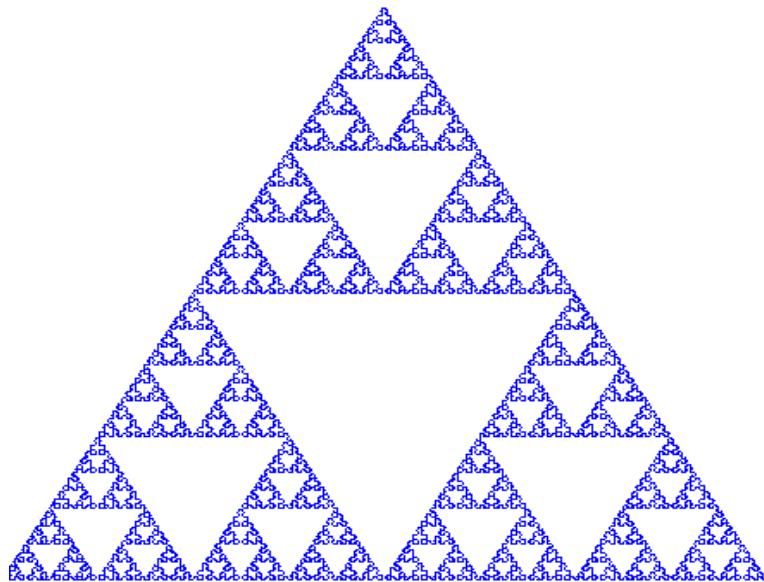


# Génie Logiciel : **Java**

**Projet « autoformation à l'algorithmique »**



# Table des matières

Projet « autoformation à l'algorithmique ».....	1
I Objectifs.....	3
II Décisions.....	3
Configuration requise :.....	3
Lancement :.....	3
Travaux sur les algorithmes :.....	4
Interface Homme Machine :.....	4
Priorités :.....	4
III Points difficiles.....	4
IV Schéma entité-association de la base de données.....	5
V Diagrammes des objets (un par package).....	5
VI Planning (diagramme de Gantt).....	9
VII Code .....	11
VIII Tests.....	11
1Tests unitaires.....	11
2Tests de l'interface enseignant.....	13
3Tests de l'interface étudiant.....	16
4Tests de l'interface algorithme.....	16
IX Fiche Génie Logiciel.....	18
X Bilan.....	20

# I Objectifs

Ce projet est l'aboutissement de connaissances acquises en programmation orientée objet et redécouvertes et approfondies en Java. L'ergonomie, l'esthétique et l'exhaustivité de la mise en oeuvre de chaque tâche seront plutôt délaissées pour que puisse être privilégié le fait d'utiliser un maximum d'aspects du java et d'approfondir davantage encore les connaissances.

Même si l'on n'explore pas toutes ses facettes, le programme en projet en lui-même, nécessite de multiples connaissances à commencer par : la conception UML, les bases de données, les composants Swing ou awt, JDBC ; d'autres connaissances s'avèrent encore bien indispensables : log4j, Junit, l'héritage et le polymorphisme, la protection, dynamicité et staticité, Ant, les environnements de développement, l'édition de liens dynamique, Javadoc.

## II Décisions

### ■ Configuration requise :

Pour l'exécution : Une application Java est hautement portable à condition qu'une JVM soit installée sur le support. Ce programme devra aussi quant à lui bénéficier d'un accès à un serveur postgres avec la base de données nécessaire pour pouvoir fonctionner correctement. Pour pouvoir afficher les descriptions HTML et codes C des algorithmes, le programme se connecte aussi à internet, le programme doit donc pouvoir bénéficier d'un accès direct à internet. Les fichiers .jar générés sont executables.

Pour la compilation et le développement : des variables d'environnement JAVA\_HOME et ANT\_HOME sont rajoutées au PATH à chaque démarrage grâce au fichier .bashrc, les versions eclipse, ant et jdk installées sont chacune les plus récentes trouvées.

### ■ Lancement :

J'ai vainement tenté de modifier le fichier MANIFEST.MF à partir du script Ant build.xml, pour que le fichier .jar soit executable, `$ java -jar MyProject-20080102.jar` me renvoie toujours :

```
Exception in thread "main" java.lang.UnsupportedClassVersionError:  
fr/univtl/ganne882/project2007/gui/Main (Unsupported major.minor version 50.0)
```

pourtant, j'ai inclus les librairies ajoutées à ce .jar mais rien n'y fait, j'ai aussi vérifié la cohérence des versions java du jdk compilant mes sources et du java executant le .jar, c'est bien le même.

## ■ Travaux sur les algorithmes :

Deux méthodes sont possibles, soit tous les étudiants proposent des solutions différentes pour chaque algorithme et l'enseignant doit donc avoir une première interface où il consulte pour chaque étudiant une solution et une seconde où il propose sa solution idéale que les étudiants vont finalement pouvoir consulter ; soit tous les étudiants travaillent de concert sur les solutions aux algorithmes, ont accès l'un au travail de l'autre et proposent à chacun leur tour les avancées effectuées sur un algorithme. C'est cette seconde solution qui a été choisie, elle paraissait plus logique.

## ■ Interface Homme Machine :

Deux IHM différentes se présentent aux deux types d'utilisateurs du programme : une interface étudiant et une interface enseignant afin de ne pas surcharger inutilement leurs espaces de travail. L'un ou l'autre type d'utilisateur est reconnu au démarrage de l'application lorsqu'il s'y connecte. Une dernière fenêtre (algorithme) est consultable par l'un ou l'autre des utilisateurs, lorsque l'étudiant clique sur « consulter/proposer » ou lorsque l'enseignant clique sur « Ajouter » ou « Editer ».

