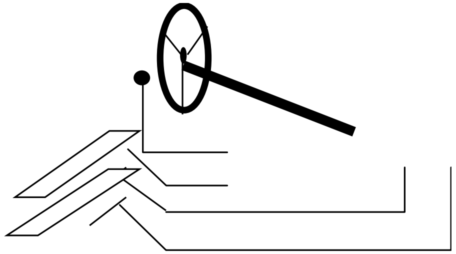
```
Public Class HashProbingLinear
{
    ....

                ....
}
```

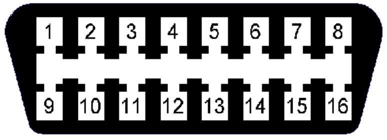
```
Public Class HashProbingQuadratic
{
    ....

                ....
}
```

```
Public Class HashProbingDouble
{
    ....
                ....

}
```

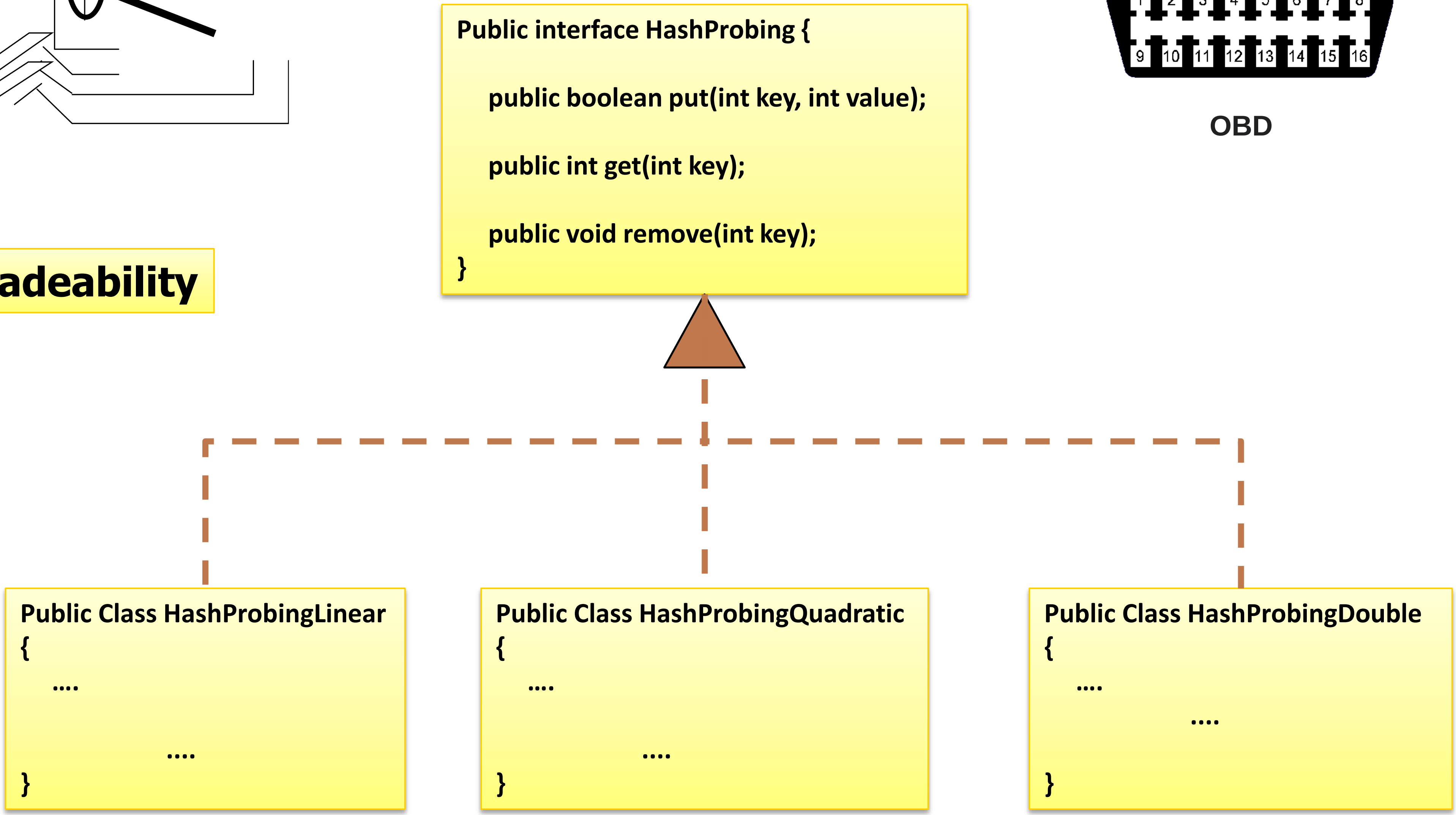
Interface Versus Implementation







OBD



```
Public interface HashProbing {

    public boolean put(int key, int value);

    public int get(int key);

    public void remove(int key);
}
```

