

IFT 2255 – Génie logiciel

Implémentation

Une bonne conception...

... contribue à la moitié de l'effort d'implémentation !

- **Rigueur**
 - S'assure que toutes les exigences sont satisfaites
- **Séparation des préoccupations**
 - **Modularité**
 - Permet le travail isolé et parallèle: composants sont indépendants des autres
 - **Abstraction**
 - Permet le travail isolé et l'intégration: interfaces garantissent que les composants vont fonctionner ensemble
- **Anticipation du changement**
 - Permet d'absorber les changements sans effort
- **Généralisation**
 - Permet de réutiliser les composants à travers le système et d'autres systèmes
- **Incrémentalité**
 - Permet de développer le logiciel avec des résultats intermédiaires

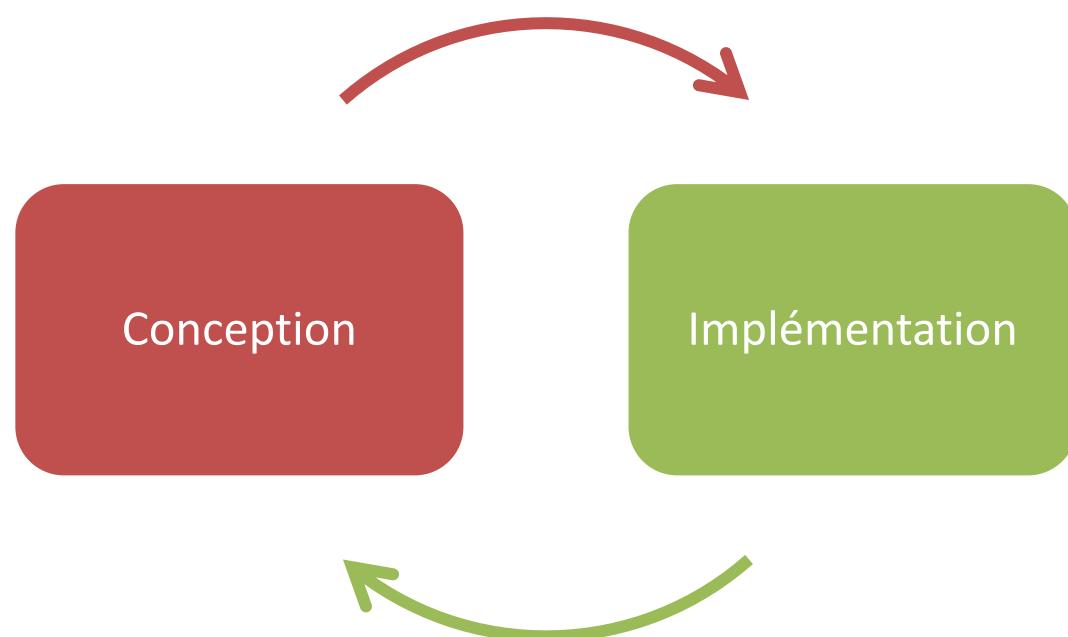
Une mauvaise conception...

... ne sera jamais implémentée !

- Manque de rigueur \Rightarrow oubli de fonctionnalités
- Manque de modularité \Rightarrow conflits entre développeurs ou travail en double
- Manque d'anticipation au changement \Rightarrow re-conception et ré-implémentation
- Manque de généralité \Rightarrow gonflement du code: allongement, ralentissement, duplication, gaspillage de ressources
- Manque d'incrémentalité \Rightarrow mauvaise estimation et évaluation du logiciel

De la conception au code

- Implémentation: processus de traduire la conception détaillée en code exécutable par une machine
- Interaction conception-implémentation
 - **L'implémentation et la conception doivent rester cohérents**



Workflow d'implémentation

- But est d'implémenter le produit logiciel cible
- Un grand système est partitionné en **sous-systèmes**
 - Implémentés en **parallèle** par des équipes de développeurs
- Sous-systèmes consistent en des **composants** ou des **artéfacts de code**
- Quand un programmeur a complété l'implémentation d'un artéfact, il le **test**
- Puis, le module est envoyé à l'équipe d'**assurance qualité** pour le tester plus en profondeur
 - Tester fait partie de l'implémentation: **l'implémentation n'est pas complète tant qu'elle n'est pas testée**

Tâches d'implémentation

- Choisir le langage de programmation le plus approprié
- Établir les normes de programmation
- Répartir l'effort de travail
- Implémenter = coder + tester
- Intégrer

Répartition de l'effort

- Les produits logiciel sont en général trop grand pour être implémentés par un seul programmeur
- Assigner différents modules à différents développeurs
 - Assiguation peut être **incrémentale**
 - Gérer les **changements** d'assiguation
 - Maladie, démission, recrue, ajustement de temps, programmeurs étoiles
- Le développement **d'interfaces** est crucial pour la programmation distribuée
 - Ils forment des **contrats** entre les modules

Choix du langage de programmation le plus approprié

Choix du langage de programmation

- Le langage est généralement spécifié dans le contrat
- Mais le contrat peut stipuler que le logiciel doit être implémenté dans le langage *le plus approprié*
- Quel langage choisir ?

Générations des langages de programmation

- **Langages de 1^{ère} Génération**

- Langage machine

```
110110001110100010010001001010010100101001001010
```

```
LOAD [26], R1
LOAD 3, R2
ADD R1, R2, R3
STORE R3, [1700000029]
```

- **Langages de 2^e Génération**

- Assembleur spécifique à une machine

- **Langages de 3^e Génération**

- Langage de haut niveau compilé vers code machine

```
ATM atm = new ATM(10);
atm.deposit(100);
atm.withdraw(20);
System.out.print(atm.getBalance());
```

- **Langages de 4^e Génération**

- Base de données, Visual Basic, Formulaires

```
for every surveyor
  if rating is excellent
    add 6500 to salary
```

- **Langages spécifiques au domaine**

- Décrits en utilisant les concept du domaine, pas le code
 - Code est automatiquement généré à partir des modèles

```
7 inputAlphabet "01"
8 outputAlphabet "oe"
9
10 start state Even
11 transition '0' / 'o' -> Odd
12 transition '1' / 'e' -> Even
13 tran
14
15 state Odd
16 transition '0' / 'e' -> Even
17 transition '1' / 'o' -> Odd
```

Chronologie des langages de programmation

Premiers langages de programmation

modernes apparaissent dans les 1950

- **1951 - Regional Assembly Language**

- 1952 - Autocode

- **1954 - FORTRAN**

- 1954 - IPL

- 1955 - FLOW-MATIC

- 1957 - COMTRAN

- 1958 - LISP

- 1958 - ALGOL

- 1959 - FACT

- **1959 - COBOL**

- **1962 - Simula**

- 1962 - SNOBOL

- 1963 - CPL

- **1964 - BASIC**

- 1964 - PL/I

Paradigmes fondamentaux

- 1968 - Logo

- **1970 - Pascal**

- 1970 - Forth

- **1972 - C**

- **1972 - Smalltalk**

- **1972 - Prolog**

- **1973 - ML**

- 1975 - Scheme

- **1978 - SQL**

Grande échelle et performance

- **1980 - C++**

- 1983 - Objective-C

- **1983 - Ada**

- 1984 - Common Lisp

- **1985 - Eiffel**

- 1986 - Erlang

- **1987 - Perl**

- 1988 - Tcl

- 1989 - FL

L'ère de l'internet

- 1990 - Haskell

- **1991 - Python**

- 1991 - Visual Basic

- **1993 - Ruby**

- 1993 - R

- 1993 - Lua

- **1995 - Java**

- 1995 - Delphi

- **1995 - JavaScript**

- **1995 - PHP**

- 1997 - Rebol

- 1999 - D

Plus récemment

- **2001 - C#**

- 2001 - Visual Basic .NET

- 2002 - F#

- 2003 - Scala

- 2003 - Factor

- 2007 - Clojure

- 2007 - Groovy

- **2009 - Go**

- 2011 - Dart

- **2014 - Swift**

Et ce n'est qu'une liste partielle...

https://en.wikipedia.org/wiki/History_of_programming_languages

FORTRAN

- « FORmula TRANslator », inventé par John Backus et IBM
- Impact révolutionnaire en informatique
 - Premier langage de haut niveau qui est exécutable
- Utilisé principalement pour le **calcul scientifique**
- D'excellents **compilateurs** existent encore aujourd'hui
 - Dernière version FORTRAN 2015

```
PROGRAM DEGRAD
! Déclaration des variables
INTEGER DEG
REAL RAD, COEFF
!
! En-tête de programme
WRITE (*, 10)
10 FORMAT (' ',20('*') /
&          ' * Degres * Radians *' /
&          ' ', 20('*') )
!
! Corps de programme
COEFF = (2.0 * 3.1416) / 360.0
DO DEG = 0, 90
    RAD = DEG * COEFF
    WRITE (*, 20) DEG, RAD
20 FORMAT (' * ',I4,' * ',F7.5,' * ')
END DO
!
! Fin du tableau
WRITE (*, 30)
30 FORMAT (' ',20('*') )
!
! Fin de programme
STOP
END PROGRAM DEGRAD
```

Lisp

- Langage **fonctionnel** développé par McCarthy comme implémentation du calcul lambda de Church
 - Fonctions sont des éléments de premier ordre (comme Object)
- Très puissant pour les calculs symboliques, beaucoup d'applications en **intelligence artificielle**
- A influencé plusieurs langages fonctionnels (ML et Haskell)

```
(defun factorial (n)
  (if (<= n 1)
      1
      (* n (factorial (- n 1)))))
```

COBOL

- **COmmon Business-Oriented Language** : Un langage commun pour la gestion de l'administration
- Très puissant pour les **calculs numériques**: le must en matière de gestion et de manipulation précise
- *Le plus répandu des langages (développé par et pour le DoD)*

```
IDENTIFICATION DIVISION.  
PROGRAM-ID. HELLO-WORLD.  
*  
ENVIRONMENT DIVISION.  
*  
DATA DIVISION.  
*  
PROCEDURE DIVISION.  
PARA-1.  
DISPLAY "Hello, world.".  
*  
EXIT PROGRAM.  
END PROGRAM HELLO-WORLD.
```

BASIC

- « Beginner's All-purpose Symbolic Instruction Code »
- Permettre aux **non scientifiques** de programmer
- Famille de langages à usage **interactif** interprétés

```
INPUT "Quel est votre nom"; UserName$  
PRINT "Bonjour "; UserName$  
DO  
    INPUT "Combien d'étoiles voulez-vous"; NumStars  
    Stars$ = ""  
    Stars$ = REPEAT$( "*", NumStars) '-ANSI BASIC  
    'Stars$ = STRING$(NumStars, "*") '-MS BASIC  
    PRINT Stars$  
    DO  
        INPUT "Voulez-vous plus d'étoiles"; Answer$  
        LOOP UNTIL Answer$ <> ""  
    LOOP WHILE UCASE$(LEFT$(Answer$, 1)) = "O"  
    PRINT "Au revoir ";  
    FOR A = 1 TO 200  
        PRINT UserName$; " ";  
    NEXT A  
    PRINT
```

