



# Objectifs du module

**O1** : Connaître les principes de mise en œuvre d'une approche **qualité** dans le processus de production du logiciel.

- **CM4** : Mettre en œuvre une approche qualité dans le processus de production du logiciel.
- **C5** : Qualité du logiciel : objectifs du système logiciel ; assurance qualité, normes, gestion des projets logiciels, documentation, cycle de vie du logiciel, architecture logicielle.
- **C6** : Principes et techniques de base des tests : familles et niveaux de tests.
- **C8** : Interaction homme-machine : prise en compte de l'utilisateur, conception de l'I.H.M., composants graphiques, choix et recommandations ergonomiques.



# Evaluation du module

**Seule une page au format A4, recto-verso, sera autorisée pour l'examen final qui portera essentiellement sur une étude de cas.**

Les **TD** se font en groupe. Ils donnent lieu en fin de module à des rendus notés par groupe.

Les **TP** se font en groupe. Ils donnent lieu en fin de module à des rendus notés par groupe.

Des **études sur des sous-thématiques** (nouvelles IHMs, agilité et web, ...) sur la base d'articles ou de livres pourront également être rendues et notées, pour un bonus sur la note de TD-TP.

Un **examen final**

# Evaluation du module

Seule une page au format A4, recto-verso, sera autorisée pour l'examen final qui portera essentiellement sur une étude de cas.

## Votre objectif ?

Les TD se font en groupe. Ils donnent lieu en fin de module à des rendus notés par groupe.

- Comprendre les principes de l'analyse / conception
- Respecter les normes : nommage des associations, ...

Les TP se font en groupe. Ils donnent lieu en fin de module à des rendus notés par groupe.

- Respecter les règles imposées : rendus à temps avec tout ce qui est demandé,

Des études sur des sous-thèmes (des IHMs, agilité et web, ...) sur la base d'articles ou de livres pourront également être rendues et notées, pour un bonus sur la note de TD-TP.

- format de rendu, ...

**Avoir la meilleure note possible pour passer en S4D !!**

Un examen final

# Plan du module

- ◆ Qualité du logiciel
- ◆ Tests ....
  - ◆ Introduction
  - ◆ Développement dirigé par les tests.
  - ◆ Mocks...
- ◆ Méthodes
  - ◆ Méthodes Agiles (XP, Scrum)
  - ◆ RUP
- ◆ IHMs

Petits rappels après les ~~vacances~~  
*projets*

Modélisation, UML  
Quiz

# Question

UML ?

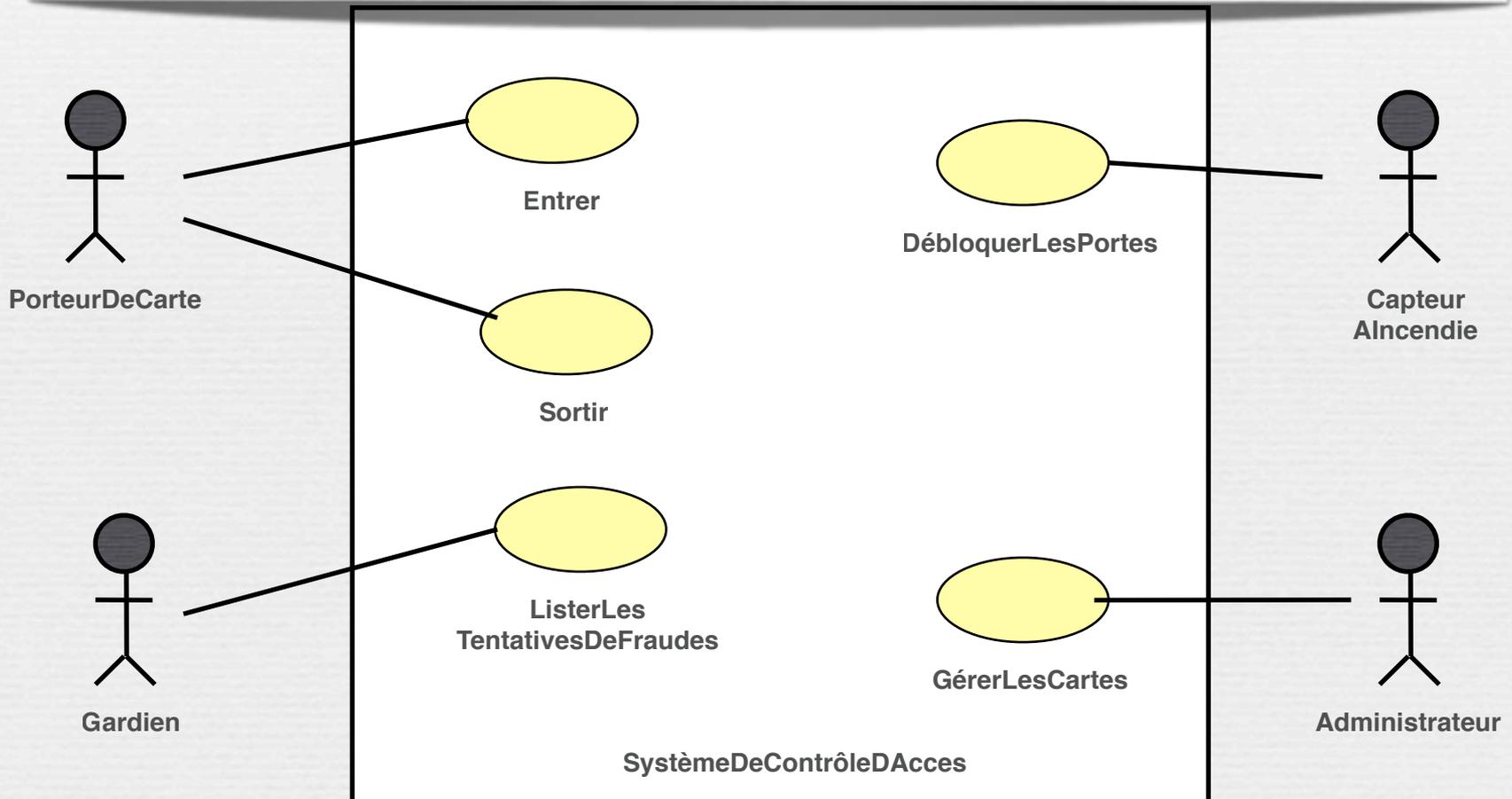
- A. Un nouvel objet non identifié
- B. Un nouveau langage
- C. Unified Modelling Language

Réponse :