## ■ Priorités :

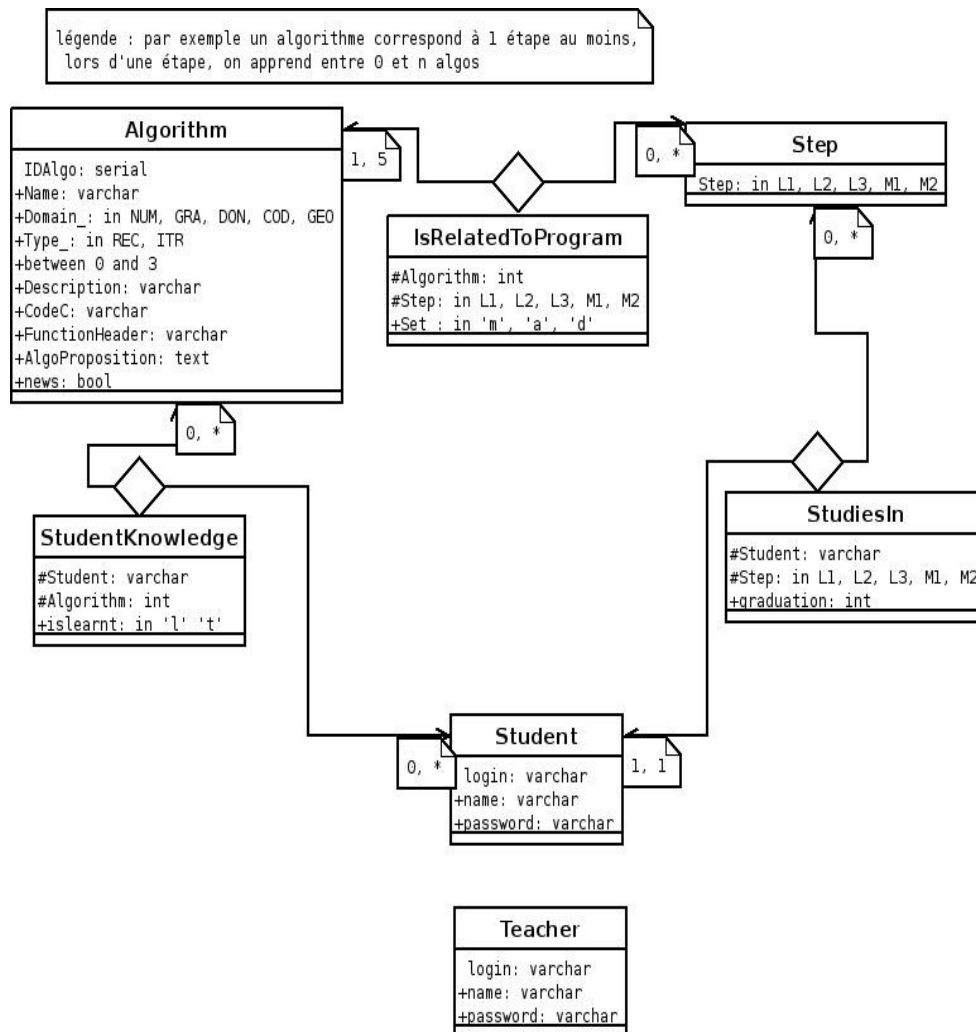
Les tâches (présentées distinctement dans la partie planning) ont été envisagées dans l'ordre suivant afin d'optimiser le temps imparti : Conception, IHM, Bases de données, Mise en relation IHM/BD, Débogage et options, Préparation d'un Compte-rendu.