Ada

- À l'origine devait être le langage standard de tous les programmes soumissionnés par DoD
- Un des langages les plus **fiables** (gestion d'exceptions)
- Supporte la **programmation concurrentielle** nativement
- Supporte la programmation orientée-objet

```
procedure Traffic is
type Airplane_ID is range 1..10; -- 10 air
task type Airplane (ID: Airplane_ID); -- task r
type Airplane_Access is access Airplane; -- refere
protected type Runway is -- the sh
    entry Assign_Aircraft (ID: Airplane_ID); -- all en
private -- protec
    Clear: Boolean := True;
end Runway;
task type Controller (My_Runway: Runway_Access) is -- task entries for synchronous message passing
    entry Request_Takeoff (ID: in Airplane_ID; Takeoff);
    entry Request_Approach(ID: in Airplane_ID; Approach);
end Controller;
task body Controller is
begin
    loop
        My_Runway.Wait_For_Clear; -- wait until runway i
        select -- wait for two types
            when Request_Approach'count = 0 => -- guard st
                accept Request_Takeoff (ID: in Airplane_ID; Ta
            do -- start o
                My_Runway.Assign_Aircraft (ID); -- reserve
                Takeoff := My_Runway; -- assign '
                end Request_Takeoff; -- end of t
            or
                accept Request_Approach (ID: in Airplane_ID; Ap
                    My_Runway.Assign_Aircraft (ID);
                    Approach := My_Runway;
                end Request_Approach;
            or -- terminate if no tas
                terminate;
            end select;
        end loop;
    end;
```

Smalltalk

- Un des premiers langages **orienté-objet pur** implémenté complètement
- Premier langage à avoir une IDE avec **interface utilisateur graphique**



```
Object subclass: #MessagePublisher
instanceVariableNames: ''
classVariableNames: ''
poolDictionaries: ''
category: 'Smalltalk Examples'
publisher := MessagePublisher new
quadMultiply: i1 and: i2
| mul |
mul := i1 * i2.
^mul * 4
```

Prolog

- Langage de **programmation logique** le plus utilisé
 - Démonstrateur de théorèmes, systèmes experts, traitement de langage naturel
- **Règles déclaratives** de logique: spécifient ce qu'on veut faire, pas comment le faire

```
mother_child(trude, sally).  
  
father_child(tom, sally).  
father_child(tom, erica).  
father_child(mike, tom).  
  
sibling(X, Y)      :- parent_child(Z, X),  
parent_child(Z, Y).  
  
parent_child(X, Y) :- father_child(X, Y).  
parent_child(X, Y) :- mother_child(X, Y).
```

```
?- sibling(sally, erica).  
Yes
```

C

- Langage de **programmation système**
 - Systèmes d'exploitation, superordinateurs, systèmes embarqués
- Le compilateur de plusieurs langages sont implémentés en C (ex: Python, PHP)
- Un des langages les plus efficaces (optimisés) et des plus utilisés encore aujourd'hui

```
#include <stdio.h>
#include <time.h>

void getSeconds(unsigned long *par);

int main () {
    unsigned long sec;
    getSeconds( &sec );
    /* print the actual value */
    printf("Number of seconds: %ld\n", sec );
    return 0;
}

void getSeconds(unsigned long *par) {
    /* get the current number of seconds */
    *par = time( NULL );
    return;
}
```

C++

- Langage moderne orienté-objet qui permet aussi la manipulation de mémoire à bas niveau: basé sur un sous-ensemble de C
- Polymorphisme, héritage multiple, Standard Template Library (collections et itérateurs génériques), pointeurs, opérations pour le garbage collection

```
#include<iostream>

int main()
{
    using std::cout;
    cout << "Hello, new world!" // standard output
        << std::endl;           // new line
}

void foo()
{
    std::cout << "Hello, new world!"
        << std::endl;
}
```

```
// messageinternet.hpp
#include<string>

class MessageInternet
{
private:
    const std::string m_sujet, m_expediteur, m_destinataire;
    attributs

public:
    MessageInternet(
        const std::string& sujet,
        const std::string& expediteur,
        const std::string& destinataire); // constructeur
    ~MessageInternet(); // destructeur
    const std::string& get_sujet() const; // méthode
    const std::string& get_expediteur() const; /
    const std::string& get_destinataire() const;
};
```

Java

- Développé par Sun Microsystems, maintenant propriété d'Oracle
- Simplification de C++ pour ne supporter que la programmation orientée-objet
- Portable et indépendant de la machine grâce à une machine virtuelle (JVM)

```
package fibsandlies;
import java.util.HashMap;

/**
 * This is an example of a Javadoc comment; Javadoc can compile documentation
 * from this text. Javadoc comments must immediately precede the class, method, or field
 */
public class FibCalculator extends Fibonacci implements Calculator {
    private static Map<Integer, Integer> memoized = new HashMap<Integer, Integer>();
    /**
     * The main method written as follows is used by the JVM as a starting point for
     */
    public static void main(String[] args) {
        memoized.put(1, 1);
        memoized.put(2, 1);
        System.out.println(fibonacci(12));
    }

    /**
     * Given a non-negative number FIBINDEX, returns
     * the Nth Fibonacci number, where N equals FIBINDEX.
     * @param fibIndex The index of the Fibonacci number
     * @return The Fibonacci number
     */
    public static int fibonacci(int fibIndex) {
        if (memoized.containsKey(fibIndex)) {
            return memoized.get(fibIndex);
        } else {
            int answer = fibonacci(fibIndex - 1) + fibonacci(fibIndex - 2);
            memoized.put(fibIndex, answer);
            return answer;
        }
    }
}
```

There were five primary goals in the creation of the Java language:

1. It must be "simple, object-oriented, and familiar".
2. It must be "robust and secure".
3. It must be "architecture-neutral and portable".
4. It must execute with "high performance".
5. It must be "interpreted, threaded, and dynamic".

C#

- Similaire à Java et C++ mais pour la plateforme Microsoft .NET
- Le *common language infrastructure* gère les objets qui peuvent être partagés entre **différents langages et paradigmes** de programmation (Visual Basic, F#)

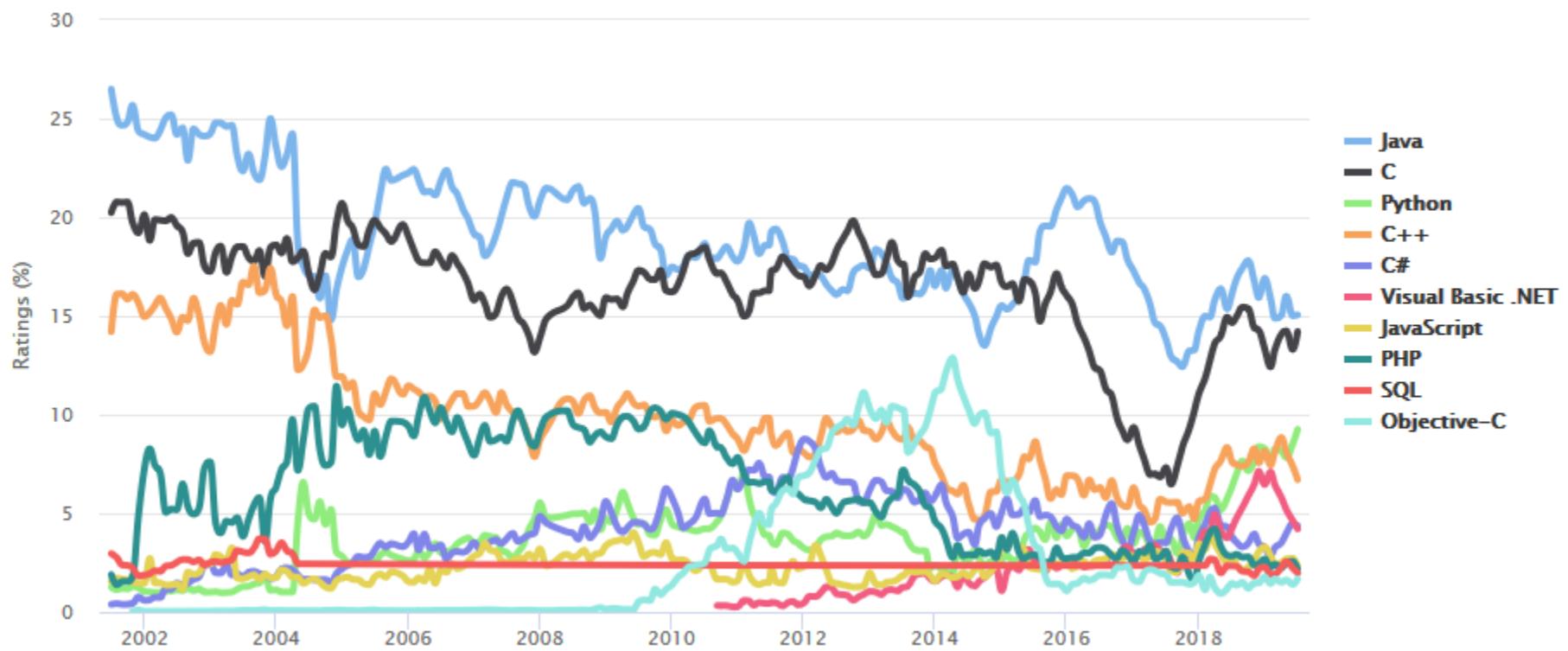
```
using System;
using System.Linq;

class Program {
    static void Main()
    {
        string[] words = {"hello", "wonderful", "LINQ", "beautiful", "world"};
        //Get only short words
        var shortWords = from word in words
                        where word.Length <= 5
                        select word;

        //Print each word out
        foreach (var word in shortWords) {
            Console.WriteLine(word);
        }
        Console.ReadLine();
    }
}
```

Estimé de la popularité des langages

Basé sur les recherches populaires sur le web



<http://www.tiobe.com/tiobe-index/>

Pourquoi y a-t-il autant de langages ?

