Rapport de Travail d'Etude et de Recherche

InfoLycée

Par BENARAB Rachid
étudiant M1 ifi & mbds

Encadré par : VIEVILLE Thierry et

BEN ABDALLAH Hamdi

Mai 2010

Table des matières

Introduction

A l'ère numérique dans laquelle nous vivons, les fondements de l'informatique constituent un savoir indispensable comme peut l'être la lecture ou l'écriture, pouvoir enseigner l'informatique à nos jeunes au lycée apparaît comme une nécessité du fait de la place de cette discipline aussi bien dans notre économie et dans notre société que parmi les outils qui nous permettent de comprendre le monde.

L'enseignement de l'informatique au lycée devrait avoir comme but principal l'apprentissage de la programmation et de l'algorithmique. L'algorithmique a une place naturelle dans tous les champs des mathématiques, dans toutes les autres disciplines, et dans la vie courante. On appelle algorithme la méthode ou la façon systématique de procéder pour faire quelque chose : trier des objets, situer des villes sur une carte, multiplier deux nombres, extraire une racine carrée, chercher un mot dans le dictionnaire...

Il est indispensable d'apprendre l'algorithmique (*et les fondements théoriques qui y conduisent*), c'est le levier pour pouvoir comprendre et maîtriser, voire adapter les logiciels que nous utilisons (*et pas uniquement les subir !*). L'apprentissage de l'algorithmique apporte beaucoup aux lycéennes et lycéens dans leur développement intellectuel, car il permet un travail par projets et demande de mettre en application des connaissances acquises, et également parcequ'il permet de construire un pont entre le langage et l'action et montre l'utilité de la rigueur scientifique.

Il est impossible de s'approprier ce savoir sans une vraie pratique, et une démarche expérimentale, il faut donc incarner cette algorithmique en faisant de la programmation (*Apprentissage des méthodes par la pratique expérimentale*). Le but de ce TER est d'inventer un mécanisme le plus ludique possible, le plus créatif, et le plus attrayant pour que des lycéens, ou des étudiants en L1, s'initient à ces notions. À l'occasion de l'écriture d'algorithmes et de petits programmes, il convient de donner aux élèves de bonnes habitudes de rigueur et de les entraîner aux pratiques systématiques de vérification et de contrôle.

Contexte du travail

2.1 Objectifs pour le lycée

La démarche algorithmique est, depuis les origines, une composante essentielle de l'activité mathématique. Au collège, les élèves ont rencontré des algorithmes (algorithmes opératoires, algorithme des différences, algorithme d'Euclide, algorithmes de construction en géométrie). Ce qui est proposé dans le programme des lycées est une formalisation en langage naturel avec traduction sur une calculatrice ou à l'aide d'un logiciel. Il s'agit de familiariser les élèves avec les grands principes d'organisation d'un algorithme : gestion des entrées-sorties, affectation d'une valeur et mise en forme d'un calcul, en opérant essentiellement sur des nombres entiers.
Dans le cadre de cette activité algorithmique, les élèves sont entraînés :

- à décrire certains algorithmes en langage naturel ou dans un langage symbolique ;
- à en réaliser quelques uns à l'aide d'un tableur ou d'un petit programme réalisé sur une calculatrice ou avec un logiciel adapté ;
- à interpréter des algorithmes plus au moins complexes.

2.2 JavaSCool

L'INRIA pilote une initiative de communication scientifique vers les lycées et participe au niveau national à l'introduction de l'apprentissage de l'informatique au lycée. Ce TER est ancré dans un contact avec les lycées de l'académie. L'objectif est d'aider à l'enseignement scolaire et citoyen des fondements de l'informatique et des sciences de l'information.

L'informatique s'ouvre vers les lycées avec le logiciel Javascool, à l'initiative de l'INRIA Javascool est conçu pour l'apprentissage au sein des lycées français des bases de la programmation. Il a été conçu à la demande des professeurs de lycées. Il permet de manipuler un Macro-Langage de programmation, basé sur le langage JAVA. Maintenant il est disponible pour la plateforme **Windows** avec la version Java'S Cool 2.0 **Beta.**

Mais dans quel langage programmons nous ? Avec Javascool nous sommes entrain d'apprendre à programmer dans un langage commun de point de vu syntaxe aux langages Java, C ou C++, et même Javascript ou PHP. La bonne nouvelle est que ce que nous apprenons ici, servira donc ensuite quel que soit le langage de programmation : ce seront les mêmes mécanismes, juste exprimés parfois un peu différemment.