# III Points difficiles

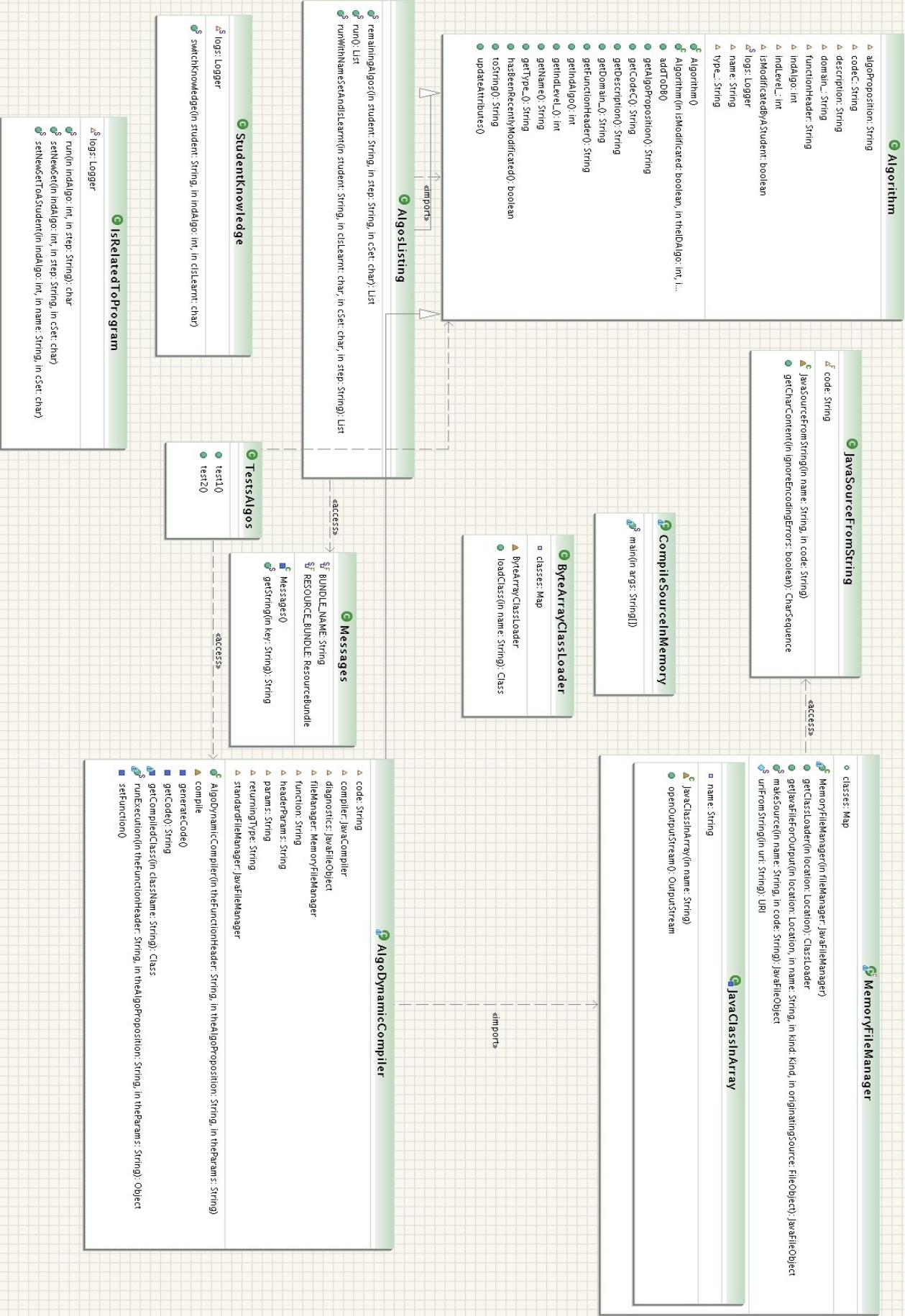
Lorsque l'on a des obligations pour les congés scolaires et d'autres encore quand des événements surviennent sur l'université, au manque de temps s'ajoutent des difficultés comme le manque de sommeil, la faim, le stress, les délais peuvent alors paraître intenable.

Un élément peu évident a été de se contraindre à utiliser des éléments du codage en Java comme log4j alors qu'on ne les juge pas nécessaires, pourtant il est sûr que finalement, ces éléments font dans leur ensemble gagner du temps.

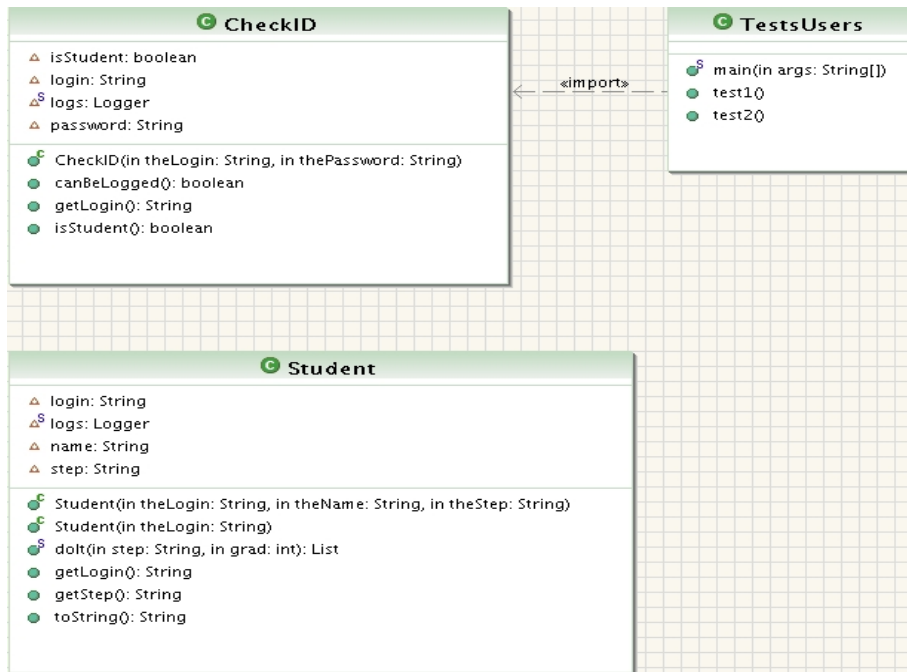
## IV Schéma entité-association de la base de données.



## V Diagrammes des objets (un par package)









## VI Planning (diagramme de Gantt)

Tâches	Semaines							
	Sujet fourni mercredi 21/11				Soutenance Mercredi 9/01			
	47	48	49	50	51	52	1	2
C1								
C2								
C3								
C5								
I1								
I2								
I3								
I4								
D1								
E4								
C4								
B1								
B2								
E3								
E1								
E2								
E5								
I5								
D2								
A1								
U1								
U2								
A2								
E6								
A3								
A4								
O1								
O2								
D3								
O3								
O4								