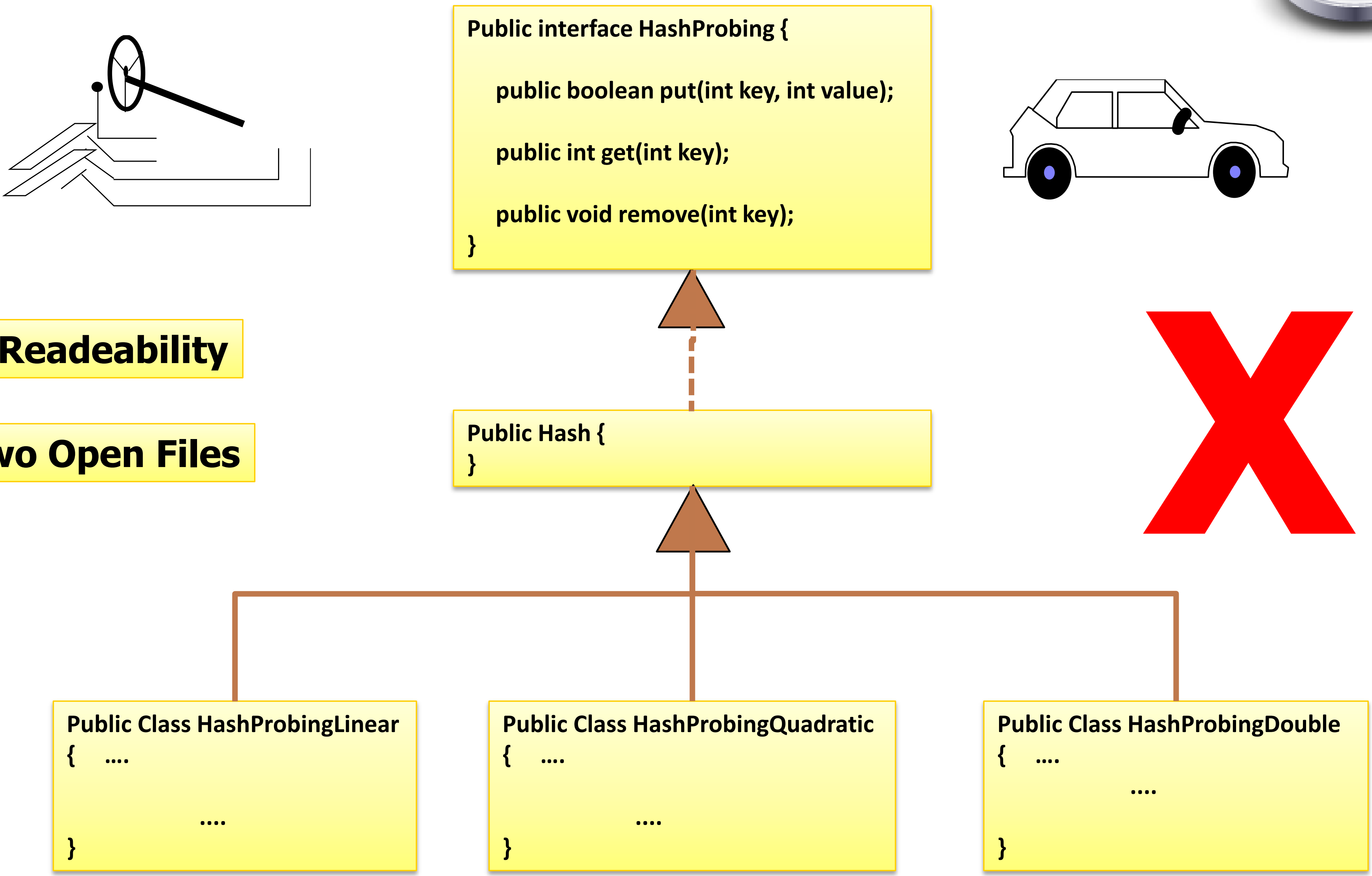
```
Public Class HashProbingLinear
{
    ....

                ....
}
```

```
Public Class HashProbingQuadratic
{
    ....

                ....
}
```

```
Public Class HashProbingDouble
{
    ....
                ....

}
```

+ Readeability

Interface Versus Implementation



```
Public interface HashProbing {

    public boolean put(int key, int value);

    public int get(int key);

    public void remove(int key);
}
```

```
Public Hash {
}
```

```
Public Class HashProbingLinear
{    ....

               ....
}
```

```
Public Class HashProbingQuadratic
{    ....

               ....
}
```

```
Public Class HashProbingDouble
{    ....
               ....

}
```

+ Readeability

Two Open Files

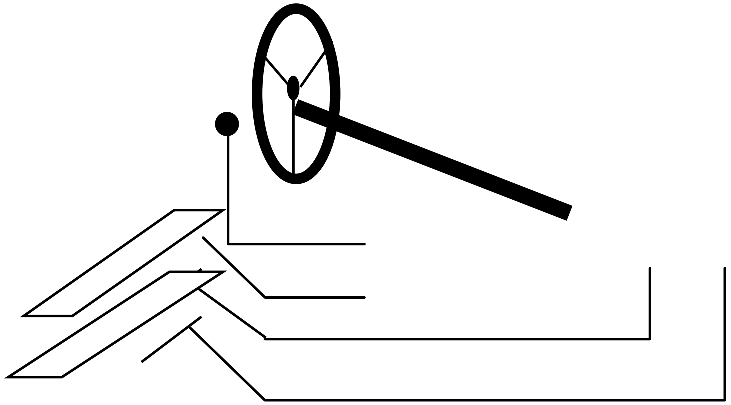
Interface Versus Implementation

No Implementation details In the Interface

```
Public interface HashProbing<K, V> {

        void put(K key, V value);

        int get(K key);

        void remove(K key);

}
```

X

+ Readeability



Interface

(specification)

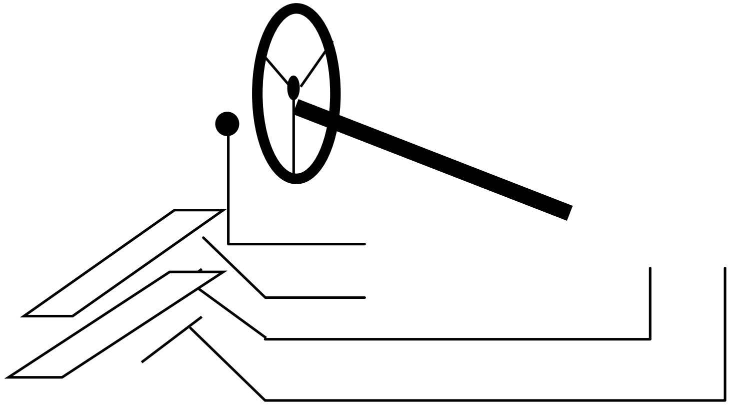
Genericity is an implementation choice

Genericity at the end of the process

Interface Versus Implementation

No Implementation details In the Interface

```
Public interface HashProbing<K, V> {

            void put(K key, V value);

            int get(K key);

            void remove(K key);

}
```