- **Évolution empirique**
 - Qu'est-ce qui fait qu'un concept de programmation est bon ou pas?
 - Début des 1970: programmation structurée où les GOTO ont été remplacés par des expressions WHILE et CASE
 - Fin des 1980: imbrication de code a donné naissance aux structures orientées-objet
- **Besoins spécifiques**
 - Beaucoup de langages ont été conçus pour un domaine spécifique
 - Science, statistiques, affaires, intelligence artificielle, internet, ...
- **Préférence personnelle – expérience industrielle**
 - La force et variété des préférences de chacun fait en sorte qu'il n'y aura jamais un langage de programmation universel
 - C'est une bonne chose pour la diversité pour des fins spécifiques
 - Diversité des conceptions, des expressions, des opinions

Bonnes pratiques de programmation

Différents styles de programmation

Étudiant: ça fonctionne

```
public int fibonacci(int x) {
    if (x == 1) {
        return 1;
    } else if (x == 2) {
        return 1;
    } else {
        return fibonacci(x - 1) + fibonacci(x - 2);
    }
}
```

Optimisation algorithmique au détriment de la compréhension

```
public int getFibonacciNumber(int n) {
    return (int) divide(subtract(exponentiate(phi(), n), exponentiate(psi(), n)),
                        subtract(phi(), psi())));
}

public double exponentiate(double a, double b) {
    if (equal(b, zero())) {
        return one();
    } else {
        return multiply(a, exponentiate(a, subtract(b, one())));
    }
}

public double phi() {
    return divide(add(one(), sqrt(add(one(), one(), one(), one(), one()))),
                  add(one(), one())));
}

public double psi() {
    return subtract(one(), phi());
}
```

Démonstration: juste pour la démo

```
public int getFibonacciNumber(int n) {
    switch(n) {
        case 1: return 1;
        case 2: return 1;
        case 3: return 2;
        case 4: return 3;
        case 5: return 5;
        case 6: return 8;
        case 7: return 13;
        default:
            // good enough for the demo, lol
            return -1;
    }
}
```

Différents styles de programmation

Startup: vers la perfection, sans l'atteindre

```
// TODO add Javadoc comments
/**
 * getFibonacciNumber
 */
// TODO Should we move this to a different file?
public int getFibonacciNumber(int n) {
    // TODO Stack may overflow with recursive implementation, switch
    // iteration approach at some point?
    if (n < 0) {
        // TODO This should probably throw an exception. Or maybe just
        // a log message?
        return -1;
    } else if (n == 0) {
        // TODO Generalize the initial conditions?
        return 0;
    } else if (n == 1) {
        return 1;
    } else {
        // TODO Spend some time with my family and kids, I've been a
        // over 48 hours straight.
        return getFibonacciNumber(n - 1) + getFibonacciNumber(n - 2);
    }
    public static final int UNITE = 1;
    public static final int UNITED = 2;

    // meowwww meow
    public int meow(int KITTENS_OF_THE_WORLD) {
        // MEOW
        if (KITTENS_OF_THE_WORLD < UNITED) {
            return KITTENS_OF_THE_WORLD;
        } else {
            // meeowwwwwww
            // meooooowwwwwwwwwwwwwwwwwwwwwww
            return meow(KITTENS_OF_THE_WORLD - UNITE)
                + meow(KITTENS_OF_THE_WORLD - UNITED);
        }
    }
}
```

Grande compagnie logiciel: normes, fiabilité, réutilisation

```
/**
 * getFibonacciNumber is a method that, given some index n, returns the nth
 * Fibonacci number.
 * @param n The index of the Fibonacci number you wish to retrieve.
 * @return The nth Fibonacci number.
 */
public CustomInteger64 getFibonacciNumber(CustomInteger64 n) {
    FibonacciDataViewBuilder builder =
        FibonacciDataViewBuilderFactory.createFibonacciDataViewBuilder(
            new FibonacciDataViewBuilderParams(n, null, null, 0, null));
    if (builder == FibonacciDataViewBuilderConstants.ERROR_STATE) {
        throw new FibonacciDataViewBuilderException();
    }
    FibonacciDataView dataView = builder.GenerateFibonacciDataView(this);
    if (dataView == FibonacciDataViewConstants.ERROR_STATE) {
        throw new FibonacciDataViewGenerationException();
    }
    return dataView.accessNextFibonacciNumber(null, null, null);
```

Fonctionne sans être compréhensible par un humain

Bonne pratiques de programmation

- Utilisez des noms **significatifs** et de façon **cohérente** pour faciliter la compréhension du code
 - Aider les programmeurs de **maintenance future**
 - **Variables** : substantifs représentant la signification ou l'unité
 - double money;
 - vs.
 - double salary;
 - **Méthodes** : verbes à l'infinitif représentant l'action ou le retour
 - getName(), update(), isDirty(), hasChildren()
 - **Classes** : titre substantif représentant le but de l'abstraction
 - Graph, Delivery, RentalHistory, Employee
 - **Paquets** : URI représentant la hiérarchie et la fonctionnalité
 - org.applicationname.gui, com.companyname.productname.tier

Noms de variables significatifs

- Supposons qu'un artefact de code contient les noms de variables suivants: freqAverage, frequencyMaximum, minFr, frqncyTotl
- Le programmeur de maintenance doit savoir si freq, frequency, fr, frqncy se réfèrent toutes à la même chose
- Si oui, utilisez le même mot, préférablement frequency, peut-être freq ou aussi frqncy, mais pas fr
- Sinon, utilisez un mot différent (ex: rate) pour une quantité différente

Noms de variables cohérents

- On peut utiliser `frequencyAverage`, `frequencyMaximum`,
`frequencyMinimum`, `frequencyTotal`
- On peut aussi utiliser `averageFrequency`, `maxFrequency`,
`minFrequency`, `totalFrequency`
- À condition que les 4 noms proviennent du **même ensemble lexical**

Noms à éviter

- Caractères faciles à confondre
 - 1 | L
 - o O 0
 - S 5
 - G 6
- Noms trompeurs ou génériques ayant plusieurs sens
 - money, date, bouton
- Synonymes
 - average/mean
- Utiliser les majuscules de façon cohérente et logique
 - **HelloWorld**, helloworld, Helloword, **helloworld**, HeLloWoRlD, HELLOWORLD

Nomenclature: Java & CamelCase

Identifier type	Rules for naming	Examples
Classes	Class names should be nouns in Upper CamelCase , with the first letter of every word capitalised. Use whole words — avoid acronyms and abbreviations (unless the abbreviation is much more widely used than the long form, such as URL or HTML).	<ul style="list-style-type: none"> • class Raster; • class ImageSprite;
Methods	Methods should be verbs in lower CamelCase or a multi-word name that begins with a verb in lowercase; that is, with the first letter lowercase and the first letters of subsequent words in uppercase.	<ul style="list-style-type: none"> • run(); • runFast(); • getBackground();
Variables	Local variables, instance variables, and class variables are also written in lower CamelCase . Variable names should not start with underscore (_) or dollar sign (\$) characters, even though both are allowed. This is in contrast to other coding conventions that state that underscores should be used to prefix all instance variables. Variable names should be short yet meaningful. The choice of a variable name should be mnemonic — that is, designed to indicate to the casual observer the intent of its use. One-character variable names should be avoided except for temporary "throwaway" variables. Common names for temporary variables are i, j, k, m, and n for integers; c, d, and e for characters.	<ul style="list-style-type: none"> • int i; • char c; • float myWidth;
Constants	Constants should be written in uppercase characters separated by underscores. Constant names may also contain digits if appropriate, but not as the first character.	<ul style="list-style-type: none"> • static final int MAX_PARTICIPANTS = 10;

Disposition du code

- Espaces, lignes blanches
 - Votre code c'est comme un texte technique: séparé en paragraphes, par idées
- Regroupement
 - Méthodes (ou variables) qui collaborent sont rapprochées
 - Grouper par visibilité
- Alignement, indentation, aération
 - Logique complexe: conditions, boucles imbriquées, contenant/contenu
- Parenthèses, accolades
 - Cohérence, mettre en valeur

Obfuscation de code

- Rend le code impénétrable, illisible et incompréhensible
- Utile pour cacher le code remis ou pour l'alléger
 - Ex: JavaScript
- Peut facilement être retranscrit automatiquement par rétro-ingénierie

```
m(f,a,s)char*s;
{char c;return f&1?a!=*s++?m(f,a,s):s[11]:f&2?a!=*s++?1+m(f,a,s):1:f&4?a--?
putchar(*s),m(f,a,s):a:f&8?*s?m(8,32,(c=m(1,*s++, "Arjan Kenter. \no$.../.\""),m(4,m(2,*s++, "POCnWAUvBVxRsoqatKJurgXYyDQbzhLwkNjdMTGeIScHFmplizef"),&c),s)):65:(m(8,34,"rgeQjPruaOnDaPeWrAaPnPcNOrPaPnPjPrCaPrPnPpPaOrvaPndeOrAnOrPnP\lnOaPnPjPaOrPnPnPnPtPnPnPraAaPnPBrnnnsrnnBaPeOrCnPOnCaPnPjPtPnPaaPnPnPnPcPn\BrAnxrAnVePrCnBjPrOnvrCnxrAnxrAnsOnvjPrOnUrOnornnnsrnnorOtCnCjPrCtPnPcnnirWtP\nCjPrCaPnOtPrCnErAnOjPrOnvtPnnrCnNrnnRePjPrPtnrUnnrntPnbtPrAaPnPcnnOrPjPrRtPn\CaPrWtCnKtPnOtPrBnCjPrOnCaPrVtPnOtOnAtnrxapnCjPrqnnaPrtorsaPnPcTpjPratPnnaPrA\apnPnPnnaPrvaPnnjPrKtPnPnWaOrWtOnnaPnPnWaPrCaPnntOjPrrtOnWanrOtPnPcPnBtCjPrYtOn\UaOrPnPvjPrwttnxjPrMnBjPrTnUjP"),0);}

main(){return m(0,75,"mIWltouQJGsBniKYvTxODAfbcUcFzSpMwNCHEgrdLaPkyVRjXeqZh");}
```

Commentaire prologue

- Au tout **début de chaque artefact** de code (classe ou méthode) on explique chaque variable dans le prologue
- Les autres programmeurs et ceux de maintenance pourront rapidement **comprendre ce que chaque variable représente**
 - Auteur(s)
 - Dates de création et modification(s)
 - Explications
 - Dépendances
 - License

The name of the code artifact
A brief description of what the code artifact does
The programmer's name
The date the code artifact was coded
The date the code artifact was approved
The name of the person who approved the code artifact
The arguments of the code artifact
A list of the name of each variable of the code artifact, preferably in alphabetical order, and a brief description of its use
The names of any files accessed by this code artifact
The names of any files changed by this code artifact
Input-output, if any
Error-handling capabilities
The name of the file containing test data (to be used later for regression testing)
A list of each modification made to the code artifact, the date the modification was made, and who approved the modification
Any known faults

Autres commentaires

- Insérer des commentaire en ligne (//) pour aider les programmeurs à **comprendre ce que le code fait**
- Les commentaires sont essentiels quand le code est **écrit d'une manière non-évidente**, utilise des **aspects non-communs du langages**, ou dépend d'une **logique complexe**
- Si le code est trop embrouillé, mélangeant, perturbant, il faut le recoder plus clairement
 - **Ne jamais promouvoir ou excuser une programmation médiocre**

QUESTION

Doit-on rédiger les commentaires avant, après ou pendant qu'on code ?