# Qu'est-ce qu'UML ?

## Diagrammes des cas d'utilisation

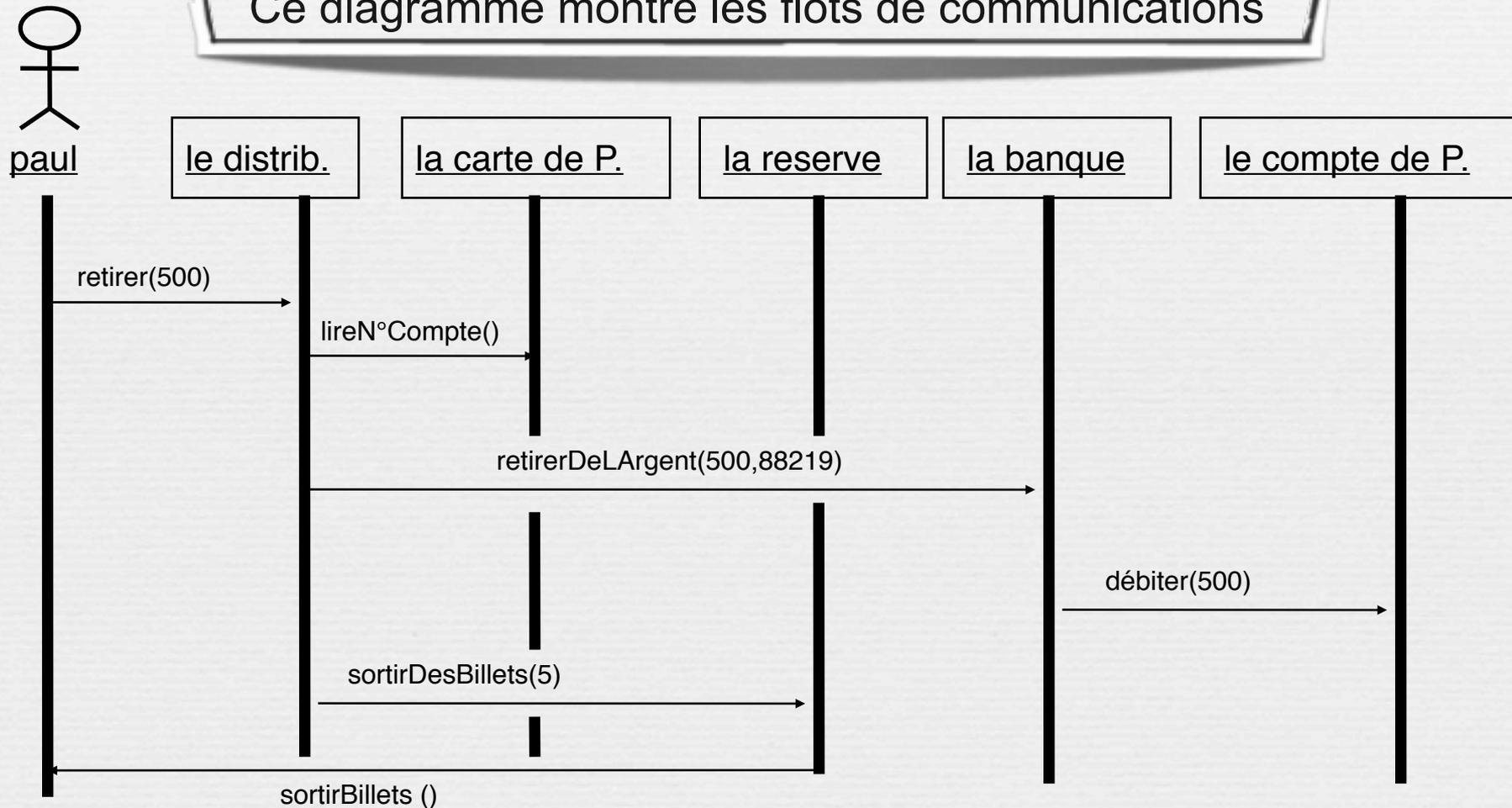
Ce diagramme montre ce que fait le système et qui l'utilise



# Qu'est-ce qu'UML ?

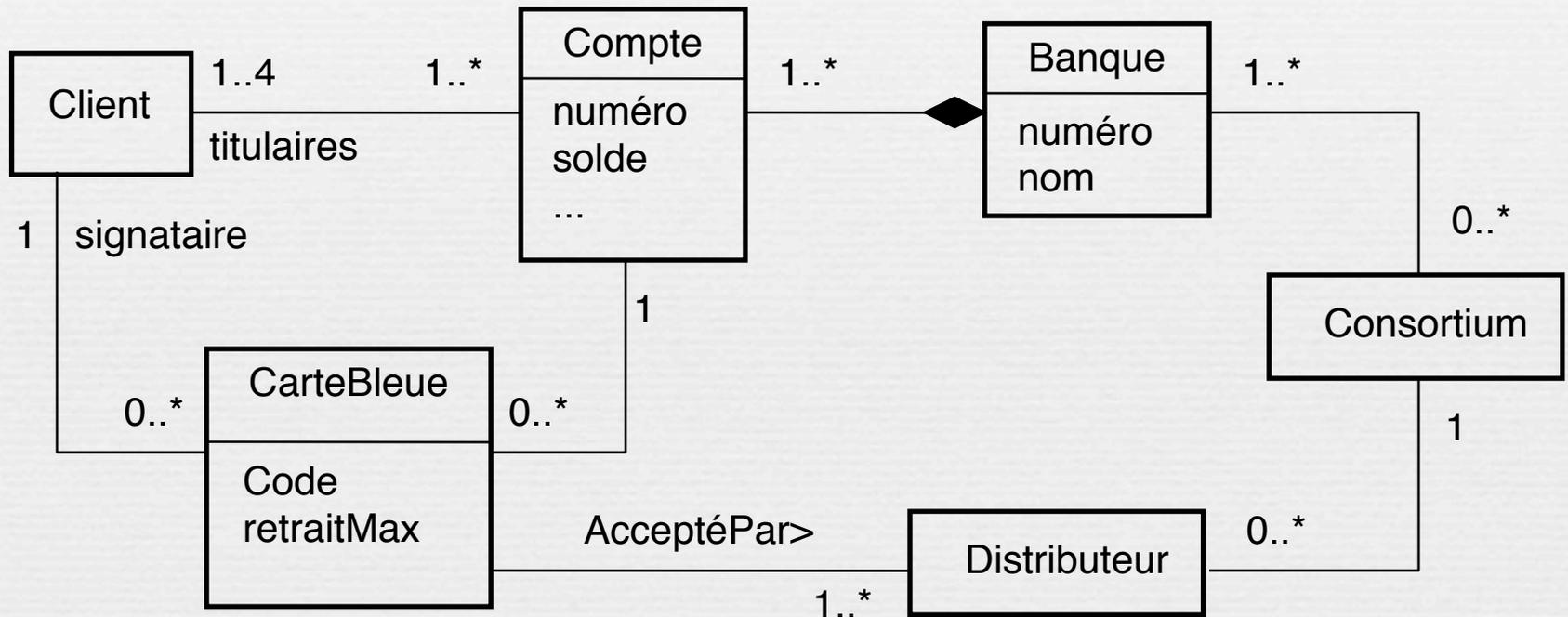
## Diagrammes de séquence

Ce diagramme montre les flots de communications



# Qu'est-ce qu'UML ?

## Diagrammes de classes



Ce diagramme montre les classes et les relations entre elles

# Qualité du logiciel ?

Merci à tous ceux qui ont rendu leurs cours et exposés  
disponibles sur le web & dans les livres,  
voir Biblio.