Pourquoi appeler ce langage commun 'Java'sCool' ? Parce que le langage principal "caché" derrière notre outil est le Java qui a été un peu simplifié ici pour le rendre "cool". D'ailleurs il est possible d'utiliser tout le langage Java ici. Simplement, un certain nombre de spécifications sont devenues optionnelles en Javascool pour faciliter les choses aux débutants. C'est donc une école de Java avec laquelle Java *is cool*. Cette idée a été mise au point par des étudiants en informatique des années précédentes de l'Université de Nice Sophia-Antipolis.

2.3 Les ingrédients des algorithmes

Depuis la rentrée 2009 le module d'Algorithmique et programmation en mathématiques est introduit aux lycées. Cette initiation utilise Java'sCool, cette forme simplifiée du langage Java ne fait appel à aucune particularité de ce langage, et les programmes que nous écrirons pourront être traduits sans peine dans un autre langage.
Le logiciel Java'sCool contient Plusieurs proglets qui seront détaillé plus loin dans ce rapport, ce parcours permet de s'approprier les cinq ingrédients des algorithmes (en utilisant la *«proglet»* `Konsol`).

Donc Java'sCool nous conduit pas à pas à :

- définir des séquences d'instructions,
- se servir des variables,
- découvrir l'instruction conditionnelle,
- se familiariser avec les fonctions,
- utiliser des boucles.

L'éditeur JavaScool permet d'insérer des lignes de programme en les choisissant dans une liste, il colorise automatiquement le texte de l'algorithme pour mieux distinguer les différents mots du langage.

2.4 Quand le professeur apprend avec l'élève

L'informatique se prête à une expérience pédagogie des plus intéressantes pour l'élève (et enrichissante pour l'enseignant). Le Professeur de Mathématiques ou de Sciences Physiques se lève devant sa classe et dit : « je ne sais (que peu ou) pas encore programmer moi-même, mais je sais comment vous apprendre à le faire, et comment apprendre avec vous ». Cette position pédagogique a été vécue plusieurs fois en 2007-2009 avec un succès complet et récurrent : élèves responsabilisés, co-apprentissage coopératif entre élèves, professeur partageant une démarche de recherche au delà de la simple transmission de savoir, plaisir partagé de la réussite, etc. Que personne ne se trompe : il ne s'agit pas de "niveler les rôles" ! Le professeur reste le référent, détenteur non pas d'un simple lot de connaissances, mais de quelque chose de plus profond : une démarche méthodologique.
En pratique, les "profs" que nous connaissons ont joué ce rôle très simplement, de manière magistrale, en tout cas avec la même discipline que tout autre cours. Les élèves ont profondément apprécié cette démarche, sans être dérouté par son aspect singulier. L'une d'elle en tirera cette belle conclusion « Oh, il est possible de tout apprendre alors ! ». C'était pour le Professeur, un des plus beaux remerciements.

Analyse critique de l’existant

3.1 PROGLETS

Une `proglet` est une petite applette qui permet de s'initier à la programmation de manière ludique, puisque les petits programmes créés animent un élément visuel ou sonore, à la fois pour mieux comprendre ces objets numériques et pour offrir une démarche expérimentale concrète, vers la compréhension des notions abstraites qui forment le fondement des sciences de l'information. Ces `proglets` sont des exemples de «`gamelet`» (c'est à dire de grains pédagogiques interactifs pour un apprentissage ludique d'une notion).

Ces `proglets` ne sont pas conçus pour apprendre la programmation en "autodidacte", un débutant risque d’être bloqué sur des détails de syntaxe ou d'implémentation, et il faut impérativement quelqu’un de votre entourage qui sait programmer et qui peut vous aider à suivre le parcours pour s'initier à la programmation.