Les trois

- Avant: expliquer l'idée, la stratégie complexe
- Pendant: pour ne pas oublier, TODO, clarifications
- Après: documentation

Commentaires de documentation

- Permettent de générer automatiquement la documentation de l'API du programme
- javadoc produit un fichier HTML (voir démo)

```

import java.io.*;
/**
 * <h1>Add Two Numbers!</h1>
 * The AddNum program implements an application that
 * simply adds two given integer numbers and Prints
 * the output on the screen.
 * <p>
 * <b>Note:</b> Giving proper comments in your program makes it more
 * user friendly and it is assumed as a high quality code.
 * @author Zara Ali
 * @version 1.0
 * @since 2014-03-31
 */
public class AddNum {
    /**
     * This method is used to add two integers. This is
     * a the simplest form of a class method, just to
     * show the usage of various javadoc Tags.
     * @param numA This is the first parameter to addNum method
     * @param numB This is the second parameter to addNum method
     * @return int This returns sum of numA and numB.
     */
    public int addNum(int numA, int numB) {
        return numA + numB;
    }
    /**
     * This is the main method which makes use of addNum method.
     * @param args Unused.
     * @return Nothing.
     * @exception IOException On input error.
     * @see IOException
     */
    public static void main(String args[]) throws IOException {
        AddNum obj = new AddNum();
        int sum = obj.addNum(10, 20);
        System.out.println("Sum of 10 and 20 is :" + sum);
    }
}

```

The screenshot shows the JavaDoc generated HTML for the `AddNum` class. The page has a header with tabs for "All Classes", "Package", "Class", "Tree", "Deprecated", "Index", and "Help". Below the tabs, there are links for "Prev Class", "Next Class", "Frames", "No Frames", "Summary", "Nested", "Field", "Const", "Method", "Detail", "Field", "Const", and "Method". A search bar is also present.

Class AddNum

`java.lang.Object`
AddNum

Add Two Numbers!

The AddNum program implements an application that simply adds two given integer numbers and Prints the output on the screen. Note:Giving proper comments in your program makes it more user friendly and it is assumed as a high quality code.

Since: 2014-03-31

Constructor Summary

Constructors

Constructor and Description
<code>AddNum()</code>

Method Summary

Methods

Modifier and Type	Method and Description
int	<code>addNum(int numA, int numB)</code> This method is used to add two integers.
static void	<code>main(java.lang.String[] args)</code> This is the main method which makes use of addNum method.

Methods inherited from class `java.lang.Object`

<code>clone</code> , <code>equals</code> , <code>finalize</code> , <code>getClass</code> , <code>hashCode</code> , <code>notify</code> , <code>notifyAll</code> , <code>toString</code> , <code>wait</code> , <code>wait</code> , <code>wait</code>

Constructor Detail

AddNum

<code>public AddNum()</code>

Method Detail

addNum

<code>public int addNum(int numA, int numB)</code>
--

This method is used to add two integers. This is a the simplest form of a class method, just to show the usage of various javadoc Tags.

Parameters:

- `numA` - This is the first parameter to addNum method
- `numB` - This is the second parameter to addNum method

Returns:

`int` This returns sum of numA and numB.

main

<code>public static void main(java.lang.String[] args)</code> <code>throws java.io.IOException</code>
--

This is the main method which makes use of addNum method.

Parameters:

- `args` - Unused.

Throws:

- `java.io.IOException` - On input error.

See Also:

- `IOException`

Conditions imbriquées

- Solution 1: Mal écrit

```
if (latitude > 30 && longitude > 120) {if (latitude <= 60 && longitude <= 150)
mapSquareNo = 1; else if (latitude <= 90 && longitude <= 150) mapSquareNo = 2
else print "Not on the map";} else print "Not on the map";
```

- Solution 2: Bien écrit, mais mal construit

```
if (latitude > 30 && longitude > 120)
{
    if (latitude <= 60 && longitude <= 150)
        mapSquareNo = 1;
    else
        if (latitude <= 90 && longitude <= 150)
            mapSquareNo = 2;
        else
            print "Not on the map";
}
else
    print "Not on the map";
```

- Solution 3: imbrication acceptable

```
if (longitude > 120 && longitude <= 150 && latitude > 30 && latitude <= 60)
mapSquareNo = 1;
else
    if (longitude > 120 && longitude <= 150 && latitude > 60 && latitude <= 90)
        mapSquareNo = 2;
    else
        print "Not on the map";
```

Anti-patrons linguistiques

- getX() fait plus qu'accéder à un champ
- isX() retourne plus qu'un booléen
- setX() retourne autre que void
- getApple() retourne une liste au lieu d'un seul objet
- Nom de l'attribut contredit son type AssocEnd start;

Arnaoudova, V., Di Penta, M. & Antoniol, G. Empirical Software

Engineering 21: 104 (2016).

<https://doi.org/10.1007/s10664-014-9350-8>

Normes de programmation

- Les normes peuvent être à la fois une bénédiction et une malédiction
- Modules peu cohésifs sont souvent causés par des règles telles que:
“Chaque module doit contenir entre 35 et 50 instructions exécutables”
- Meilleure règle: *“Le programmeur doit consulter son supérieur avant de construire un module de moins de 35 ou de plus de 50 instructions exécutables”*
- **Aucune norme ne sera jamais universellement applicable**
- Les normes telles que la 1^{ère} seront ignorées
- Idéalement, une norme devrait être **vérifiable sans intervention humaine**
- Le but de standardiser est de rendre la **maintenance plus facile**

Exemples de bonnes normes

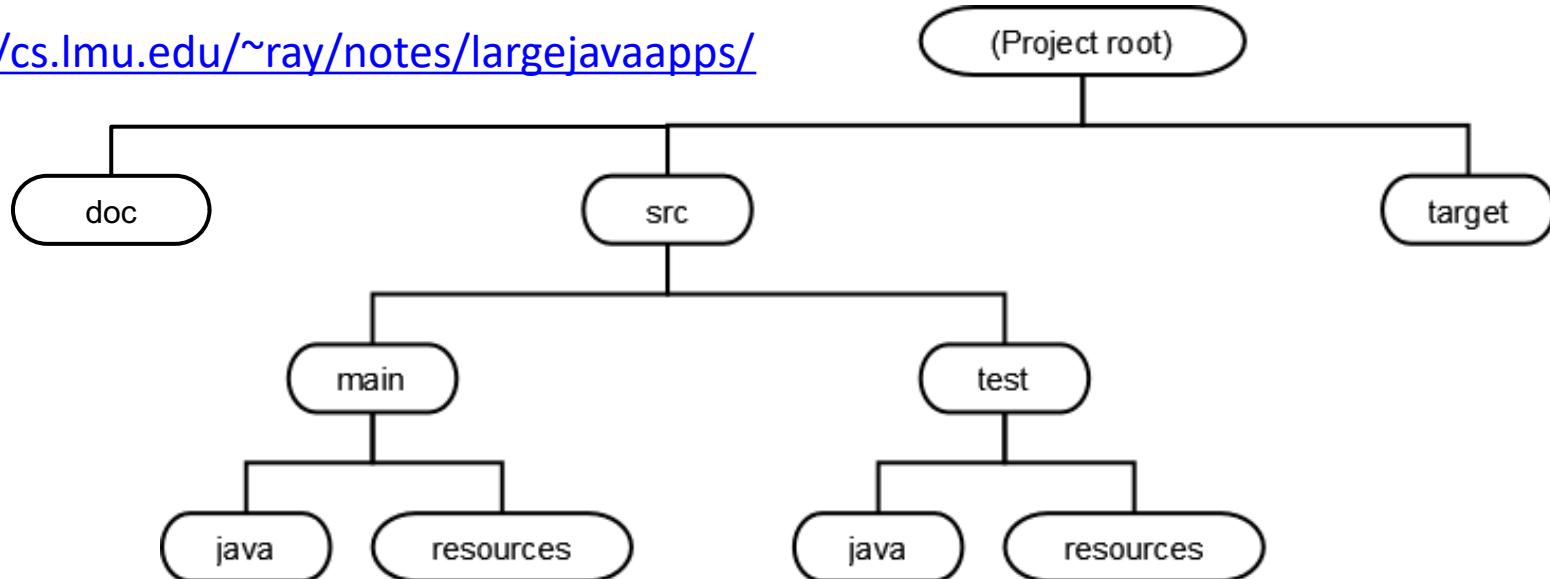
- *L'imbrication d'expressions SI ne doit pas dépasser 3 niveaux, sauf avec approbation*
- *Éviter l'utilisation des GOTO. Néanmoins, avec l'approbation du chef d'équipe , un GOTO vers l'avant peut être utilisé pour le traitement d'erreur*

Mauvaises pratiques de programmation

- <http://thc.org/root/phun/unmaintain.html>
- <http://mindprod.com/jgloss/unmain.html>

Organiser un gros projet Java

<http://cs.lmu.edu/~ray/notes/largejavaapps/>



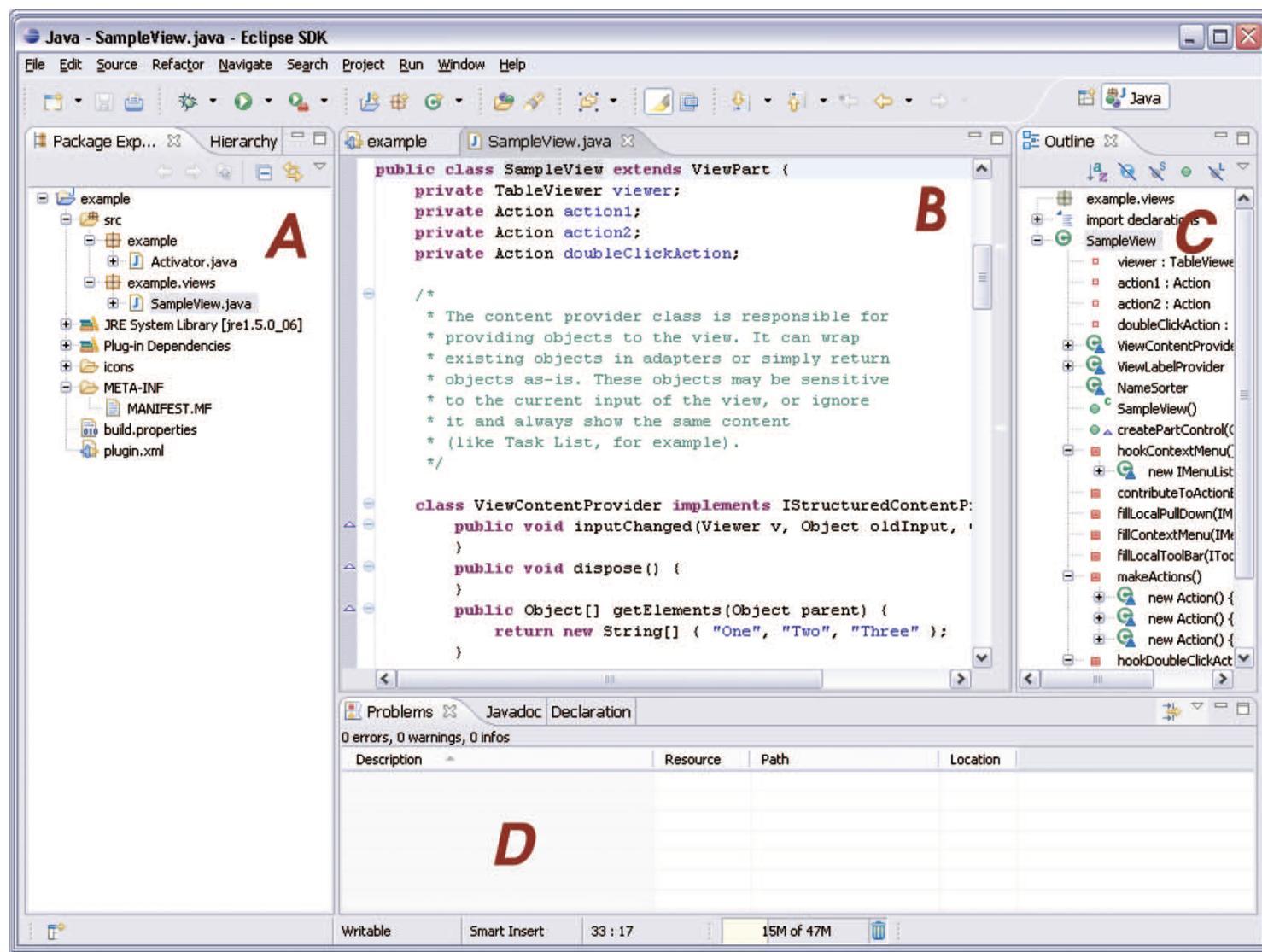
- Organisation structurée des répertoires et fichiers
 - `src/main/java/` : code source
 - `src/test/` : tous le code et autres fichiers pour les tests. Doit correspondre à la structure de `main/`
 - `src/main/resources/` : fichiers textes, images, etc.
 - `target/` : fichiers classes compilés
 - `doc/` : la documentation générée