M. Blay-Fornarino  
blay@unice.fr,  
<http://users.polytech.unice.fr/~blay/>  
IUT Département Informatique 2<sup>e</sup> année

# Bibliographie

- ❧ Reflexion on Software Quality and Maintenance, Alexandre Bergel, Chili
- ❧ Cours de Production Du Logiciel, La qualité logicielle, Thierry Milan, Toulouse
- ❧ Yann-Gaël Guéhéneuc cours , Université de Montréal, <http://www-etud.iro.umontreal.ca/~ptidej/yann-gael/Work/Publications/>

# Le logiciel ...

S'il vous fallait préparer à l'avance tous les ordres à donner à des individus totalement stupides, mais absolument obéissants, pour qu'ils réalisent une tâche complexe, vous diriez que c'est là un problème de management nouveau : les gens ne sont, en général, ni stupides, ni obéissants, ce qui aide à faire face à l'imprévu. Si, en plus, la moindre erreur provoque une catastrophe, vous diriez que la réussite d'un tel plan d'action tient du miracle. Vous venez pourtant de définir ce qu'est la fabrication d'un logiciel. Pour que la réussite ne tienne pas que du miracle, il faut donc une rigueur de fer, pas mal de culture, et de bons outils tels que ceux qui apparaissent aujourd'hui, ce qui n'est malheureusement pas encore assez connu.

**LOGICIELS : COMMENT CHASSER LES BUGS ?**

par

**Gérard BERRY 1996**

Directeur de Centre de Mathématiques Appliquées  
de l'École des mines de Paris

membre de l'Académie des sciences française (depuis 2002),  
de l'Académie des technologies (depuis 2005),  
et de l'Academia Europaea (depuis 1998)

# Qualité ?

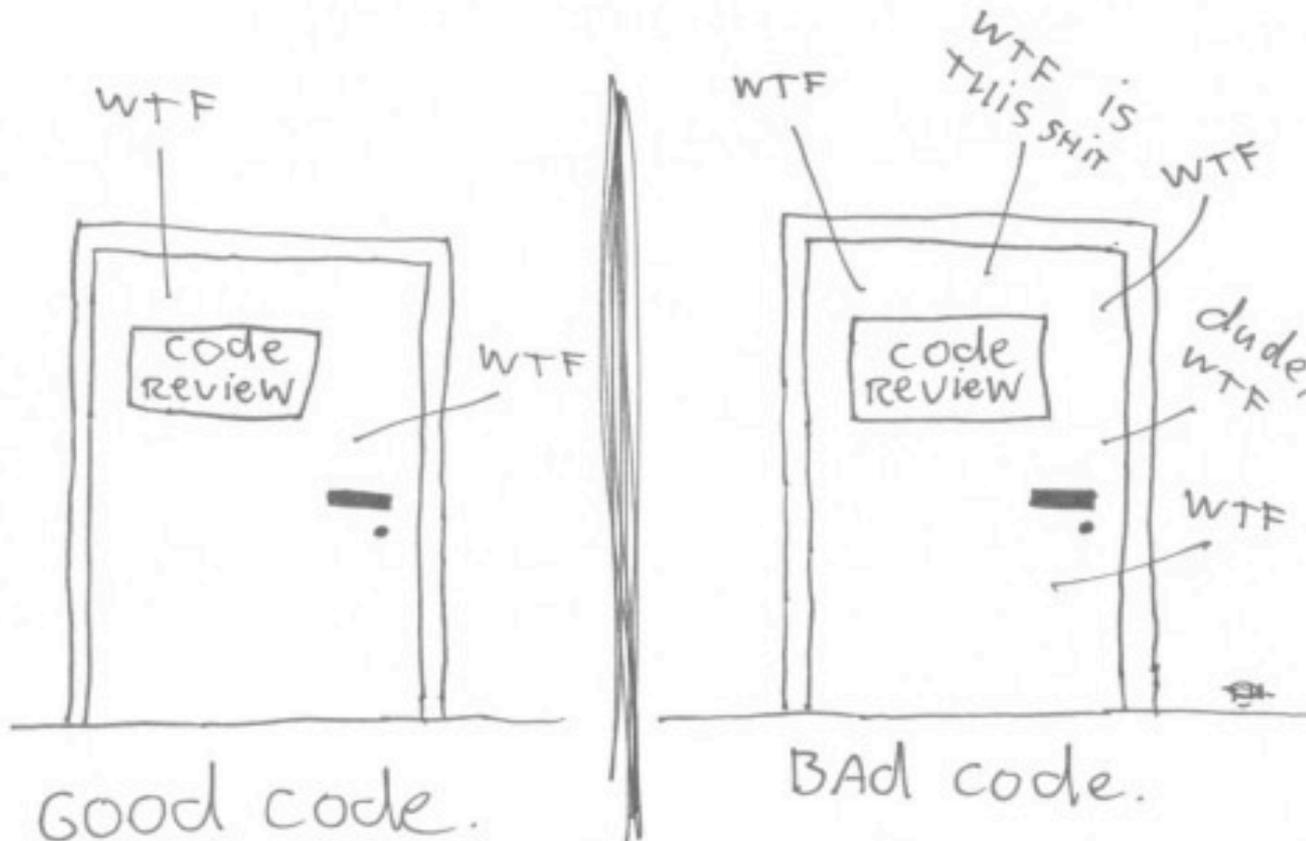
- Exemple
- On compare deux logiciels
  - Le premier a encore 10 erreurs résiduelles
  - Le second n'a que trois erreurs résiduelles
- Est-ce que le second est de meilleure qualité que le premier?

- La qualité pour n'importe quel processus est difficile à définir
- Il faut identifier
  - Des facteurs
  - Des critères
  - Des métriques