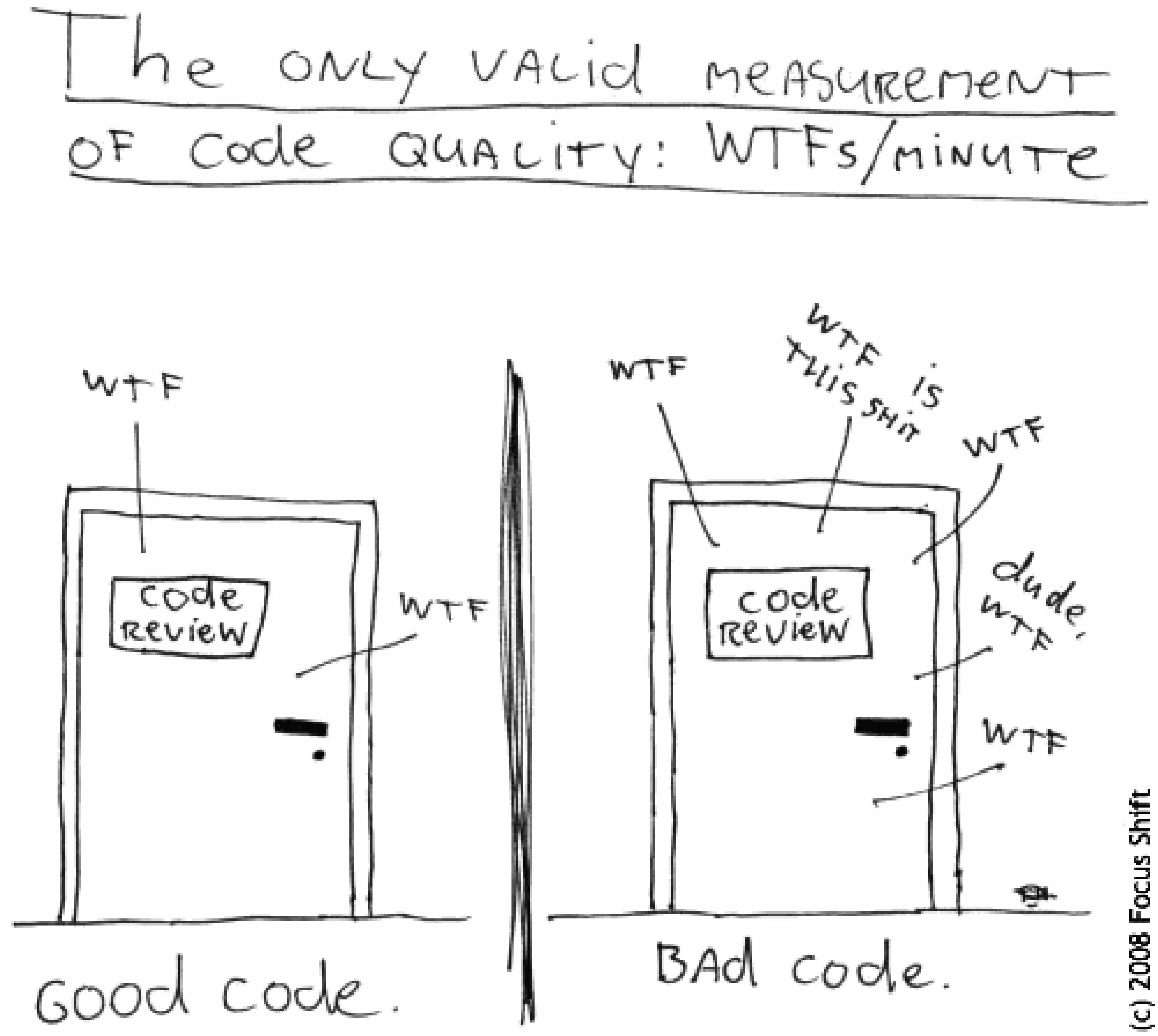
X

Interface

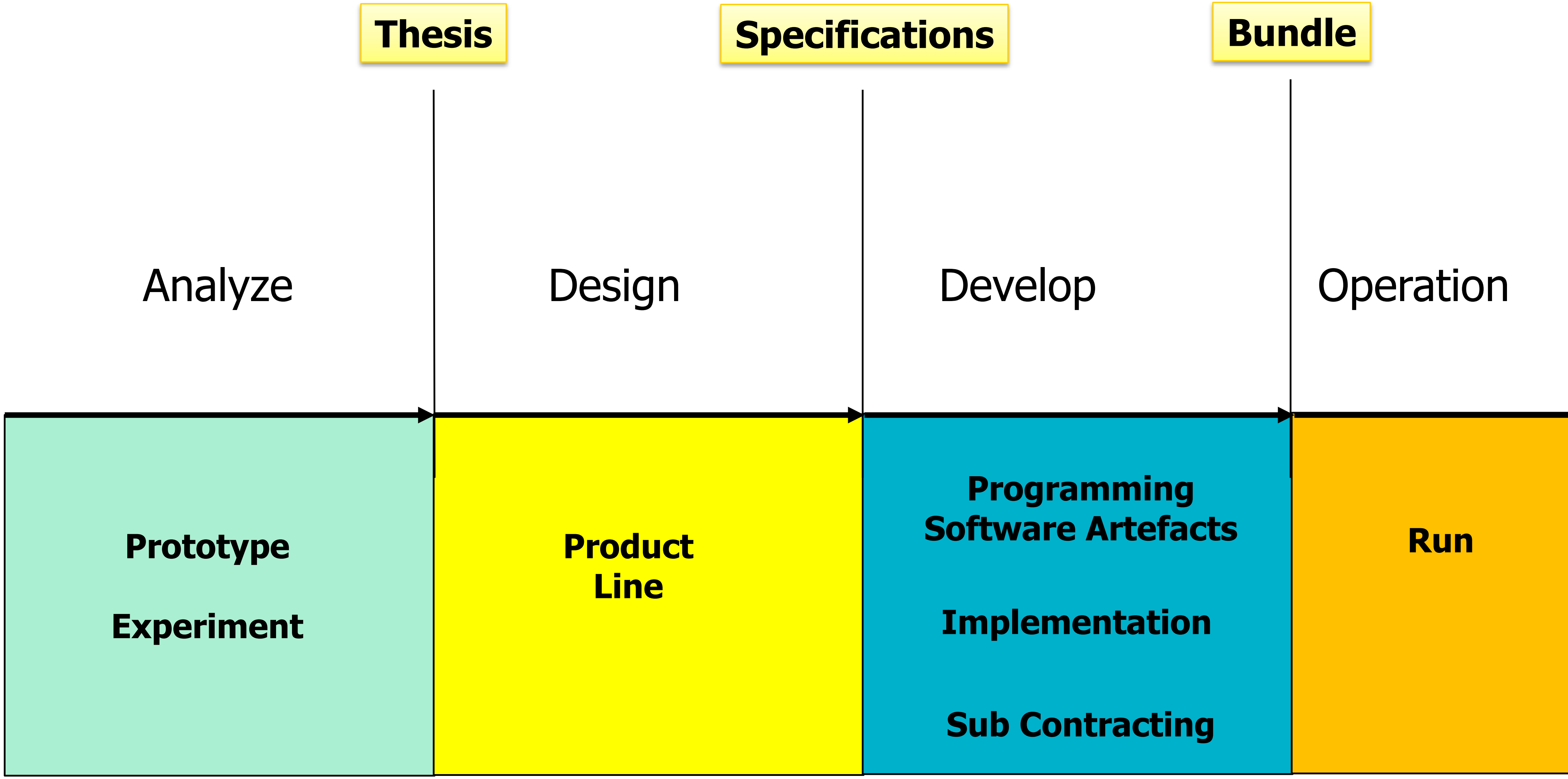
(specification)

Genericity is not applicable in client/server

Interface Repository


The only valid measurement of code quality: WTFs/minute
WTF
code review
WTF
Good code.
WTF
WTF is this shit
WTF
code review
dude, WTF
WTF
BAd code.
(c) 2008 Focus Shift