Outils de programmation

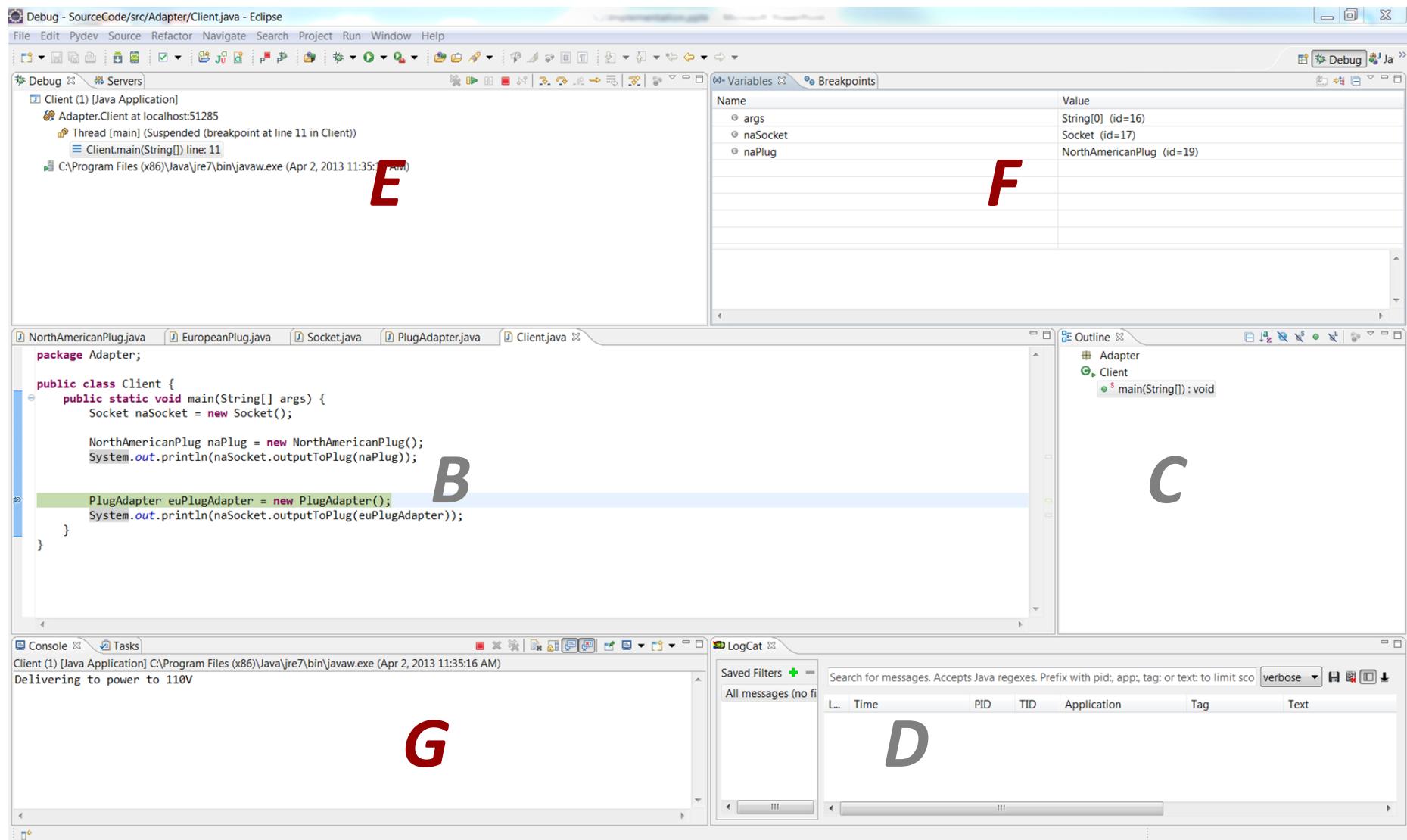
Environnement de développement intégré

- Les développeurs utilisent différents outils durant la vie du logiciel
 - Vérificateur d'interfaces, compilateur, éditeur texte, etc.
- Les outils peuvent être combiné dans un **environnement** qui apporte un support par ordinateur à plusieurs activités
 - Programmation, test, contrôle de configuration, gestion de révisions, moteur de production, interpréteur, débogueur
- Une **IDE** intègre ces environnements et outils dans une interface utilisateur commune et uniforme
 - Même présentation
 - Outils communiquent avec des données compatibles
- Idéalement, une IDE devrait **intégrer tout le processus** de développement

Utilisation d'une IDE : Eclipse



Perspective de débogage



Utilisation des différentes vues Éclipse

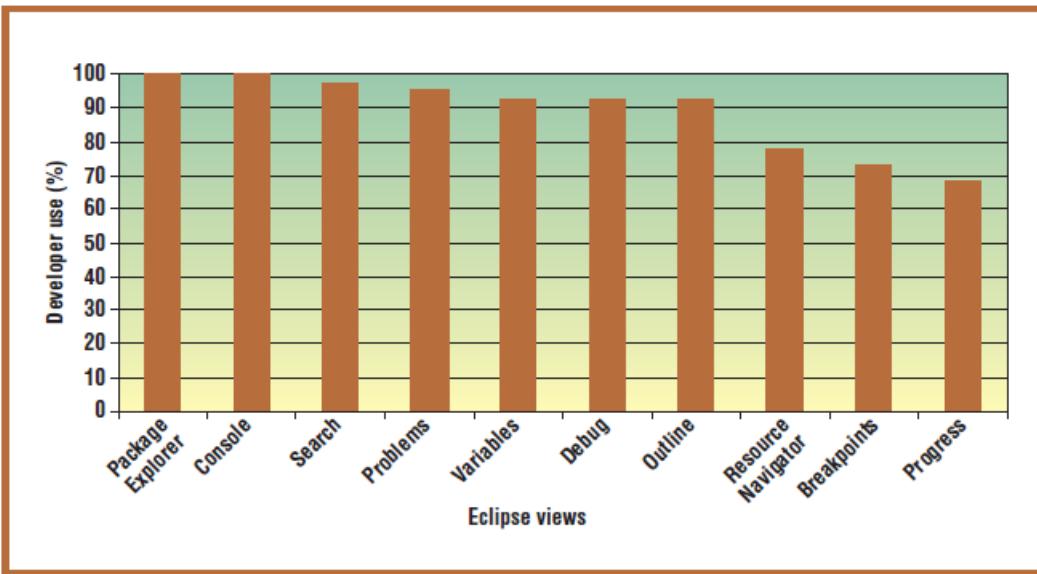
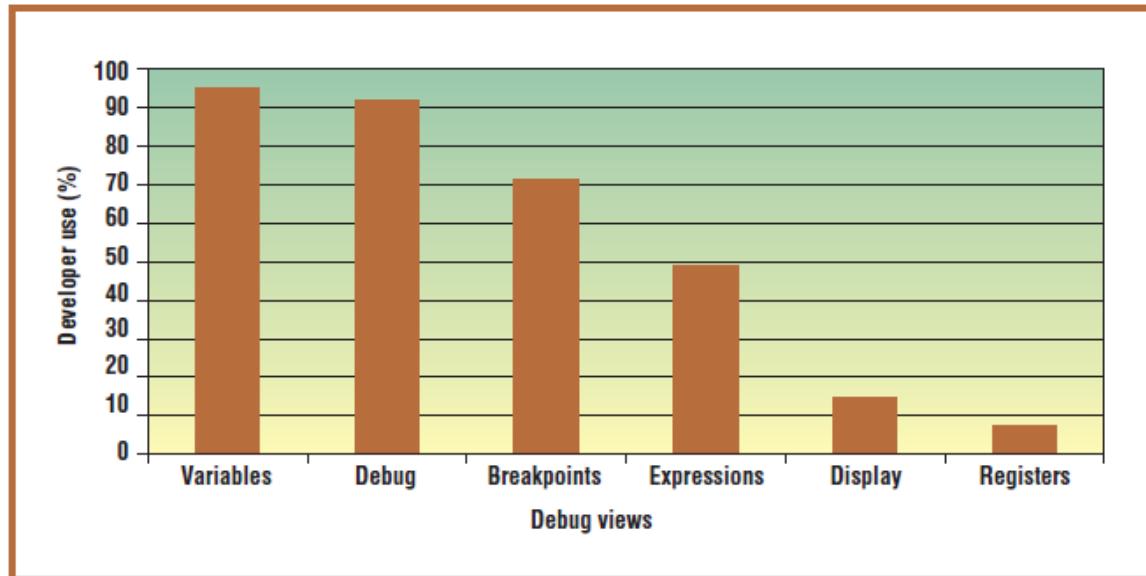


Figure 3. The top 10 of 42 views shipped with the standard Eclipse distribution, based on the percentage of the 41 developers who made at least one selection in each view.

Figure 8. The percentage of the 41 developers using each of the six debugging views.