# Evaluation de la Qualité ?

# Evaluation

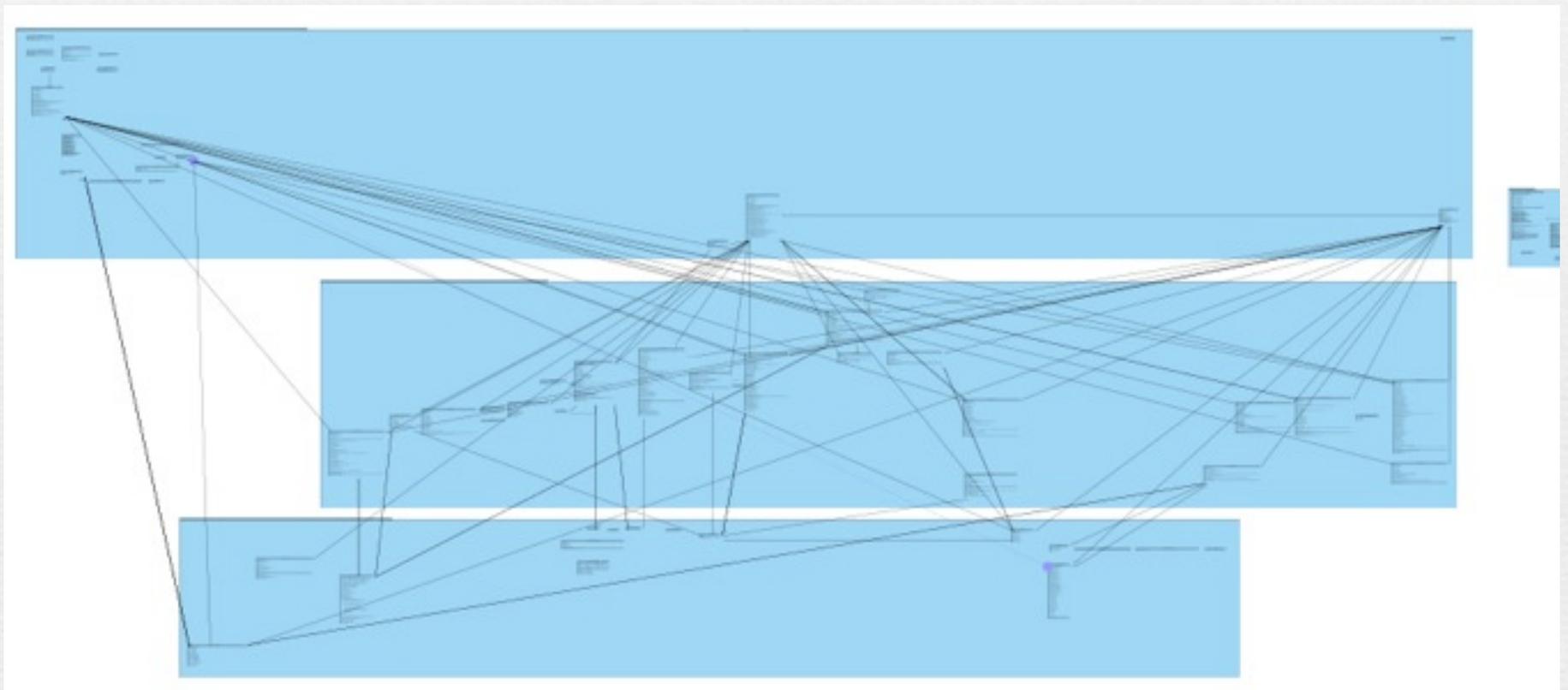
The ONLY VALID MEASUREMENT  
OF Code QUALITY: WTFs/minute



(c) 2008 Focus Shift/OSNews/Thom Holwerda - <http://www.osnews.com/comics>

# Evaluation

Code à «améliorer» au cours d'un stage IUT 2011





# Intuitivement, la qualité ...

à partir du cours d'Alexandre Bergel

# QUALITÉ

```
/* Hello World program */  
  
#include<stdio.h>  
  
int main()  
{  
    printf("Hello World");  
  
}
```

**no return  
statement  
in the  
function**



```
/* Hello World program */  
  
#include<stdio.h>  
  
int main()  
{  
    printf("Hello World");  
  
}
```

**no return  
statement  
in the  
function**

```
/* Hello World program */  
  
#include<stdio.h>  
  
→ int main()  
  {  
    printf("Hello World");  
  }
```

9 lines of code  
7 seconds

In Mozilla:

dom/base/nsDOMWindowUtils.cpp

```
/* -*- Mode: C++; tab-width: 2; indent-  
tabs-mode: nil; c-basic-offset: 2 -*- */  
/* ***** BEGIN LICENSE BLOCK *****  
 * Version: MPL 1.1/GPL 2.0/LGPL 2.1  
 * * * * *  
#include "nsIDOMHTMLCanvasElement.h"  
#include "nsICanvasElement.h"  
#include "gfxContext.h"  
#include "gfxImageSurface.h"  
* * *
```

**belong to  
the core**

In Mozilla:



**dom/base/nsDOMWindowUtils.cpp**

```
/* -*- Mode: C++; tab-width: 2; indent-  
tabs-mode: nil; c-basic-offset: 2 -*- */  
/* ***** BEGIN LICENSE BLOCK *****  
 * Version: MPL 1.1/GPL 2.0/LGPL 2.1  
 * * * * *  
#include "nsIDOMHTMLCanvasElement.h"  
#include "nsICanvasElement.h"  
#include "gfxContext.h"  
#include "gfxImageSurface.h"  
* * * * *
```

**belong to  
gfx package**

**belong to  
the core**

In Mozilla:



`dom/base/nsDOMWindowUtils.cpp`

```
/* -*- Mode: C++; tab-width: 2; indent-  
tabs-mode: nil; c-basic-offset: 2 -*- */  
/* ***** BEGIN LICENSE BLOCK *****  
 * Version: Mozilla/2.0/GPL 2.1  
...  
#include "nsIDOMHTMLCanvasElement.h"  
#include "nsICanvasElement.h"  
#include "gfxContext.h"  
#include "gfxImageSurface.h"  
...
```

695 lines of code

3 minutes

**belong to  
gfx package**



In Swing:

JComponent.java

```
protected String getBorderTitle(Border b) {  
    String s;  
    if (b instanceof TitledBorder) {  
        return ((TitledBorder) b).getTitle();  
    } else if (b instanceof CompoundBorder) {  
        s = getBorderTitle(((CompoundBorder)  
            b).getInsideBorder());  
        if (s == null) {  
            s = getBorderTitle(((CompoundBorder)  
                b).getOutsideBorder());  
        }  
    }  
    return s;  
    } else {  
        return null;  
    }  
}
```

...

In Swing:

JComponent.java

```
protected String getBorderTitle(Border b) {
    String s;
    if (b instanceof TitledBorder) {
        return ((TitledBorder) b).getTitle();
    } else if (b instanceof CompoundBorder) {
        s = getBorderTitle(((CompoundBorder)
            b).getInsideBorder());
        if (s == null) {
            s = getBorderTitle(((CompoundBorder)
                b).getOutsideBorder());
        }
    }
    return s;
} else {
    return null;
}
```

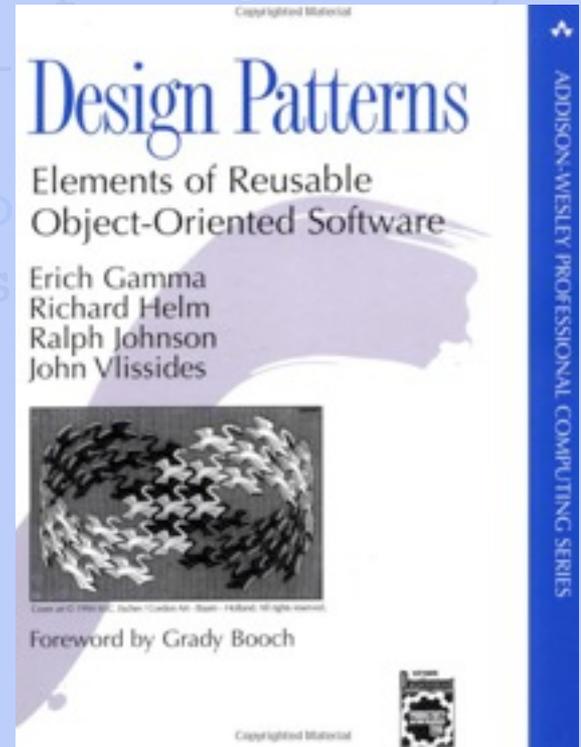
...