1. Plusieurs proglets sont disponibles pour s'initier à la programmation et créer des exercices animés et interactifs
    - La *«proglet»* `Konsol`,
        - utilisée pour les apprentissages initiaux des ingrédients des algorithmes et
        - tous les exercices qui nécessitent des lectures au clavier comme dans une «console».
    - La *«proglet»* `Scope`,
        - utilisée pour les exercices qui nécessitent un tracé de courbes où un petit affichage graphique.

- La «*proglet*» `Smiley`,
  - utilisée pour les exercices qui nécessitent de manipuler des images ou des graphismes 2D et une activité spécifique.
- La «*proglet*» `Dicho`, utilisée pour une activité spécifique (recherche dichotomique).
- La «*proglet*» `Conva`, utilisée pour une activité spécifique (conversion analogique-numérique).
- La «*proglet*» `Tortue`, simule la «bonne vieille» tortue logo, donc permet de faire beaucoup d'autres activités.
- La «*proglet*» `Synthe`, est utilisée pour des activités (plutôt d'un club de maths ou de science, ou lors d'un TPE) de découverte du signal sonore.

2. Chaque proglet est un code source libre, programmée en Java et documenté pour sa libre redistribution, avec un accès au code source, et permettre des développements dérivés, un enseignant qui connait un peu de programmation peut facilement développer ces propre proglet's.

- **Ecrire notre premier programme**: Voici un exemple de programme minimal qui affiche la chaine de caractère "HelloWorld":

```
void main() {

   println("Hello World !");

}
```

- **Compiler et exécuter notre premier programme**: Comment faire exécuter ce programme minimal par l'ordinateur ? Il y a trois étapes.
  1.  D'abord toujours ... sauver le texte source du programme dans un fichier d'extension `.jvs`, par exemple `Hi1.jvs`,
  2.  Puis . . compiler le programme: celà veut dire le traduire dans le langage de l'ordinateur.
     - En effet, l'ordinateur est une machine électronique et le "langage qu'il comprend" est extrêmement basique (ouvrir tel interrupteur, connecter telle partie du circuit à une autre, etc..) et tout ceci est bien trop fastidieux à décrire directement (l'action **println**, par exemple, comporte elle-même des milliers d'opérations élémentaires !).

- Alors, nous disposons d'un langage *formel de haut niveau* celui que nous découvrons ici et que nous appelons `Java'sCool` et qui permet de spécifier des instructions de la manière la plus efficace possible.

Compiler signifie alors *traduire le langage `Java'sCool` en langage machine*.

Noter que le compilateur (le traducteur de JavaScool en langage machine) est prévu pour renseigner au maximum sur l'erreur rencontrée, le numéro de ligne est fourni, le message d'erreur essaye de décrire le problème, etc.

3.  Enfin ... exécuter le programme : cela veut dire demander à l'ordinateur de partir de **void main** et d'exécuter toutes les instructions données. Dans notre cas, nous obtiendrons l'écriture de **Hello World !**... et nous aurons en main la base pour faire des choses bien plus intéressantes et plus sophistiquées !

3.2 AlgoBox

A l'aide d'un mini-langage algorithmique en français et d'une interface simple et ergonomique, ce logiciel permet de concevoir et de tester facilement les algorithmes que l'on peut rencontrer dans l'enseignement des mathématiques au secondaire.

Le code des algorithmes se construit pas à pas à l'aide de commandes prêtes à l'emploi (lire, afficher, affecter, instruction si...alors, boucles pour...de...à et tant que...) : cela permet au débutant de se concentrer principalement sur la logique algorithmique plutôt que sur l'apprentissage d'une syntaxe complexe. Une fois l'algorithme mis au point, il peut facilement être exécuté afin d'en vérifier la validité. De la même façon, AlgoBox peut aussi servir à la compréhension d'algorithmes déjà écrits.

Afin de pouvoir approcher les situations mathématiques diverses que l'on peut rencontrer au lycée, AlgoBox inclut aussi la possibilité d'utiliser une fonction numérique ainsi que de tracer des points et des segments dans un repère préalablement défini.

Par contre, AlgoBox ne permet pas de définir des fonctions au sens informatique du terme, ni de faire de la récursivité. Ce programme n'est donc pas un environnement de développement complet au sens classique du terme : il est avant tout un logiciel pédagogique d'initiation à l'algorithmique simple d'emploi et adapté à l'esprit des nouveaux programmes de mathématiques au lycée.