Tâches	
Conception :	
<b>C1</b>	Presentation et compréhension du projet
<b>C2</b>	Dessins de l'interface
<b>C3</b>	Gestion de priorités des options compte tenu du manque de temps
<b>C4</b>	Conception entité-association de la base de données
<b>C5</b>	Conception du modèle UML des classes Java
Environnement de développement :	
<b>E1</b>	Choix de version d'Eclipse, d'Ant
<b>E2</b>	Installation et configuration des modules log4j, EclipseUML, JDBCpostgres8.2, Junit.
<b>E3</b>	Redaction d'un script Ant pour la compilation
<b>E4</b>	Installation d'un outil graphique pour SGBD postgres.
<b>E5</b>	Redaction d'un script log4j
<b>E6</b>	Création de classes Junit, servant à tester les éléments de chaque package
Bases de données :	
<b>B1</b>	Creation des tables et des contraintes
<b>B2</b>	Remplissage des tables avec suffisamment d'exemples utiles
IHM :	
<b>I1</b>	LoginDialog
<b>I2</b>	Fenêtre principale de l'interface enseignant
<b>I3</b>	Fenêtre principale de l'interface étudiant
<b>I4</b>	Fenêtre principale d'accès commun pour le travail sur un algorithme
<b>I5</b>	Codage des ecouteurs sans gestion des requêtes à la BD
Gestion des utilisateurs	
<b>U1</b>	Fonctions et requêtes utiles au LoginDialog
<b>U2</b>	Classe etudiant utile aux consultations des enseignants sur leur travail.
Gestion des algorithmes	
<b>A1</b>	Classe de base Algorithme et méthodes utiles de requêtes
<b>A2</b>	Requêtes restantes utiles aux derniers Jcomponents écoutés.
<b>A3</b>	Analyse des besoins en étapes et classes supplémentaires et de l'exemple de compilateur-executeur dynamique de J. Seinturier
<b>A4</b>	Adaptation de codes aux besoins du programme pour la compilation dynamique
Divers :	
<b>D1</b>	Ajout de commentaires au code
<b>D2</b>	Procédé aux tests
<b>D3</b>	Rédaction d'un rapport de présentation
Tâches rendues optionnelles	
<b>O1</b>	Ajout d'un mot de passe à la base de données et au lancement du programme
<b>O2</b>	Externalisation des Strings en ressource bundle pour la traduction du programme
<b>O3</b>	Ajout à la BD d'une table Code liée à la table Student et Algorithme pour que les étudiants puissent chacun fournir un code personnalisé et non pas tous travailler sur le même algo
<b>O4</b>	Ajout à la BD d'un champ tests à la table Algorithme pour pouvoir procéder à une série de tests lors de l'essai d'un algo compilé dynamiquement

d'autres options pouvaient être ajoutées mais elles sont longues : le fait de pouvoir créer des étudiants en les allouant à une étape et une promotion, le fait de pouvoir supprimer des algos (avec une demande de confirmation), la création d'un menu principal comprenant le manuel utilisateur, et d'autres informations comme le sujet du projet et le copyright.

# VII Code

## VIII Tests

### 1 Tests unitaires

Certains éléments peuvent être testés par Junit au lieu de faire des essais coups sur coups en passant par l'interface graphique, par exemple :

- la connexion au programme à travers la LoginDialog :

la suite de tests de la classe TestsGUI procède à ce test avec un login et un mot de passe (erroné puis adéquat) pour un enseignant puis pour un étudiant.

- la compilation dynamique :

la suite de tests de la classe TestsAlgos procède à ce test avec à la suite deux genre d'algos différents mais simples.

- Résultat des suites de tests :

```
junit:
[java] ..2008-01-03 14:29:56,980 [main] DEBUG : before construction of instance
[java] 2008-01-03 14:29:56,984 [main] DEBUG : returning type : int
[java] 2008-01-03 14:29:56,985 [main] DEBUG : function called : mult
[java] 2008-01-03 14:29:56,986 [main] DEBUG : params : 2, 3
[java] 2008-01-03 14:29:57,157 [main] INFO : System Compiler:
com.sun.tools.javac.api.JavacTool@1a0c10f
[java] 2008-01-03 14:29:57,249 [main] DEBUG : function header : int mult (int a, int b)
[java] 2008-01-03 14:29:57,250 [main] DEBUG : code : public class AlgoToExecute {
[java] public int mult (int a, int b) {
[java] return (a*b);
[java] }
[java] public int functionNoArgs () {
[java] System.out.println("resultat :: " +mult(2, 3));
[java] return mult(2, 3);
[java] }
[java] public static void main (String [] args) {
[java] AlgoToExecute ae = new AlgoToExecute();
```

```

[java] System.out.println("result : " + ae.functionNoArgs());
[java] }
[java] }
[java] compiled IADC
[java] compiled AlgoToExecute
[java] 2008-01-03 14:29:58,612 [main] INFO : Compilation improved
[java] 2008-01-03 14:29:58,613 [main] DEBUG : classname : AlgoToExecute
[java] 2008-01-03 14:29:58,615 [main] DEBUG : seeking searched method functionNoArgs
[java] 2008-01-03 14:29:58,616 [main] DEBUG : Method 0: public int
AlgoToExecute.functionNoArgs()
[java] 2008-01-03 14:29:58,616 [main] DEBUG : (right method found)
[java] 2008-01-03 14:29:58,617 [main] DEBUG : Creating instance of class AlgoToExecute
[java] resultat :: 6
[java] object : 2 x 3 = 6
[java] integer : 2 x 3 = 6
[java] 2008-01-03 14:29:58,625 [main] DEBUG : returning type : String
[java] 2008-01-03 14:29:58,627 [main] DEBUG : function called : displaySomething
[java] 2008-01-03 14:29:58,628 [main] DEBUG : params : 2
[java] 2008-01-03 14:29:58,629 [main] INFO : System Compiler:
com.sun.tools.javac.api.JavacTool@17d5d2a
[java] 2008-01-03 14:29:58,629 [main] DEBUG : function header : String
displaySomething (int a)
[java] 2008-01-03 14:29:58,630 [main] DEBUG : code : public class AlgoToExecute {
[java] public String displaySomething (int a) {
[java] if (a<4) return "test OK"; else return "test messed up";
[java] }
[java] public String functionNoArgs () {
[java] System.out.println("resultat :: " +displaySomething(2));
[java] return displaySomething(2);
[java] }
[java] public static void main (String [] args) {
[java] AlgoToExecute ae = new AlgoToExecute();
[java] System.out.println("result : " + ae.functionNoArgs());
[java] }
[java] }
[java] compiled IADC