In Swing:

JComponent.java

```
protected String getBorderTitle(Border b) {
    String s;
    if (b instanceof TitledBorder) {
        return ((TitledBorder) b).getTitle();
    } else if (b instanceof JComponent) {
        s = getBorderTitle(((JComponent) b).getInsets());
    }
    if (s == null) {
        s = getBorderTitle(((JComponent) b).getOutputStream());
    }
    return s;
} else {
    return null;
}
```

## Need Double Dispatch



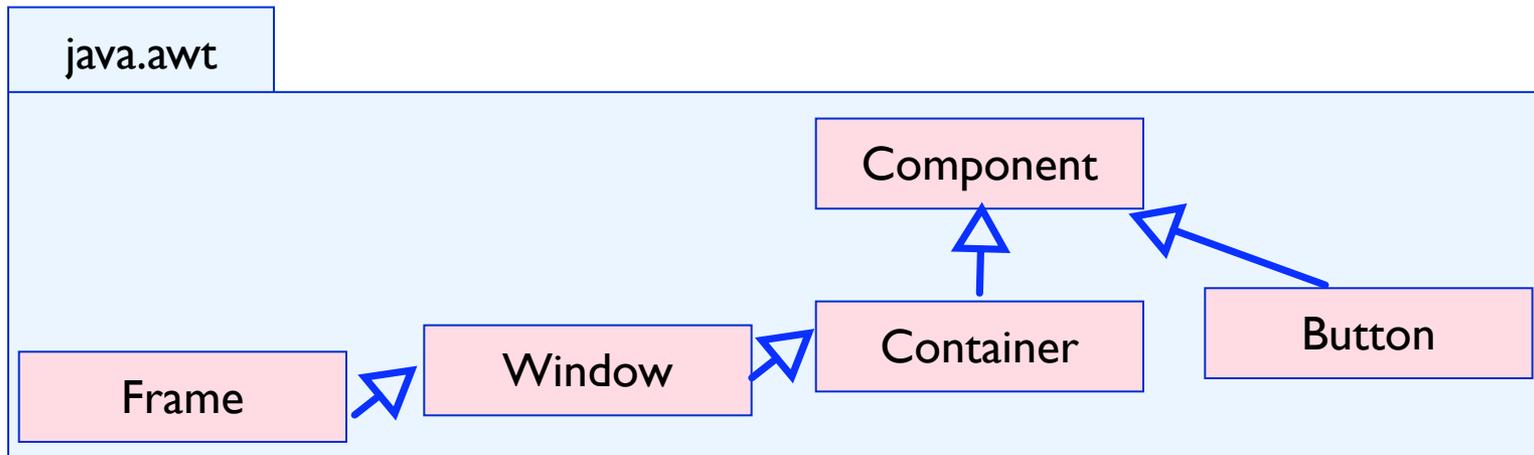
In Swing:

JComponent.java

```
protected String getBorderTitle(Border b) {  
    String s;  
    if (b instanceof TitledBorder) {  
        return ((TitledBorder) b).getTitle();  
    } else if (b instanceof CompoundBorder) {  
        s = getBorderTitle(((CompoundBorder)  
            b).getInsideBorder());  
        if (s == null) {  
            s = getBorderTitle(((CompoundBorder)  
                b).getOutsideBorder());  
        }  
    }  
    return s;  
    } else {  
        return null;  
    }  
}
```

5471 lines of code  
12 minutes

# Presentation of AWT

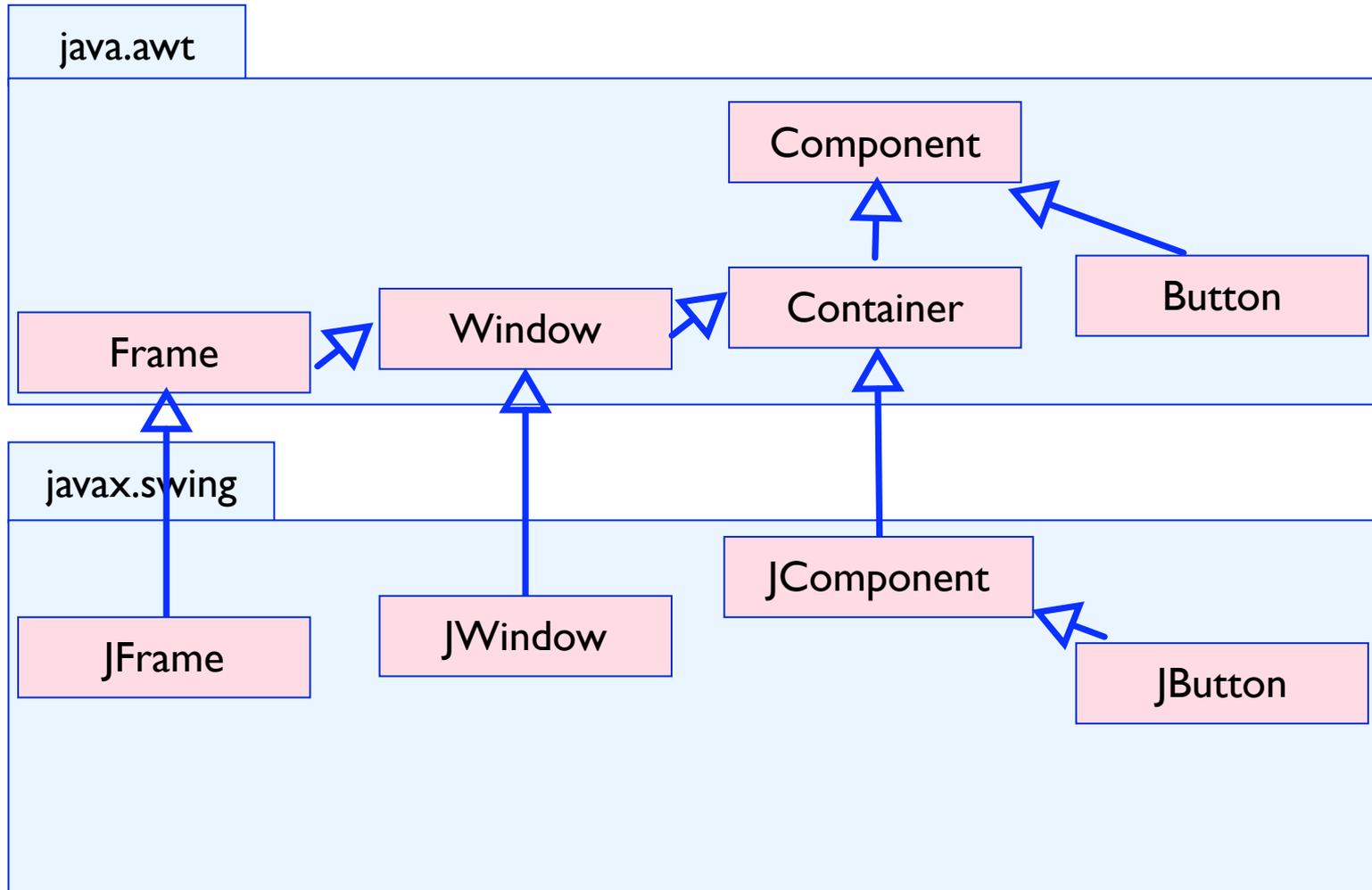


In the AWT framework:

Widgets are components (i.e., inherit from `Component`)

A frame is a window (Frame is a subclass of `Window`)