AlgoBox est un logiciel libre, multi-plateforme et gratuit, sa conception doit beaucoup à la qualité et à la modernité de l'environnement de développement libre Qt, il s'installe très facilement et peut-être exécuté sur une simple clé USB, il a les caractéristiques suivantes :

- Le code de l'algorithme se construit pas à pas **de façon hiérarchique et structurée** grâce à des instructions de base que l'on insère en cliquant sur les boutons de l'interface : l'utilisateur se concentre ainsi sur l'algorithme lui-même et il est fortement incité, par le fonctionnement même du programme, à faire preuve d'un minimum de rigueur.
  Dans son mode de fonctionnement par défaut, AlgoBox n'utilise pas un éditeur de texte classique et **une grande partie du code est inséré automatiquement** : l'activité de l'utilisateur consiste donc plus à réfléchir aux instructions qu'il doit utiliser qu'à aligner des lignes de code.

Pour les algorithmes longs, **un mode "éditeur de texte" permet aux utilisateurs avancés de taper directement le code à l'aide d'un éditeur incorporé** muni des fonctions traditionnelles (coloration syntaxique, auto-complétion, chercher-remplacer, etc...)

- **Tous les algorithmes** élaborés dans la fenêtre principale du programme **peuvent être exécutés et testés en deux clics**.
- AlgoBox permet aussi l'exécution d'algorithmes en mode **"pas à pas pédagogique"**.
- **Des exemples d'algorithme sont fournis** avec le programme.
- En plus d'une sauvegarde classique, le code l'algorithme peut être imprimé et exporté sous forme de fichier texte.
- L'algorithme peut aussi être exporté sous la forme d'une page web autonome dans laquelle il est possible de l'exécuter (cette page web étant en fait une copie conforme de la fenêtre de test du programme). **Les algorithmes élaborés peuvent donc être très facilement inclus dans des sites internet**.
- Après exécution d'un algorithme, **l'ensemble des résultats (sorties, graphique et code de l'algorithme) peut être exporté dans un document au format pdf**.

Aussi AlgoBox fonctionne **de façon totalement autonome** et ne nécessite aucune installation complexe d'environnement de programmation.**Le programme existe pour les systèmes Linux, MacOsX et Windows et peut-être installé facilement sur les postes personnels des élèves et des professeurs : il ne nécessite l'installation d'aucun programme supplémentaire.**

3.3 Pourquoi ne pas utiliser algobox plutôt que JavaScool ?

Le logiciel algobox permet de concevoir et de tester facilement les algorithmes que l'on peut rencontrer dans l'enseignement des mathématiques au secondaire. Il est plus facile à prendre en main que JavaScool car il évite de se soucier de la syntaxe et permet, d'insérer les instructions à la main sans les taper. Il est de plus très bien adapté aux algorithmes de mathématiques de secondes, avec beaucoup d'exercices prêts à être utilisés. Les auteurs d'algobox expliquent qu'il a été conçu pour appréhender de suite les structures logiques algorithmiques fondamentales, en choisissant d'introduire un nouveau pseudo-code et mini-langage algorithmique. Et c'est un logiciel libre, multi-plateformes, ce qui est précieux. En revanche, il a trois inconvénients:

- Du fait qu'il ne confronte pas l'élève à écrire du code, il ne permet pas d'appréhender la *syntaxe* des structures abstraites qui forment les ingrédients des algorithmes. C'est une simplification, donc une limitation pédagogique, au sens où en utilisant un éditeur graphique l'élève est moins incité à s'approprier les abstractions à apprendre.
- Il reste limité à des interactions de type console (lire/écrire une valeur) et l'affichage de courbe et ne permet pas de manipuler des objets numériques.
- Il «réinvente» un nouveau langage, et une interaction spécifique avec le système, ce qui va nécessiter de la part de l'élève un apprentissage provisoire supplémentaire.

Application

4.1 Choix technique

Le but est de proposer un mécanisme le plus ludique possible, le plus créatif, et le plus attirant pour que des lycéens, s'initient aux bases de la programmation sans se préoccuper des problèmes de syntaxe qui sont avec Javascool, ce nouveau parcours d'apprentissage ne sera pas basé sur les «ingrédients» des algorithmes, mais sur des outils facile à fort potentiel didactiques.