Software Life Cycle



Thesis

Specifications

Bundle

Analyze

Design

Develop

Operation

Prototype

Experiment

Product Line

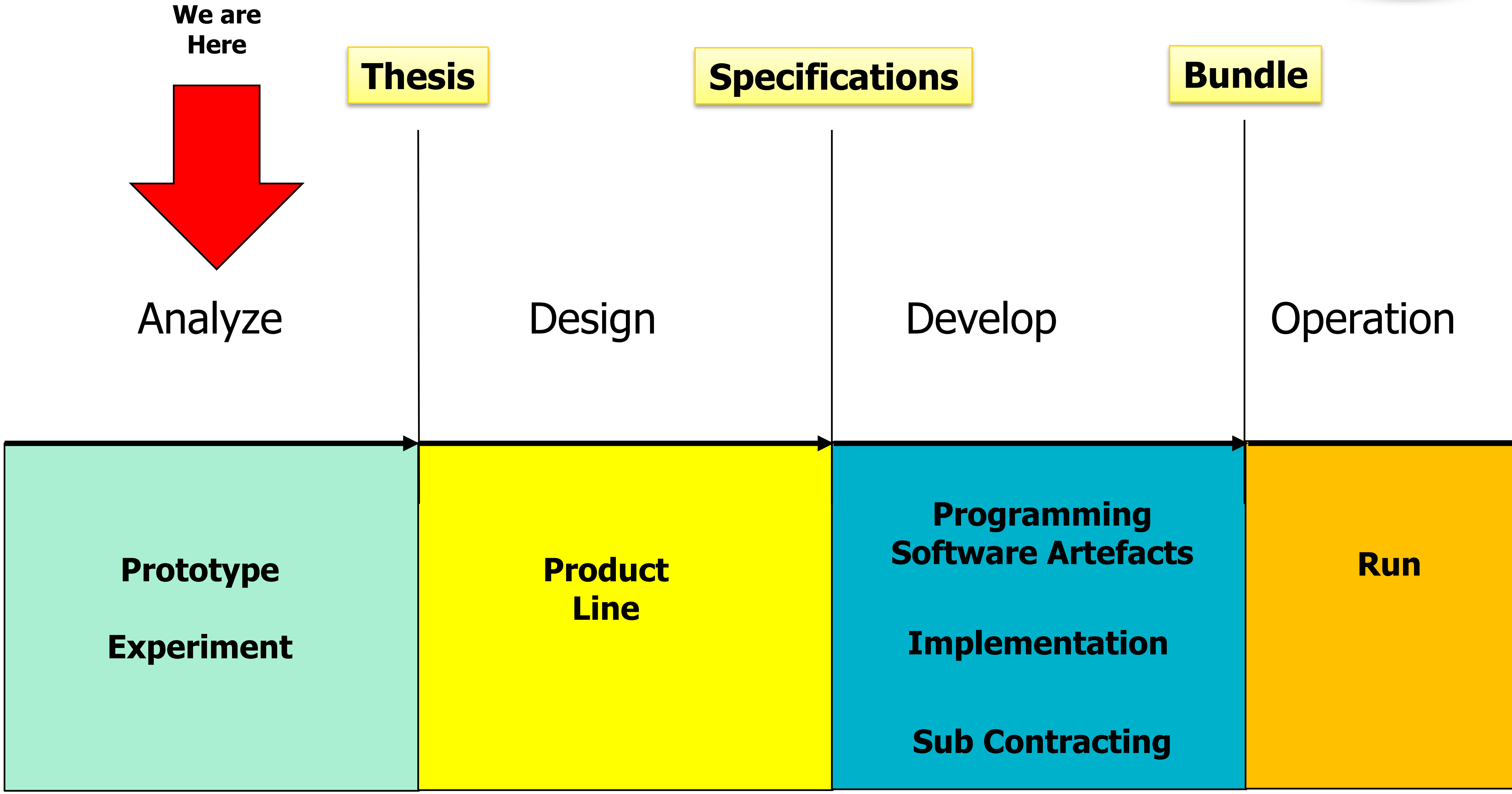
Programming Software Artefacts

Implementation

Sub Contracting

Run

Software Life Cycle



We are Here

Thesis

Specifications

Bundle

Analyze

Design

Develop

Operation

Prototype

Experiment

Product Line

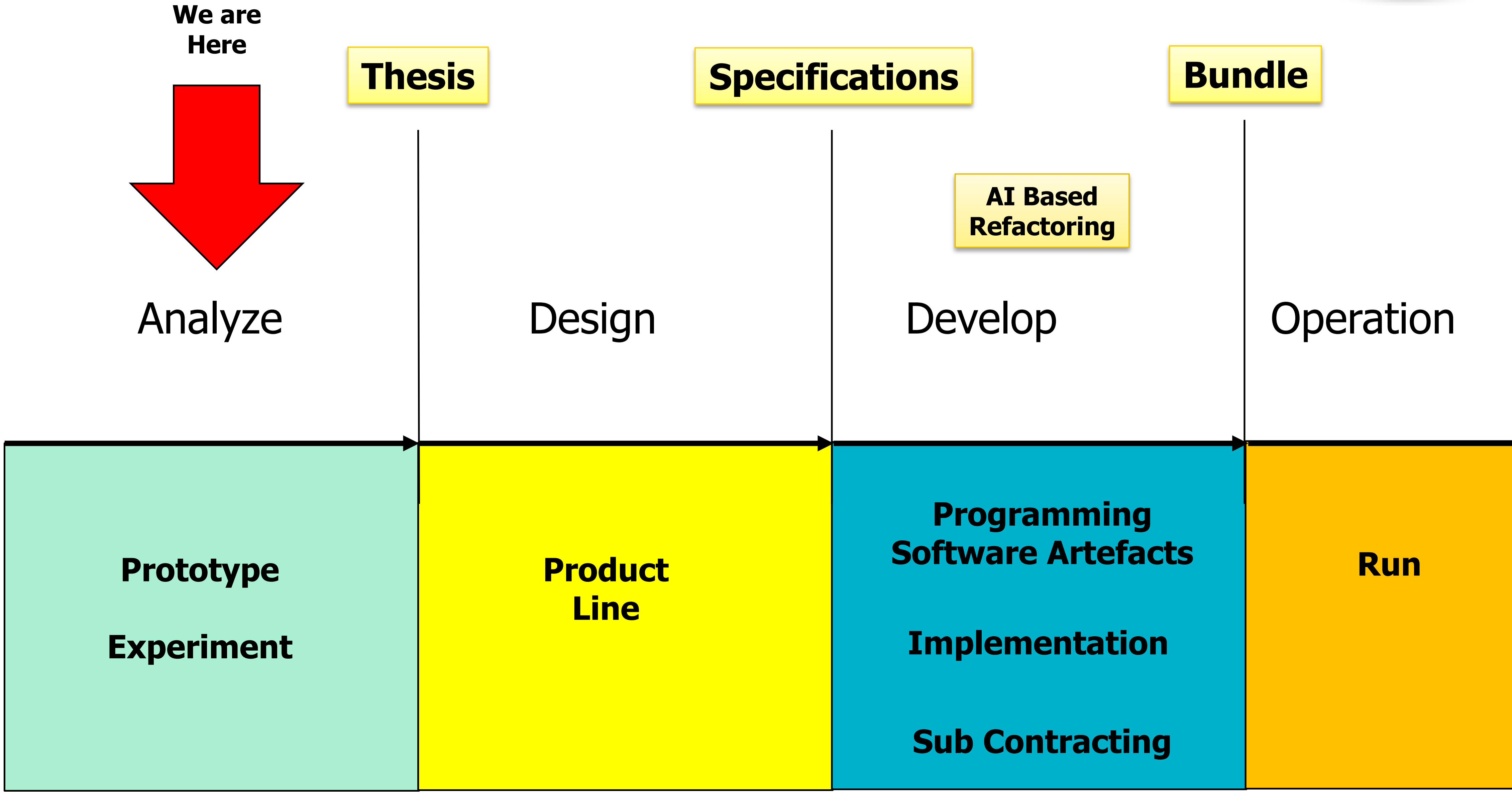
Programming Software Artefacts

Implementation

Sub Contracting

Run

Software Life Cycle



We are Here

Thesis

Specifications

Bundle

AI Based Refactoring

Analyze

Design

Develop

Operation

Prototype

Experiment

Product Line

Programming Software Artefacts

Implementation

Sub Contracting

Run

Java 8 Interface evolutions : default

```
public interface HashProbing {
		public static final int M = 11;
		public char [] keys = new char[M];
		public int [] values = new int[M];

		default int hash(char key) {
				int asciiCode = (int) key - 64;
				return (asciiCode % M);
		}
```

It is not an Interface !