```

```
[java] compiled AlgoToExecute
[java] 2008-01-03 14:29:58,861 [main] INFO : Compilation improved
[java] 2008-01-03 14:29:58,862 [main] DEBUG : classname : AlgoToExecute
[java] 2008-01-03 14:29:58,863 [main] DEBUG : seeking searched method functionNoArgs
[java] 2008-01-03 14:29:58,864 [main] DEBUG : Method 0: public java.lang.String
AlgoToExecute.functionNoArgs()
[java] 2008-01-03 14:29:58,864 [main] DEBUG : (right method found)
[java] 2008-01-03 14:29:58,865 [main] DEBUG : Creating instance of class AlgoToExecute
[java] resultat :: test OK
[java] result for second junit test = test OK
[java] .2008-01-03 14:30:03,290 [main] DEBUG : query : select * from Algorithm order by
name
[java] .2008-01-03 14:30:08,694 [main] DEBUG : query : select a.step, b.name from
studiesin a, student b where 'ganne882' = a.student and a.student = b.login
(...)
[java] ..2008-01-03 14:30:10,200 [main] DEBUG : query : select login, password from
teacher
[java] 2008-01-03 14:30:10,240 [main] DEBUG : query : select login, password from
student
[java] .2008-01-03 14:30:10,264 [main] DEBUG : query : select login, password from
teacher
[java] 2008-01-03 14:30:10,267 [main] DEBUG : query : select login, password from
student
[java] Time: 13,59
[java] OK (8 tests)
```

## 2 Tests de l'interface enseignant



Veuillez vous identifier

Identifiant :

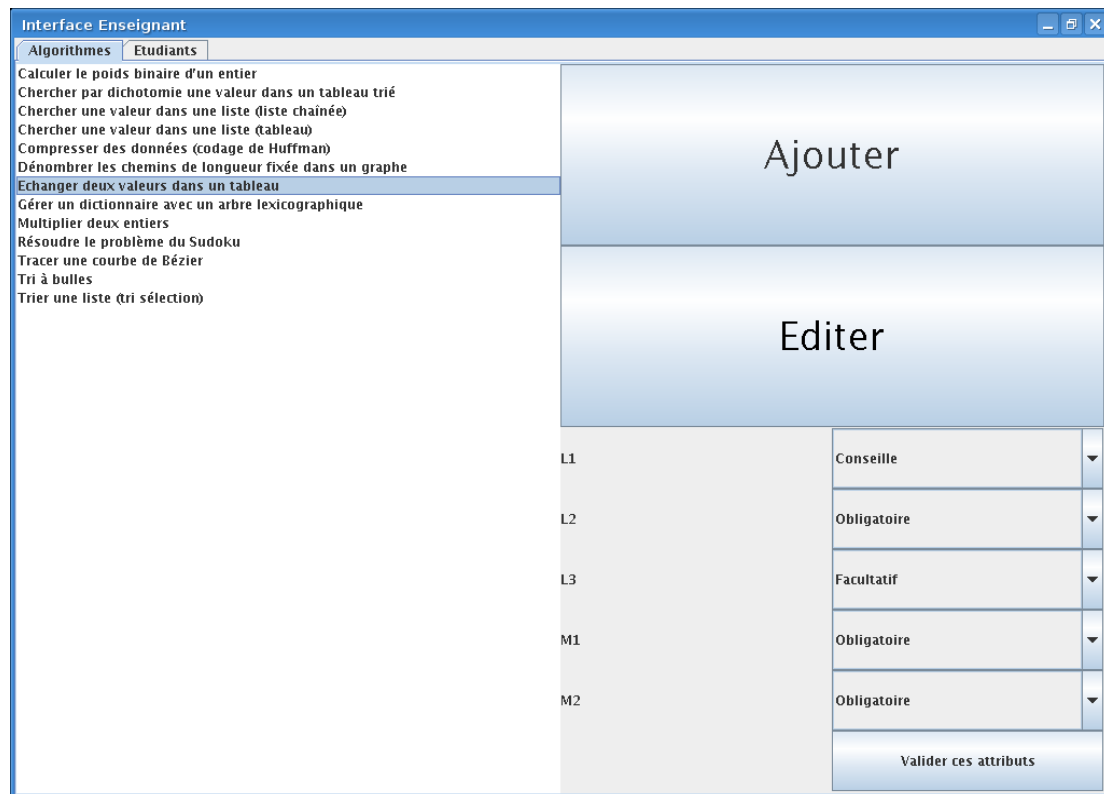
bruno

Mot de passe :

....

OK Quitter

L'enseignant se connecte tout d'abord comme suit et est reconnu...