Utilisation des commandes IDE

Table 2

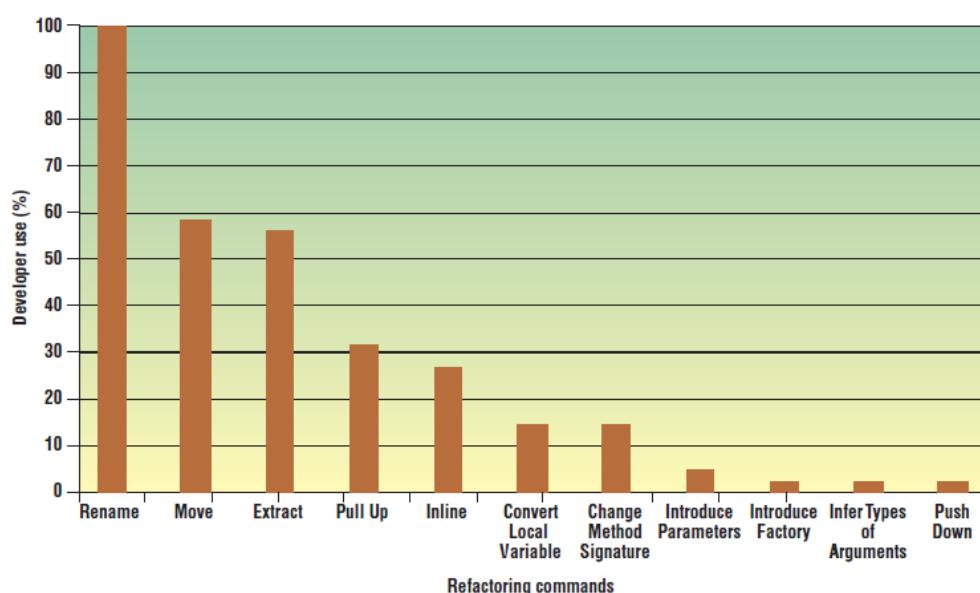
Top 10 commands executed by the most developers

Command	Identifier	No. of users
Delete	org.eclipse.ui.edit.delete	41
Save	org.eclipse.ui.file.save	41
Paste	org.eclipse.ui.edit.paste	41
Content assist	org.eclipse.ui.edit.text.contentAssist.proposals	41
Copy	org.eclipse.ui.edit.copy	41
Undo	org.eclipse.ui.edit.undo	41
Cut	org.eclipse.ui.edit.cut	40
Refresh	org.eclipse.ui.file.refresh	40
Show view	org.eclipse.ui.window.showViewsShortlist	40

Table 3

Top 10 commands executed across all 41 developers

Command	Identifier	Use (%)
Delete	org.eclipse.ui.edit.delete	14.3
Save	org.eclipse.ui.file.save	11.3
Next word	org.eclipse.ui.edit.text.goto.wordNext	7.3
Paste	org.eclipse.ui.edit.paste	6.8
Content assist	org.eclipse.ui.edit.text.contentAssist.proposals	6.7
Previous word	org.eclipse.ui.edit.text.goto.wordPrevious	5.9
Copy	org.eclipse.ui.edit.copy	4.6
Select previous word	org.eclipse.ui.edit.text.select.wordPrevious	3.4
Step (debug)	org.eclipse.debug.ui.debugview.toolbar.stepOver	3.2

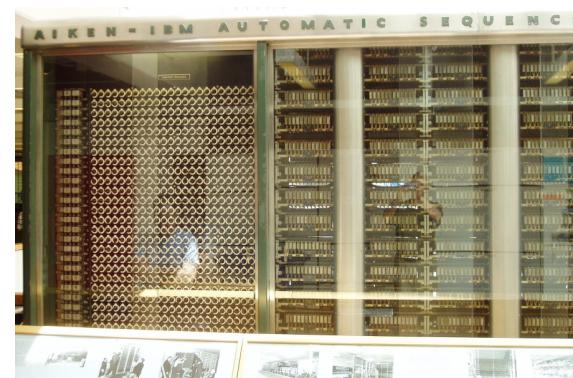


Débogage

Origine du terme « bug »

9/9

0800 Anton starts
 1000 " stopped - anton ✓ { 1.2700 9.037 847 025
 13' UC (03) MP-MC 1.2847 9.037 846 995 connect
 (03) PRO 2 2.130476415
 connect 2.130676415
 Relays 6-2 m 033 failed special speed test
 in relay " 11.00 test.
 Relays changed
 1100 Started Cosine Tape (Sine check)
 1525 Started Multi Adder Test.
 1545 Relay #70 Panel F
 (Moth) in relay.
 First actual case of bug being found.
 1600 Anton starts.
 1700 closed down.



- En 1946, **Grace Hopper** a rejoint le Computation Laboratory à Harvard
- Elle travaillait sur des applications militaires Mark II de chars d'assauts
- Les opérateurs ont retracé une erreur dans Mark II à cause d'un **papillon de nuit** coincé dans un transmetteur
- L'insecte (*bug*) a été enlevé et collé dans le livre de registre



Comment un développeur perçoit un bug



<http://modeling-languages.com/wp-content/uploads/2012/08/howtofaceabug.jpg>

Sévérité des défauts

- **Bloqueur:** empêche de poursuivre les tests jusqu'à ce qu'il soit corrigé ou une alternative est identifiée
- **Critique:** impossible d'éviter la perturbation d'opérations essentielles; sécurité compromise
- **Majeur:** opération essentielle est affectée, mais on peut continuer
- **Mineur:** opération non-essentielle est perturbée
- **Inconséquent:** pas d'impact significatif sur les opérations

*IEEE Standard 1044-2009:
IEEE Standard Classification for Software Anomalies*

Débogage

- Processus méthodique de trouver et réduire le nombre de **défauts** dans un programme **lors de l'exécution** afin qu'il se comporte tel qu'attendu
- Un débogueur **simule l'exécution** du code à examiner en pouvant l'exécuter et le **suspendre** lorsque des conditions spécifiques sont rencontrées
- Caractéristiques:
 - **Exécution en étapes (step)**: animation du programme exécutant une expression à la fois
 - **Suspension (break)**: faire une pause afin d'examiner l'état courant

Méthodes de débogage

- Assertions
- Exceptions
- Traçage
- Log manuel
- Débogueur interactif
- Débogage en direct

Assertion

- Vérifier une condition lors de l'exécution et mettre fin au programme en cas d'échec

```
int total = countNumberOfUsers();  
if (total % 2 == 0) {  
    // total is even  
} else {  
    // total is odd and non-negative  
    assert total % 2 == 1;  
}
```

- Utilisé dans les tests unitaires

Exception

- Déetecter une erreur logique ou un cas extrême
- **Lancer** une exception quand une erreur se produit
- **Traiter** une exception pour corriger l'erreur
- Continuer l'exécution, arrêter l'exécution, ou propager (récurativement) l'exception au module supérieur

```
try {
    line = console.readLine();

    if (line.length() == 0) {
        throw new EmptyLineException("The line read from console was empty!");
    }

    console.println("Hello %s!" % line);
    console.println("The program ran successfully");
}
catch (EmptyLineException e) {
    console.println("Hello!");
}
catch (Exception e) {
    console.println("Error: " + e.message());
}
finally {
    console.println("The program terminates now");
}
```

Traçage

- La **trace d'appels (stack trace)** retrace l'historique d'exécution du programme
- Garde une trace de tous les appels de méthodes faits

```
def a():
    b()

def b():
    c()

def c():
    erreur()

a()
```

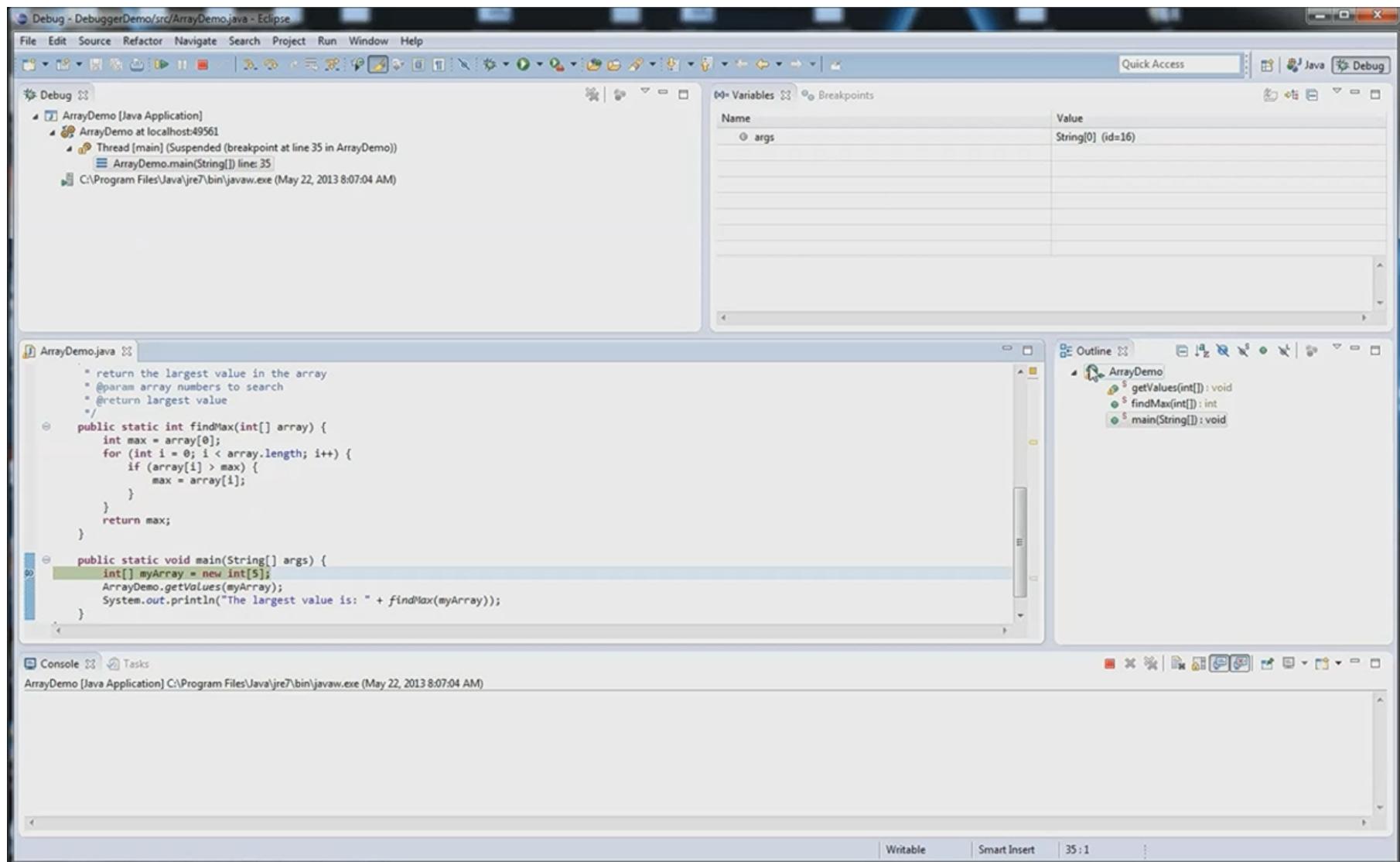
```
Traceback (most recent call last):
  File "tb.py", line 10, in <module>
    a()
  File "tb.py", line 2, in a
    b()
  File "tb.py", line 5, in b
    c()
  File "tb.py", line 8, in c
    erreur()
NameError: global name 'erreur' is not defined
```

Log manuel

- Afficher l'information sur **l'état** du programme ou son **flux de contrôle** dans un log
- Instructions manuellement insérées dans le code
- S'affiche dans la console, un fichier, etc.