No more encapsulation

Java 8 Interface evolutions : default

```
public interface HashProbing {
        public static final int M = 14;
        public char [] keys = new char[M];
        public int [] values = new int[M];

        default int hash(char key) {
                int asciiCode = (int) key - 64;
                return (asciiCode % M);
        }
```

Java was technical
Now it is political

No more encapsulation

Java 8 Interface evolutions : private

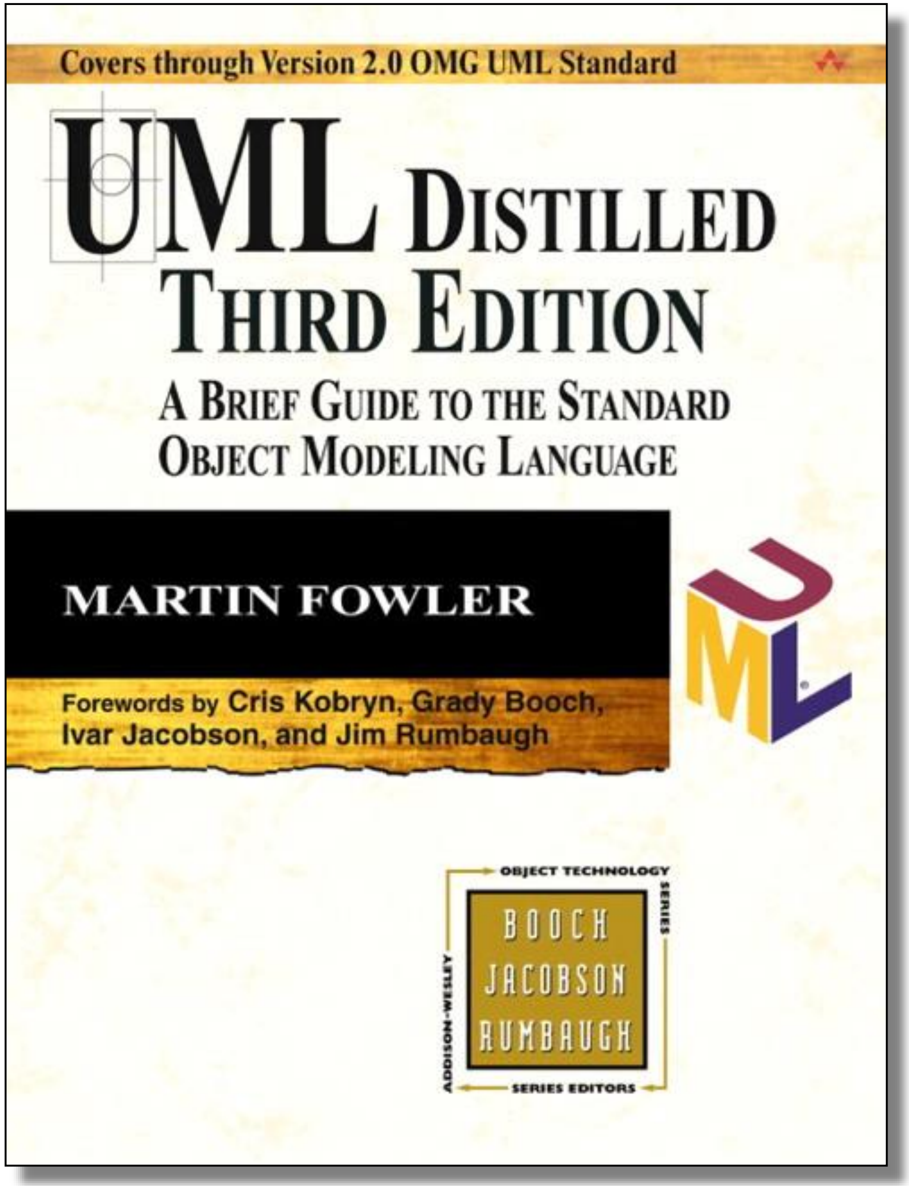
X

```
public interface HashProbing {

      private void method4() {
            System.out.println("private method");
      }
}
```

Indeed

It is not an Interface !

Martin Fowler (1997)



Covers through Version 2.0 OMG UML Standard

UML DISTILLED

THIRD EDITION

A BRIEF GUIDE TO THE STANDARD OBJECT MODELING LANGUAGE

MARTIN FOWLER

Forewords by Cris Kobryn, Grady Booch, Ivar Jacobson, and Jim Rumbaugh

OBJECT TECHNOLOGY SERIES

BOOCH JACOBSON RUMBAUGH

ADDISON-WESLEY

SERIES EDITORS

Changes Sources During Development

- Requirements :
  - Customers Discover What they Really Want During or at the End of Developments
- Technology
  - Performances Are Increasing With Time
- Skill
  - We Learn and Understand the Problem and We Discover the Right Solution on the Job
- Short Term Politic
  - No Comments