Il accède ensuite à cette fenêtre munie de deux onglets. Sur ce premier onglet, l'enseignant peut sélectionner un algorithme dans la liste pour y travailler, s'il en choisit plusieurs, avec la touche Ctrl c'est le premier qui est considéré, tandis qu'avec Shift c'est le dernier. Les attributs de l'algorithme sont affichés dans les JComboBox dans le coin bas droite de l'onglet, il peut aisément les modifier. Le cas où aucune donnée concernant l'attribut dans la base est traité, il est affiché « non défini » mais lorsque l'attribut a été donné, il ne peut revenir à « non défini », ce malgré l'affichage de cette valeur (petit défaut). Dans le cas où un étudiant a modifié l'attribut auquel il a accès sur un algorithme (c'est à dire la proposition de solution) le bouton editer se transforme alors comme suit en cliquant sur l'élément en question :



Il peut d'une manière complètement indépendante consulter l'avancement des travaux de ses étudiants sur l'onglet Etudiants :

Interface Enseignant

Algorithmes Etudiants

Nom : Gregory Anne

Etape : M1


Promotion : 2008

Algorithmes Obligatoires	Algorithmes Conseilles	Algorithmes Facultatifs
<p>_____ALGOS APPRIS_____</p> <p>Gérer un dictionnaire avec un arbre lexicographique</p> <p>_____ALGOS A APPRENDRE_____</p> <p>Chercher une valeur dans une liste (tableau)</p> <p>Echanger deux valeurs dans un tableau</p> <p>_____ALGOS SANS INFORMATION D'ATTRIBUTS_____</p> <p>Chercher par dichotomie une valeur dans un tablea...</p>	<p>_____ALGOS SANS INFORMATION D'ATTRIBUTS_____</p> <p>Calculer le poids binaire d'un entier</p>	<p>_____ALGOS A APPRENDRE_____</p> <p>Compresser des données (codage de Huffman)</p>

Exporter les algos sans info de connaissance selectionnes en algos appris pour toute la promo  
 Exporter les algos sans info de connaissance selectionnes en algos non appris pour toute la promo  
 Exporter les algos sans info de connaissance selectionnes en algos appris pour cet etudiant  
 Exporter les algos sans info de connaissance selectionnes en algos non appris pour cet etudiant

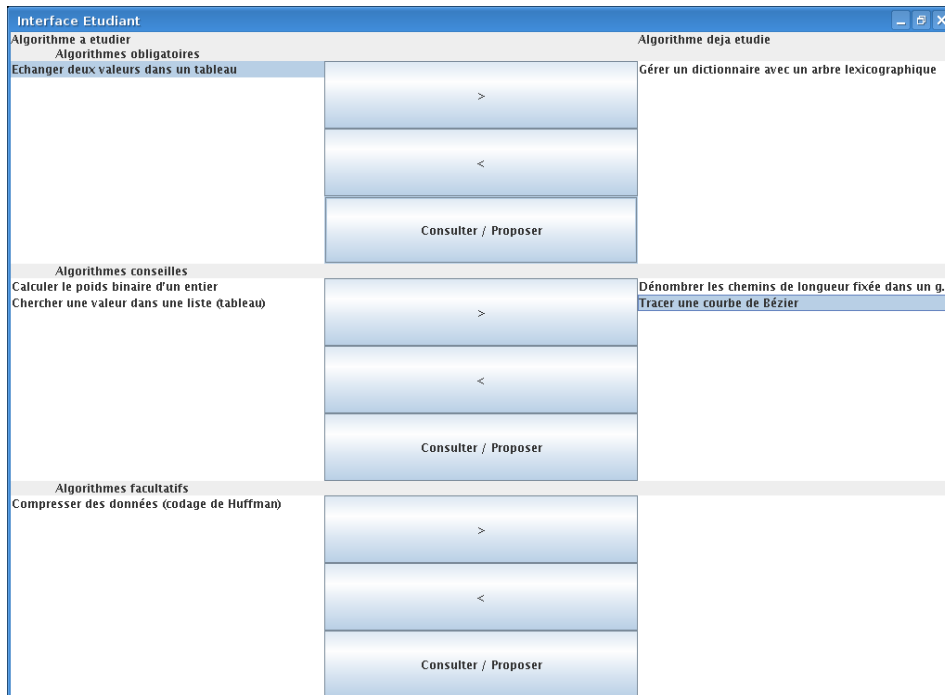
Les listes sont d'abord vides mais l'utilisateur choisit une promo puis une étape puis un nom, cet ordre là est guidé puisque les combobox ne sont rendues accessibles qu'au fur et à mesure. Les quatre boutons au bas sont explicites et leur effet est visible immédiatement puisque les listes sont rafraichies en fonction. Le cas est géré si plusieurs algorithmes sont considérés en même temps pour des opérations et aussi s'il y a incohérence entre les algorithmes selectionnés et l'opération réclamée : la fenetre d'avertissement suivante s'affiche :

Message

 certains elements selectionnes ne correspondaient pas a des algos sans information de connaissance, aucune operation n'a ete entreprise les concernant