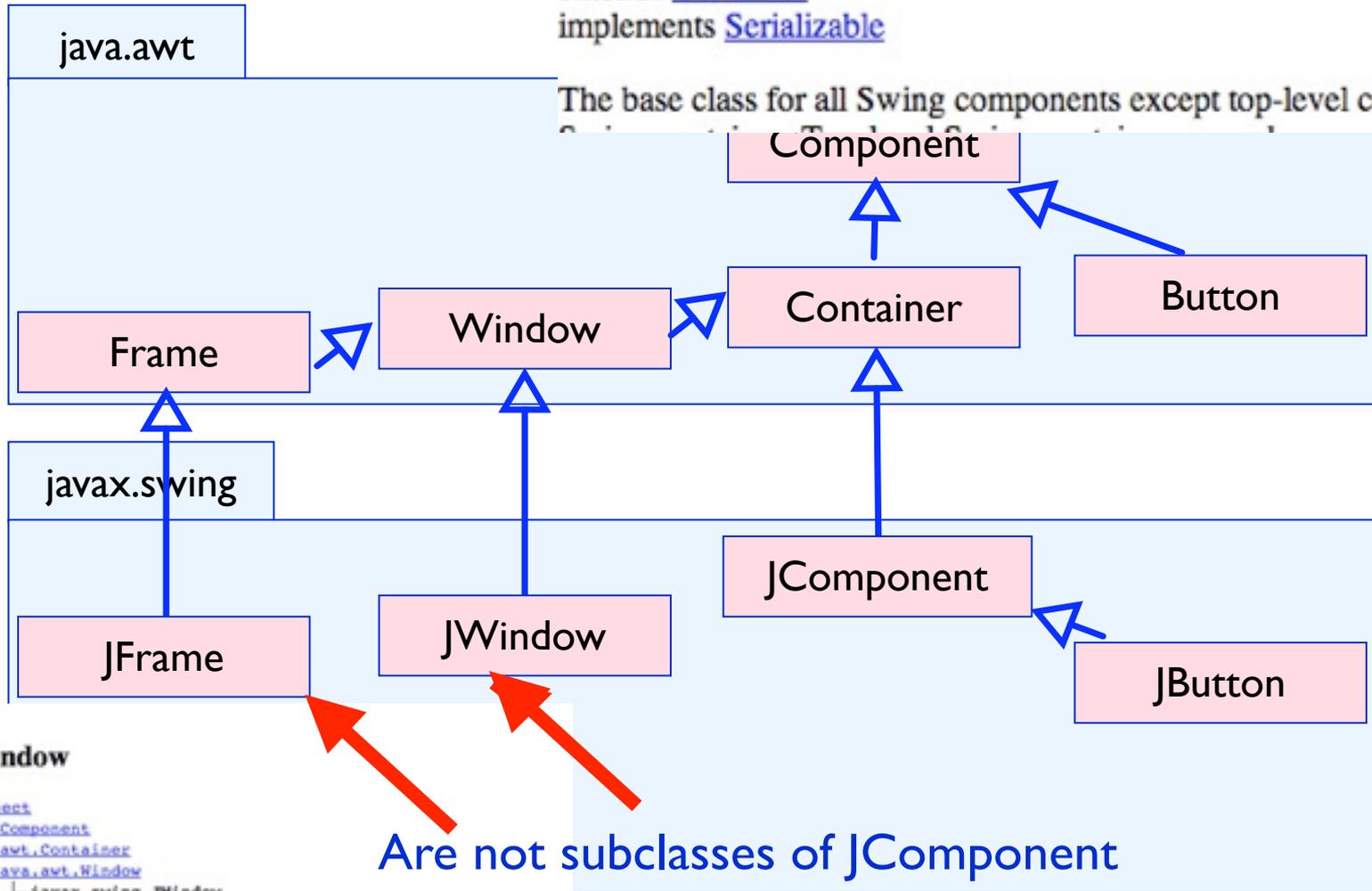
# Swing at the top of AWT



# Problem #1: Brocken Inheritance

```
public abstract class JComponent  
extends Container  
implements Serializable
```

The base class for all Swing components except top-level containers.



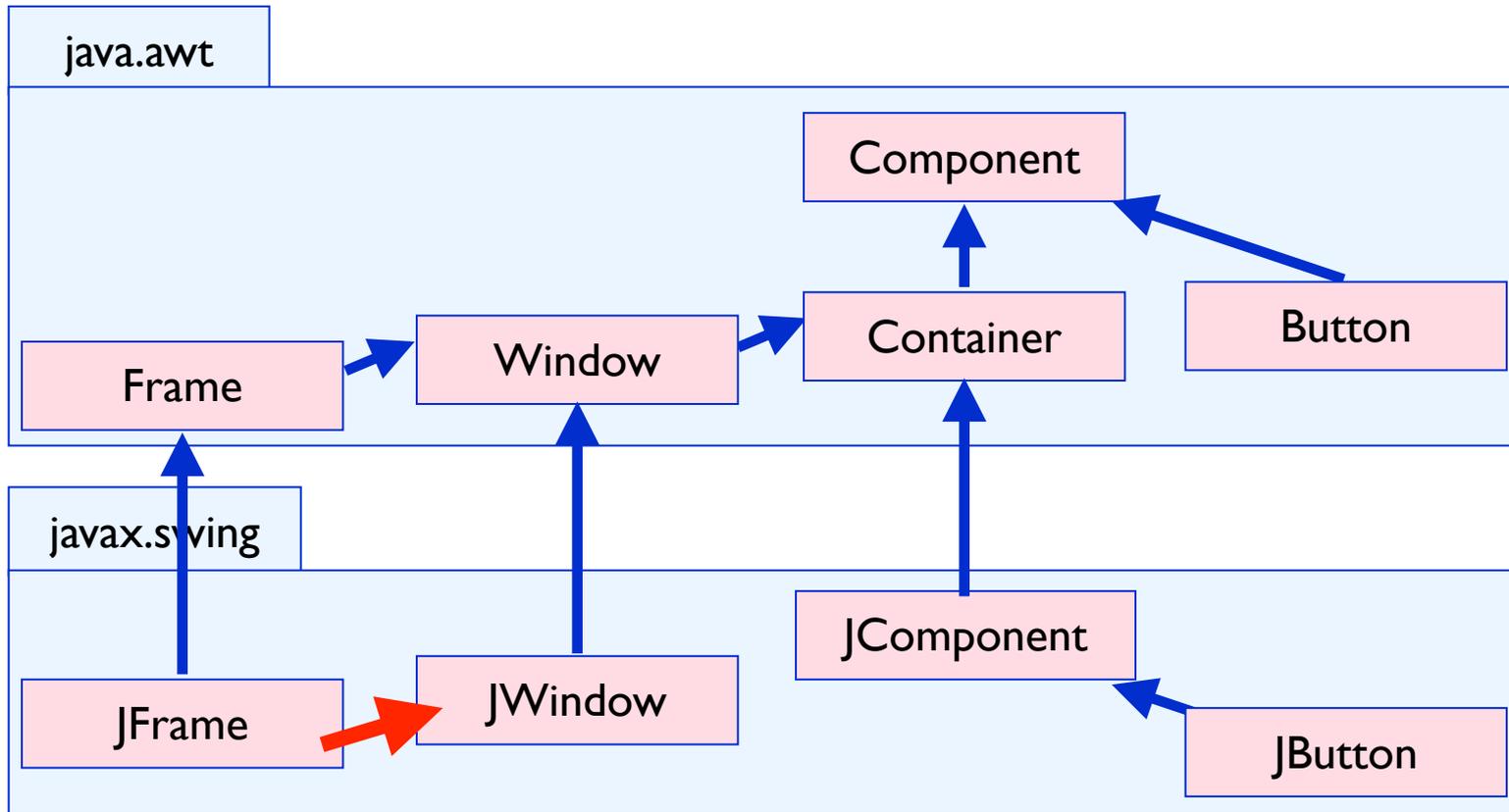
javax.swing  
Class JWindow

```
java.lang.Object  
├─ java.awt.Component  
├─ java.awt.Container  
├─ java.awt.Window  
└─ javax.swing.JWindow
```