Martin FOWLER

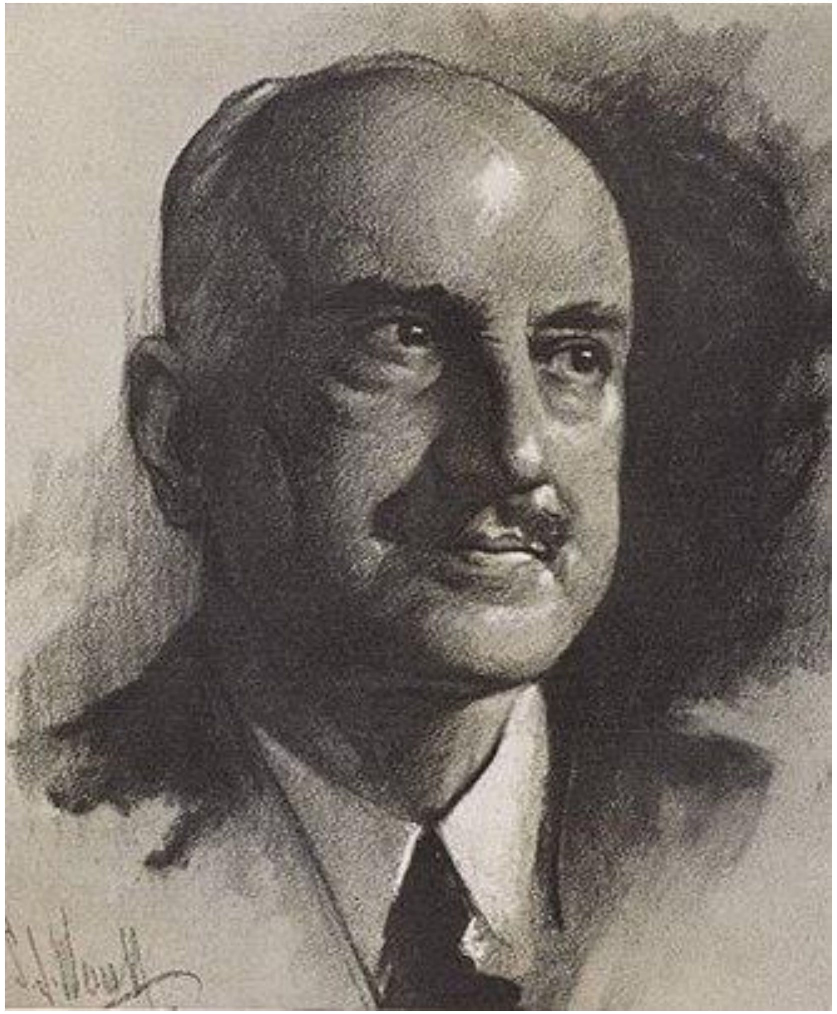
George Santayana

(December 16, 1863 – September 26 1952)

George Santayana (1905) Reason in Common Sense, p. 284, volume 1 of The Life of Reason

"Those who cannot remember the past are condemned to repeat it”

Studying history is necessary to avoid repeating past mistakes.



Java History

- Conceived of by James Gosling in 1995.

On January 27, 2010 Sun Microsystems was acquired by the Oracle Corporation.

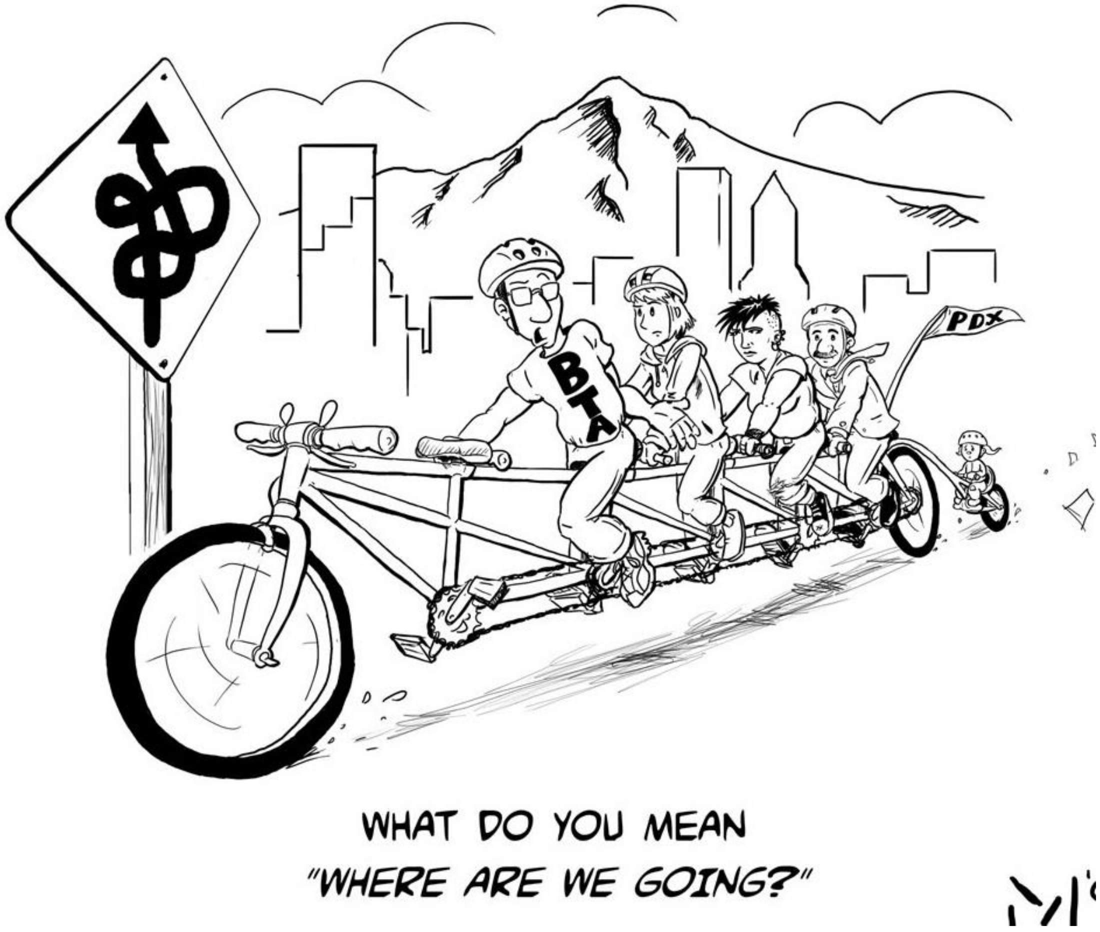
Java useless political evolutions

- Marketing Competition with :
  - .NET C#
  - PHP
  - Python
  - Javascript/Node.js
  - Go
  - Kotlin
  - ..
  - .....
  - A new competitor every six months

Java useless political evolutions

- Design by committee anti pattern

I AM SURE WE CAN COME UP WITH A COMPROMISE
I CAN'T STAND THE COLOR!
I AM NOT SURE ABOUT THE LAYOUT..
I LIKE THE BLUE!