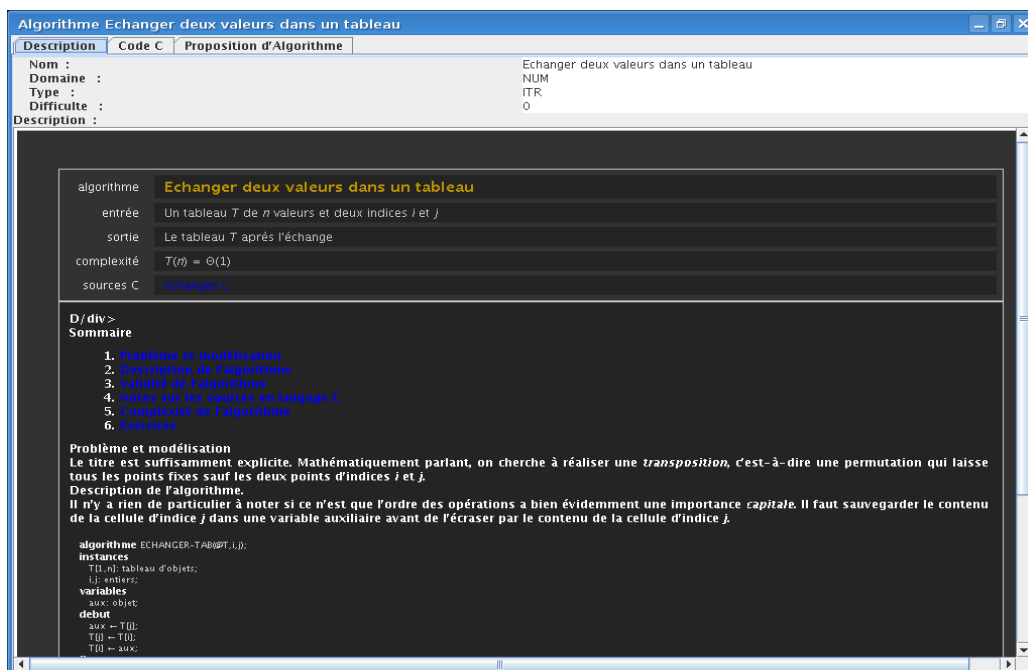
OK

### 3 Tests de l'interface étudiant



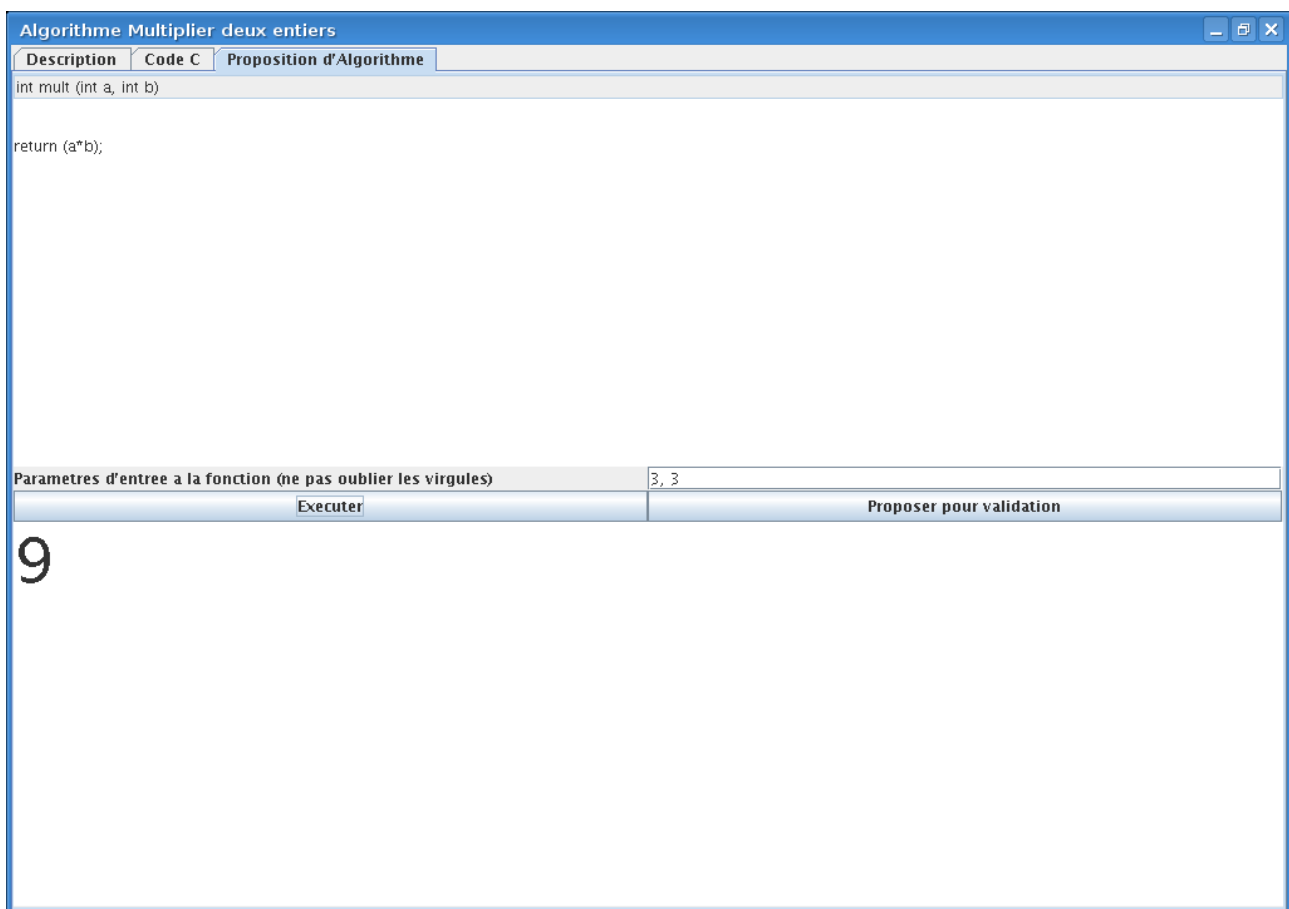
Après s'être connecté de la même manière qu'un enseignant, l'étudiant arrive sur cette fenêtre, d'où il peut manipuler les algorithmes que l'enseignant lui a préalablement mis à disposition. Il leur attribue une valeur 'su' ou 'non su' sur laquelle il peut indéfiniment revenir. Lorsque plusieurs éléments sont sélectionnés, c'est l'élément le plus haut dans la liste qui est considéré (avec Shift ou Ctrl). En cliquant sur le bouton « Consulter/proposer » il accède à la fenêtre algorithme mais n'a pas les privilèges de l'enseignant.