All Implemented Interfaces:  
Accessible, ImageObserver, MenuContainer, RootPaneContainer, Serializable

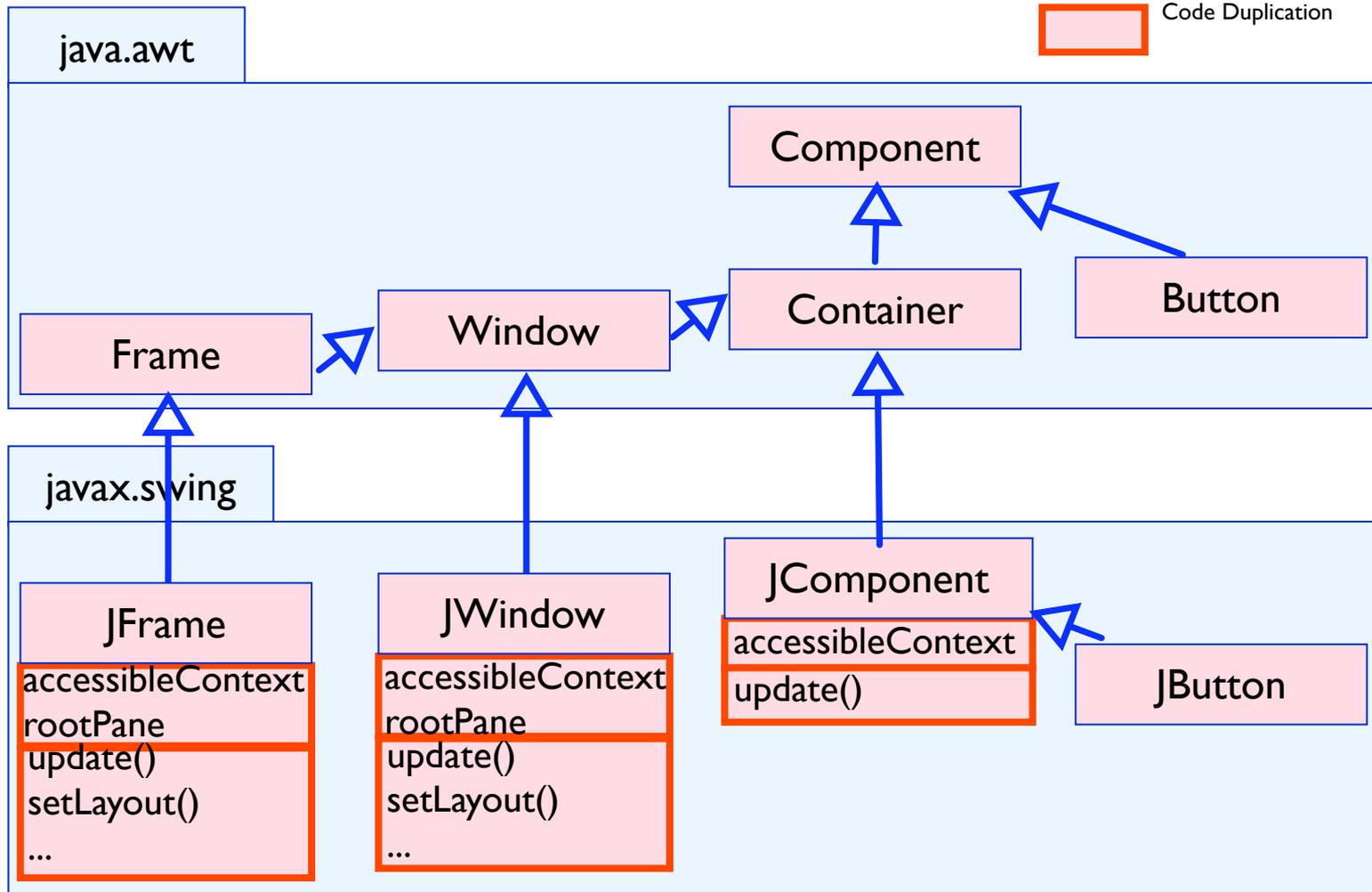
Are not subclasses of JComponent

# Problem #1: Brocken Inheritance



Missing inheritance link between JFrame and JWindow

# Problem #2: Code Duplication



# Problem #3: Explicit Type Checks and Casts

---

```
public class Container extends Component {  
    Component components[] = new Component [0];  
    public Component add (Component comp) {...}  
}
```

```
public class JComponent extends Container {  
    public void paintChildren (Graphics g) {  
        for (; i>=0 ; i--) {  
            Component comp = getComponent (i);  
            isJComponent = (comp instanceof JComponent);  
            ...  
            (JComponent comp).getBounds();  
        }  
    }  
}}
```

# Supporting Unanticipated Changes

---

AWT couldn't *be enhanced* without risk of *breaking* existing code

Swing is, therefore, *built on the top* of AWT using *subclassing*

As a result, *Swing is a big mess internally!*

# Why do we care to have a messy Swing ?

---

Swing appeared in 1998, and *has not evolved since!*

*Swing is too heavy* to be ported to PDA, cellphones, ...

SWT\* is *becoming* a *new standard*.

*Either a system evolves, or it is dead.* [Lehmans74]

\***SWT** is an [open source](#) widget toolkit for Java designed to provide efficient, portable access to the user-interface facilities of the operating systems on which it is implemented.







# Qu'est-ce que la qualité ?

*appréciation globale d'un logiciel,  
basée sur de nombreux indicateurs*

- ☛ ISO/CEI 9126 : **vocabulaire** visant à classer l'ensemble des attributs d'un logiciel relevant de la qualité

ISO/IEC 9126-1,<sup>[1]</sup> classifies [software quality](#) in a structured set of characteristics and sub-characteristics.

# ISO/CEI 9126

- ❧ **Capacité fonctionnelle (*functionality*)**
  - ❧ Répond-il aux besoins de ses utilisateurs ?