Compromise of compromises

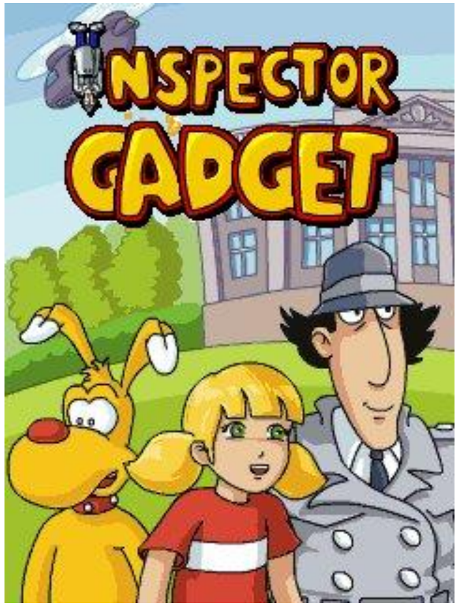
No leadership

Java useless political evolutions

- .NET (Microsoft)
  - Microsoft
  - Windows
    - ASP : Active Server Pages
    - Webforms
- Java EE : Java Enterprise Edition
  - Sun, Java
    - JSP : Java Server Page
    - JSF : Java Server Face
  - Open source
- ....
- ...

Java useless political evolutions

- Java language syntaxic and idiomatic evolutions are:
  - Gadget
  - Toy
  - …



LEGO Toying vs House Building

Java useless political evolutions

- Annotations : @ (++ pluggable annotations)
- Enum (V5) : Polymorphism Anti Pattern
- JDBC : Toy, separation of concerns
- Stream : Toy, separation of concerns
- Json : Interoperability Abstraction Break
- Interface default (V8) : Abstraction Anti Pattern
- Interface private (V9) : Interface Anti Pattern
- Local-Variable Type Inference (V10) : Safety Anti Pattern
- HTTP client (V12) : Toy, separation of concerns
- ….
- Records (V14) : OOD Anti Pattern
- ….
- ….

LEGO TOY vs House Building

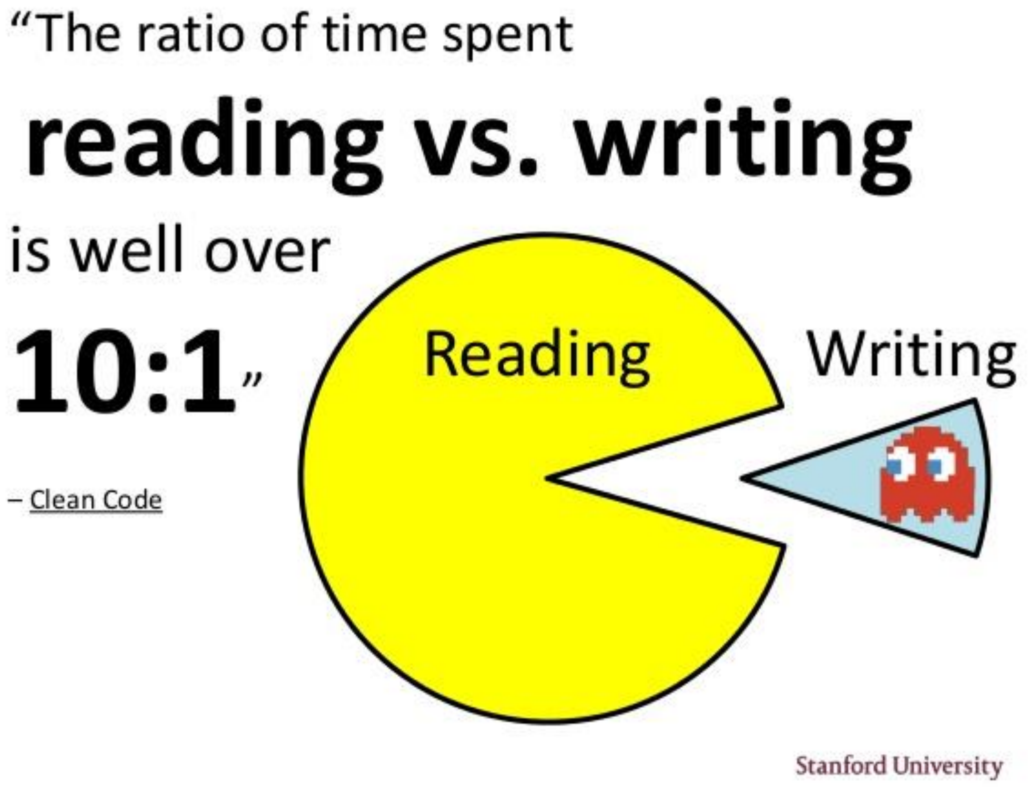
Java useless political evolutions

- New features !!!
- Syntactic sugar
  - Writing faster

    - « It Doesn't Matter How Fast You Write Code, what’s matter is Readability »

Java useless political evolutions

- The code is usually written once but read many times.
- Emphasis has to be placed on program readability over ease of writing.



“The ratio of time spent **reading vs. writing** is well over **10:1**”

– Clean Code

Stanford University

Java useless political evolutions

- Syntactic sugar : Writing faster

- Program variables shall be explicitly declared and their type be specified.

- Since the type of a variable is invariant, compilers can ensure that operations on variables are compatible with the properties intended for objects of the type.

Java useless political evolutions

- Syntactic sugar : Writing faster
- Error-prone notations have to be avoided.
- The syntax of the language shall avoided the use of encoded forms in favor of more English-like constructs.
- No ternary operators

Java useless political evolutions

- Quotes

- « Making code easy to read makes it easier to write. »

Software Handoff