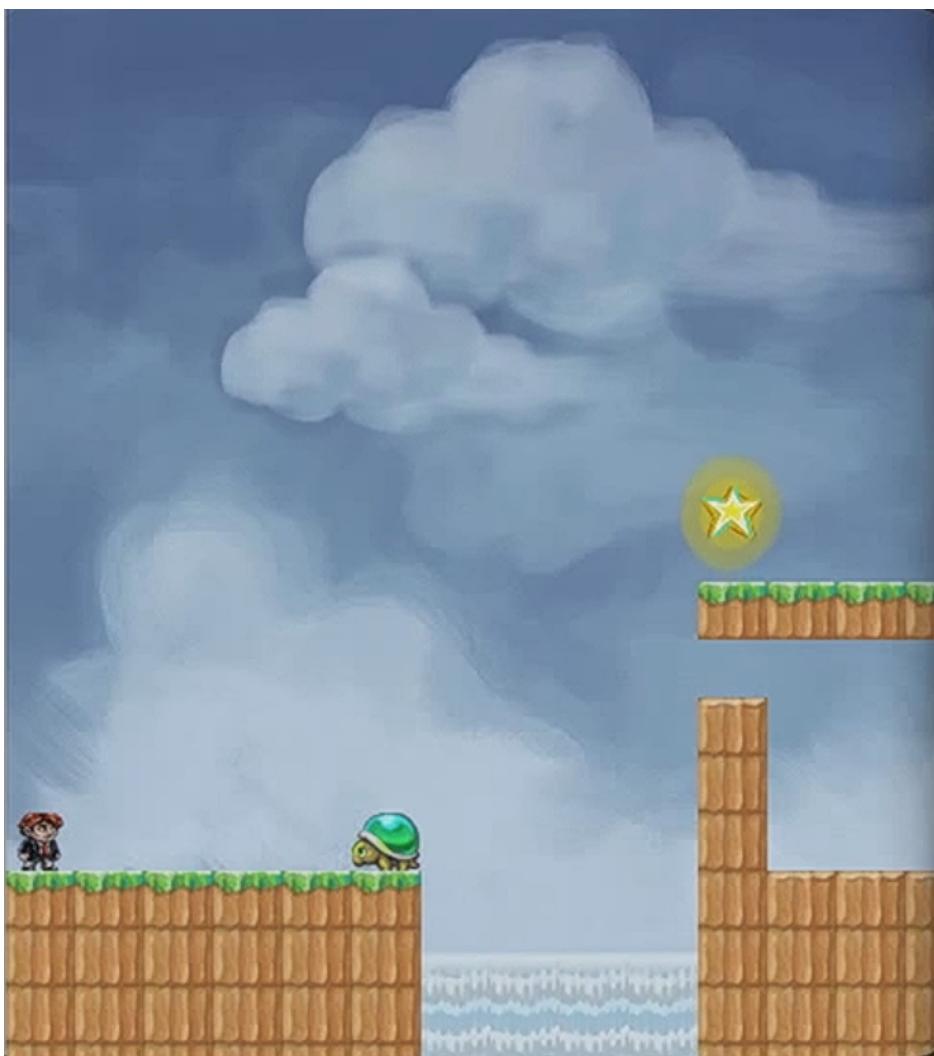
```
import java.util.logging.*;
class myClass {
    private Logger logger = Logger.getLogger(this.getClass().getPackage().getName());
    void m() {
        logger.severe("Message de haute sévérité");
        logger.warning("Message d'alerte");
        logger.info("Message d'information");
    }
}
```

Débogueur interactif



<https://www.youtube.com/watch?v=hdNfx3DNHRQ>

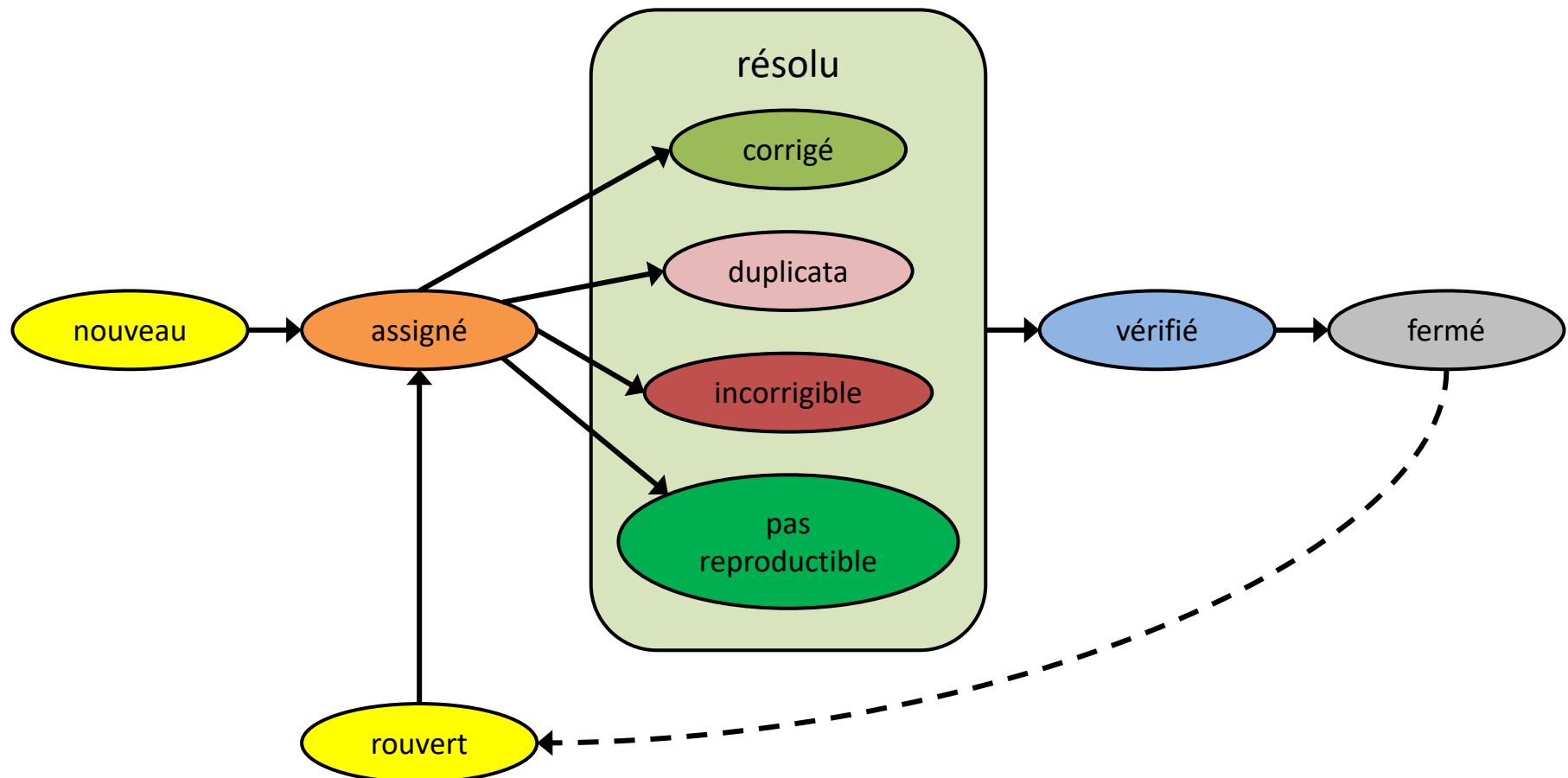
Débogage en direct



```
//=====
// GameCharacter
var GameCharacter = new Class({  
    Extends: GameObject,  
  
    initialize: function () {  
        this.x = 0;  
        this.y = -1;  
        this.yVelocity = 0;  
        this.jumping = false;  
        this.running = false;  
        this.facingLeft = false;  
    },  
  
    tick: function (dt, commands) {  
        var oldX = this.x;  
  
        if (commands.left) {  
            this.x -= 8 * dt;  
        }  
        if (commands.right) {  
            this.x += 8 * dt;  
        }  
  
        if (commands.up && !this.jumping) {  
            this.yVelocity = -25;  
        }  
  
        this.yVelocity += 100 * dt;  
        this.y += this.yVelocity * dt;  
  
        var hitInfo = this.hitTest();  
  
        this.jumping = (hitInfo.bumpedY >= this.y);  
        if (hitInfo.bumpedY != this.y) {  
            this.yVelocity = 0;
```

<https://www.youtube.com/embed/EGqwXt90ZqA?start=640&end=875>

Cycle de vie d'un bogue



Logiciel de suivi de bogues

JIRA Dashboards Projects Issues Agile Tempo Test Sessions Accounts Create Issue Quick Search ? Settings User

Filters New filter Find filters My Open Issues Reported by Me Recently Viewed All Issues Favourite Filters 2.0 Blocker issues Angry Nerds Testin... Angry Nerds updat...