### 4 Tests de l'interface algorithme





L'interface algorithme est composée de trois onglets : Description, Code C, et Proposition d'Algorithme. Les descriptions d'algorithmes une fois terminées sont disponibles sur le site <http://zanotti.univ-tln.fr/>, une fois que la description est achevée, l'enseignant remplace donc la description temporaire qu'il a entrée dans la base par l'adresse de la page contenant la description. Le programme distingue donc les deux cas de figure (il vérifie que la description commence par http:// ou pas). Si la description est reconnue comme un code html, celle-ci est traduite et affichée comme tel, sinon le texte est affiché brutalement. Le programme gère le fait que la connexion internet puisse être déficiente ou encore que l'adresse indiquée n'est pas la bonne en affichant cette information. Le fonctionnement est le même pour l'onglet Code C : il est censé être disponible sur internet.



L'onglet proposition d'algorithme est l'onglet par lequel l'étudiant peut tester ces solutions d'algorithmes. La zone de texte du bas est l'endroit où est affichée la valeur de retour de l'algorithme en fonction des paramètres donnés, une fois la fonction en question compilée puis exécutée. Si le nombre de paramètres n'est pas le bon, ou si l'objet retourné ne correspond pas au type attendu dans l'entête de la fonction donné dans le champ de texte en haut, rien n'est affiché dans la fenêtre mais l'erreur peut être constatée sur la sortie standard ou dans le fichier de logs selon ce qui est précisé dans le script de log4j. Il est indiqué :

```
[java] 2008-01-03 16:37:57,999 [AWT-EventQueue-0] ERROR : compilation failed
```

Si l'utilisateur est un étudiant le bouton de droite est nommé « proposer pour validation » et le changement est indiqué à l'enseignant comme dit (bouton éditer : rouge). Si l'utilisateur est un

enseignant, le bouton de droite s'appelle « Valider toutes les modifications », ce qui ré-enregistre tous les attributs dans la base de données mais il n'est plus indiqué qu'il y a eu des nouveautés à propos de cet algorithme.

## IX Fiche Génie Logiciel

ce script a été élaboré pour le decompote des lignes de code :

```
#!/bin/bash
echo "nombre de lignes en tout : "
tr "\t" "\n" < $1 | tr -s "\n" | tr -s "\t" | wc -l
echo "nombre de lignes de code : "
cat $1 | sed 's/!^.*\*/!!' | sed '/^\/,^*V/d' | sed '/^ \*V/d' | sed '/^ \*/d' | wc -l
echo "nombre de lignes de commentaires : "
expr `tr "\t" "\n" < $1 | tr -s "\n" | tr -s "\t" | wc -l` - `cat $1 | sed 's/!^.*\*/!!' | sed '/^\/,^*V/d' | sed '/^ \*V/d' | sed '/^ \*/d' | wc -l`
```

<i>Étapes</i>	<i>Nombre de lignes effectives</i>	<i>Temps passé (h)</i>
<b>Analyse</b>		3
<b>Conception</b>		2
<b>Codage</b>		
AlgoDynamicCompiler	182	0
Algorithm	167	4
AlgosListing	126	2
IsRelatedToProgram	93	2
Messages	17	0
StudentKnowledge	36	3
TestsAlgos	31	1
AlgosFrame	136	4
AlgosTab	147	4
MainFrameStudent	185	4
StudentsTab	280	3
MainFrameTeacher	31	2
Messages	17	0
TestsGUI.java	22	1
DescriptionTab	116	3
LoginDialog	98	1
Main	53	1
PropositionTab	85	4
CheckID	67	3

<i>Étapes</i>	<i>Nombre de lignes effectives</i>	<i>Temps passé (h)</i>
Student	96	3
TestsUsers	20	1
<b>Mise au point</b>		
AlgoDynamicCompiler		12
Algorithm		3
AlgosListing		1
IsRelatedToProgram		4
Messages		1
StudentKnowledge		2
TestsAlgos		1
AlgosFrame		2
AlgosTab		1
MainFrameStudent		3
StudentsTab		6
MainFrameTeacher		3
Messages		1
TestsGUI.java		1
DescriptionTab		3
LoginDialog		3
Main		3
PropositionTab		2
CheckID		3
Student		1
TestsUsers		0
<b>Tests sur IHM</b>		2
Commentaires	45+92+16+31+8+27+19+13+25+18 +7+25+12+19+65+6+19+21+13 = 481	8
Rédaction CR		8

Totaux : 2005 125

Ratio de productivité 16,04

\* les « lignes de codes effectives » sont les nombres de ligne du fichier, lignes de commentaires une fois soustraites.

# **X Bilan**

Ce projet a été fini avec le regret de ne pas avoir pu fournir un programme complètement abouti même si c'était l'idée de départ ; même si les éléments restant en chantier ne restent que des détails, il est bien plus satisfaisant d'avoir le temps de les achever. Néanmoins, le sujet et le concept de projet ont permis de découvrir idéalement le plus d'éléments de la programmation Java possibles.