# Capacité fonctionnelle

## ☞ Définition

– Ensemble d'attributs portant sur l'existence de fonctions et leurs propriétés; les fonctions sont celles qui satisfont aux besoins exprimés ou implicites

## ➡ Sous-caractéristiques

- **Aptitude** : présence et adéquation d'une série de fonctions pour les tâches données
- **Exactitude** : résultats ou effets justes ou convenus
- **Interopérabilité** : interactions avec d'autres systèmes
- **Sécurité** : accès non autorisé (accidentel ou délibéré) aux programmes et données

# ISO/CEI 9126

- **Capacité fonctionnelle** (*functionality*)
  - Répond-il aux besoins de ses utilisateurs ?

- **Fiabilité** (*reliability*)

- Est-il en mesure d'assurer un niveau de qualité de service suffisant pour satisfaire les besoins opérationnels de ses utilisateurs ?

# Fiabilité



## ☞ Définition

– Ensemble d'attributs portant sur l'aptitude du logiciel à maintenir son niveau de service dans des conditions précises et pendant une période déterminée

## ➡ Sous-caractéristiques

- **Maturité** : fréquence des défaillances dues à des défauts
- **Tolérance aux fautes** : aptitude à maintenir un niveau de service donné en cas de défaut ou d'attaque
- **Possibilité de récupération** : capacité à rétablir son niveau de service et de restaurer les données directement affectées en cas de défaillance ; temps et effort nécessaire pour le faire

# ISO/CEI 9126

- **Capacité fonctionnelle (*functionality*)**
  - Répond-il aux besoins de ses utilisateurs ?
- **Fiabilité (*reliability*)**
  - Est-il en mesure d'assurer un niveau de qualité de service suffisant pour satisfaire les besoins opérationnels de ses utilisateurs ?
- **Maintenabilité (*maintainability*)**
  - Est-il facile d'adapter le logiciel à de nouveaux besoins ou à de nouvelles contraintes ?

# Maintenabilité

## ☞ Définition

– Ensemble d'attributs portant sur l'effort nécessaire pour faire des modifications données

## ➡ Sous-caractéristiques

– **Facilité d'analyse** : effort nécessaire pour diagnostiquer les déficiences et leurs causes ou pour identifier les parties à modifier

– **Facilité de modification** : effort nécessaire pour modifier,

remédier aux défauts ou adapter à l'environnement

– **Stabilité** : risque des effets inattendus des modifications

– **Facilité de test** : effort pour valider le logiciel modifié



# ISO/CEI 9126

❧ Facilité d'utilisation (*usability*)

❧ Peut-il être utilisé à moindre effort ?

# Facilité d'utilisation

## ☞ Définition

– Ensemble d'attributs portant sur l'effort nécessaire pour l'utilisation et l'évaluation individuelle de cette utilisation par un ensemble défini ou implicite d'utilisateurs

## ➔ Sous-caractéristiques

– **Facilité de compréhension** : effort de l'utilisateur pour comprendre la logique et la mise en œuvre

– **Facilité d'apprentissage** : effort de l'utilisateur pour apprendre son utilisation

– **Facilité d'exploitation** : effort que doit faire l'utilisateur pour exploiter et contrôler l'exploitation du logiciel



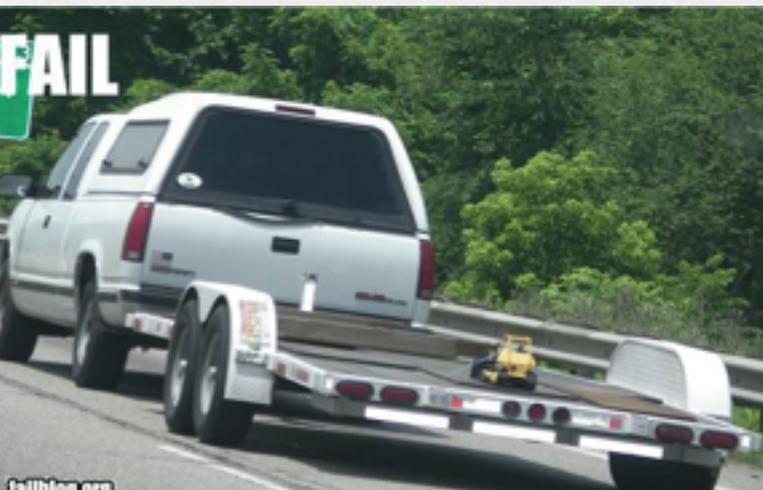
# ISO/CEI 9126

- Facilité d'utilisation (*usability*)

- Peut-il être utilisé à moindre effort ?

- Rendement / Scalabilité (*efficiency*)

- Les ressources matérielles nécessaires à l'exécution du logiciel sont-elles en rapport avec sa rentabilité ?



# Rendement

## ☞ Définition

– Ensemble d'attributs portant sur le rapport existant entre le niveau de service d'un logiciel et la quantité de ressources utilisées, dans des conditions déterminées

## ➡ Sous-caractéristiques

–**Temps** : temps de réponses et de traitement ; débits lors de l'exécution de sa fonction

–**Ressources** : quantité de ressources utilisées ; durée de leur utilisation par fonction

# ISO/CEI 9126

- ❧ Facilité d'utilisation (*usability*)
  - ❧ Peut-il être utilisé à moindre effort ?
- ❧ Rendement / Scalabilité (*efficiency*)
  - ❧ Les ressources matérielles nécessaires à l'exécution du logiciel sont-elles en rapport avec sa rentabilité ?
- ❧ Portabilité (*portability*)
  - ❧ Peut-il être transféré facilement d'une plateforme ou d'un environnement à un autre ?

# Portabilité

## ☞ Définition

– Ensemble d'attributs portant sur l'aptitude du logiciel à être transféré d'un environnement à un autre

## ➡ Sous-caractéristiques

– **Facilité d'adaptation** : possibilité d'adaptation à différents environnements donnés sans que l'on ait recours à d'autres actions ou moyens que ceux prévus à cet effet par le logiciel.

– **Facilité d'installation** : effort nécessaire pour installer le logiciel dans un environnement donné.

– **Conformité aux règles de portabilité** : conformité aux normes et aux conventions ayant trait à la portabilité.

– **Interchangeabilité** : possibilité et effort d'utilisation du logiciel à la place d'un autre logiciel donné dans le même environnement.

# Comment assurer la qualité ?

## → Les tests automatisés

### → *Capacité fonctionnelle e<sup>3</sup> Rendement*

→ Tests fonctionnels

→ Tests de charge

### → *Fiabilité*

→ Tests unitaires

→ Tests d'intégration

### → *Maintenabilité*

→ Tests de non régression

## → Les métriques, les standards de codage, ...

→ *Fiabilité*

→ *Maintenabilité*

## → Le refactoring

→ *Maintenabilité*

## → La méthodologie (voir cours suivants)

### → Normes

→ **CMMi** : Modèle de référence, ensemble structuré de bonnes pratiques, destiné à appréhender, évaluer et améliorer les activités des entreprises d'ingénierie

→ **ISO/CEI 9126**

# On retient au moins

- ❧ Quelles sont les propriétés que je peux/dois vérifier lorsque je fournis un logiciel?
  - ❧ Ne pas répondre à cette question, revient à faire un gâteau sans se poser les questions :
    - ❧ Est-il assez/trop sucré, y-en a-t-il assez pour tous, va-t-il résister au transport?
  - ❧ Ne pas répondre à la question dans les rendus de TD/TP/exam au moins...