High priority for 2.0 Owned by: Christina Bang

project = ANGRY AND updated >= -1d OR com| comment Comment Count - cf[10133] component

T	Key	Summary	Assignee	Reporter	P	Status	Resolution	Created	Updated	Due
BUG	ANGRY-232	Housecleaning crew didn't finish the job	Unassigned	penny	BUG	Open	Unresolved	14/Sep/12	17/Sep/12	
BUG	ANGRY-226	My archived issue	Edwin Wong	Edwin Wong	BUG	Closed	Fixed	02/Aug/12	18/Oct/12	
BUG	ANGRY-186	ANGRY-148 / This is the first sub task	Unassigned	Michael Tokar	BUG	Open	Unresolved	19/Jul/12	18/Oct/12	
BUG	ANGRY-168	ANGRY-163 / Cancel button look	Unassigned	Christina Bang	BUG	Open	Unresolved	15/May/12	09/Aug/12	
BUG	ANGRY-132	I should be able to hit enter in addition to 'submit'.	Christina Bang	Ken Olofson	BUG	In Progress	Unresolved	04/May/12	16/Jun/12	10/Jun/12
BUG	ANGRY-132	/ Decide on new or bricks	Bryan Rollins	Bryan Rollins	BUG	In QA Review	Fixed	15/Apr/12	15/Apr/12	
BUG	ANGRY-132	metimes leaves bug upside in chair	Christina Bang	Penny	BUG	Open	Unresolved	08/Mar/12	18/Oct/12	
BUG	ANGRY-132	are falling down stently	Unassigned	Roy Krishna	BUG	Open	Unresolved	20/Jan/12	18/Oct/12	
BUG	ANGRY-132	nic Voting	Ross Chaldecott	Roy Krishna	BUG	To Do	Unresolved	31/Oct/11	15/Feb/12	
BUG	ANGRY-85	/ subby	Unassigned	Roy Krishna	BUG	Open	Unresolved	12/Sep/11	09/Aug/12	
BUG	ANGRY-85	ond issue created via	Unassigned	Bryan Rollins	BUG	Open	Unresolved	06/Sep/11	18/Oct/12	

GOT FEEDBACK? Log In Quick Search

Angry Nerds / ANGRY-56 Level 6 looks the same as level 5

Edit Assign Assign To Me Comment More Actions Back to Todo Back to In Progress Workflow Share Views

Details

Type: Bug Priority: Major Status: Done (View Workflow) Resolution: Fixed

Affects Version/s: 1.2 Component/s: Nerd Actions

Description

Can you make a new background for level 6? Maybe a different sky color.

Issue Links

Get Satisfaction Topic [Get Satisfaction] I have a question about that one thing

Salesforce.com Account [Salesforce.com] GenePoint

Sauce Labs Bug [Sauce Labs] A bug in Firefox 5 (edited in JIRA)

Zendesk Ticket [Zendesk] 233 - There seems to be an issue

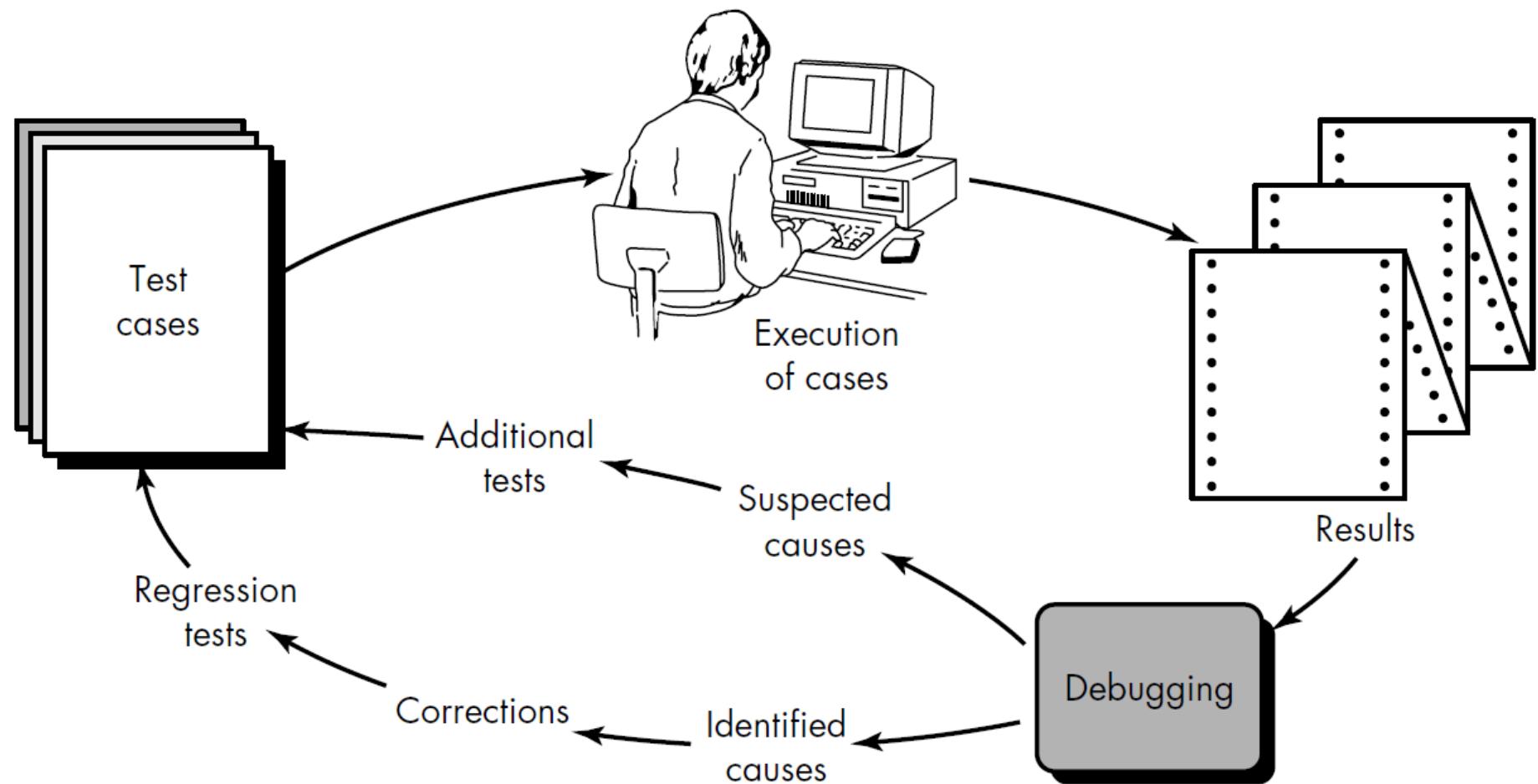
Activity

All Comments Work Log History Activity Test Sessions Builds

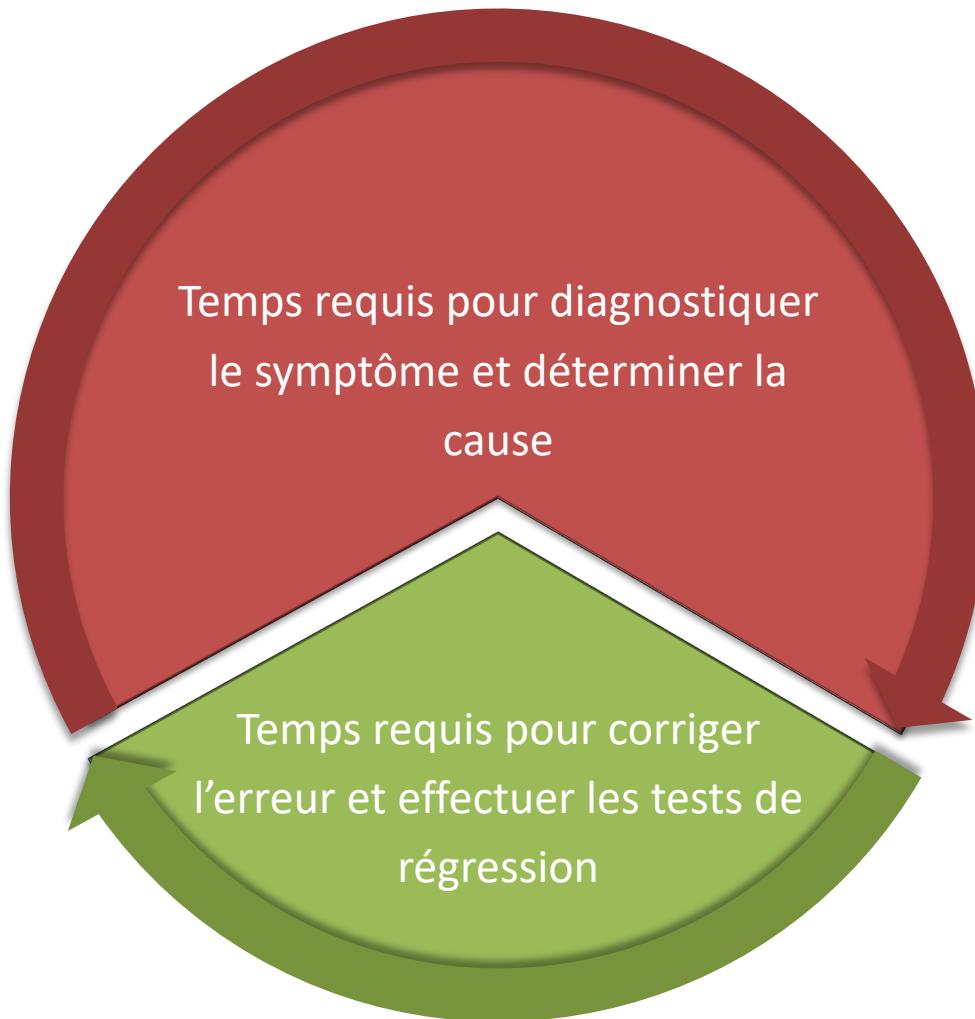
Christina Bang created a link from ANGRY-56 - Level 6 to ANERDS-62 (JAC) 3 hours ago

Christina Bang changed the status to Done on ANGRY-56 - Level 6 with a resolution of 'Fixed' 3 hours ago

Processus de débogage

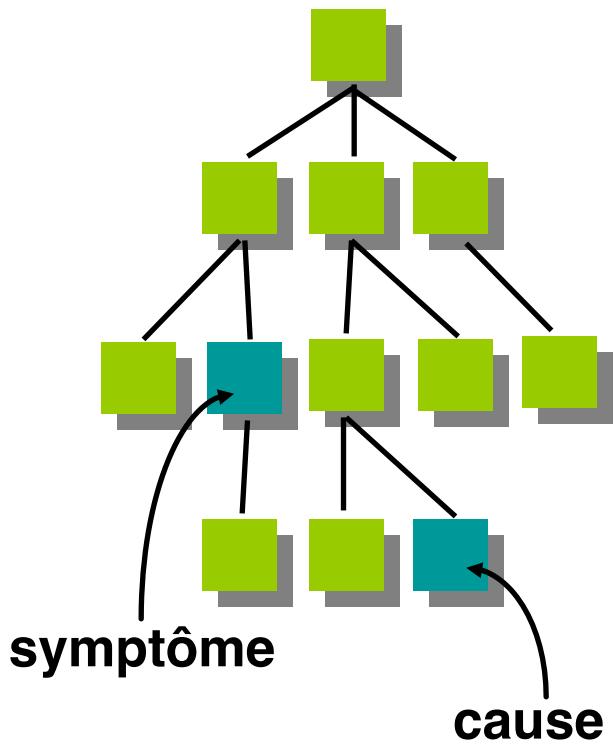


Effort de débogage



Symptômes et causes

- Symptômes et causes peuvent être **séparés géographiquement**
- Symptôme peut disparaître quand un **autre problème** est corrigé
- Symptôme peut être **intermittent**
- Cause peut être due à une **combinaison de manipulations correctes**
- Cause peut être due à une **erreur système** ou du **compilateur**
- Cause peut être due à une **hypothèse** que l'on croit vraie



Correction de bugs et échelle de désastre