Vu les caractéristiques et les avantages pédagogiques d'Algobox, on a trouvé indispensable d'intégrer dans Javascool les avantages pédagogiques d'Algobox. Le principe est de faire un **mini**-langage **algorithmique** ("pseudo-code") qui sera simple à comprendre et à utiliser (**les instructions sont en français)**. Les élèves n'ont pas à apprendre toute la syntaxe de Javascool pour pouvoir écrire un Algorithme, ils seront inities à l'algorithmique avec notre nouveau outil qui les conduit pas à pas vers l'utilisation des différentes proglets.

Nous allons utiliser essentiellement la bibliothèque Swing de Java, ce choix est imposé dans ce projet pour éviter les problèmes d'intégration dans Javascool. Donc c'est la programmation événementielle de java, par là, entendez programmation d'interface graphique avec le langage Java au contraire d'Agobox qui est entièrement en C++.

Interface Principale

La fenêtre principale hérite de l'interface JFrame :
A ce niveau juste on peut créer la fenêtre principale, et choisir sa taille

```
public class Algotree extends Jframe {....}
```

Les composants utilisés doivent interagir avec l'utilisateur et doivent être bien paramétré pour qu'ils fonctionnent à notre convenance, les plus utiliser sont les Jtree, les boutons, les zones de texte, les cases à cocher...

L’arbre
On utilise l’objet JTree permet d'afficher une hiérarchie de **DefaultMutableTreeNode :**

```
Private Jtree arbre;
```

Ce composant affichant/éditant des données sous forme hiérarchique permet la description hiérarchique de données comme dans algobox, mais encore plus facile puisque tous les élèves ont déjà vu ce genre d’arbre pour explorer des dossiers dans des fenêtres qui se présente sous la même forme que notre Arbre juste ils auront à faire à une version plus riche avec des boutons actionnable à tout moment.

Ici Le code de l'algorithme se construit pas à pas **de façon hiérarchique et structurée** grâce à des instructions de base que l'on insère en cliquant sur les boutons de l'interface : l'utilisateur se concentre ainsi sur l'algorithme lui-même. On n'utilise pas un éditeur de texte classique et **une grande partie du code est inséré automatiquement** : l'activité de l'utilisateur consiste donc plus à réfléchir aux instructions qu'il doit utiliser qu'à aligner des lignes de code.

Le JTree doit être éditable :

```
arbre.setEditable(true) ;
```

Le modèle des noeuds :

Tous les noeuds sont des objets DefaultMutableTreeNode en commençant par la racine :

```
private DefaultMutableTreeNode racine ;
```

Un nœud de l'arbre peut avoir au plus un parent et 0 ou plusieurs enfants.

On récupère le nœud sélectionné avec la méthode :

```
arbre.getLastSelectedPathComponent() ;
```

On peut récupérer le contenu d'un nœud avec la méthode **toString()** qui retourne la représentation de la chaîne de son objet utilisateur

Le modèle d'arbre :

```
private DefaultTreeModel model ;
```

Après dimensionnement, positionnement, et titrage de la fenêtre on invoque la méthode buildTree() pour construire notre arbre qui aura la forme suivante :

4.2 Gestion des évènements

Dans cette partie je vais expliquer les différentes actions sur les boutons, j'explique aussi les difficultés rencontré et les solutions apportées aux problèmes

4.2.1 **Déclarer une variable**

Dans la partie déclaration on appuis sur le bouton «**DECLARER UNE VARIABLES»,** avant tout j'ai implémenter un ActionListener pour interagir avec les cliques sur ce bouton, on utilisant la fonction :

```
declarer.addActionListener(new ActionLitener(){ …}) ;
```

La méthode pour gérer l'action appliquée est :

```
Public void actionPerformed(ActionEvent evnt ){...}
```

La première chose à gérer est qu'un noeud doit être sélectionné pour qu'on puisse ajouter un enfant :

```
If(arbre.getLastSelectedPathComponent()!=null){...}
```

Le problème qui apparaît à ce niveau est qu'on peut déclarer une variable en sélectionnant n'importe quel noeud de notre arbre, plus précisément dans la partie INSTRUCTION ou en sélectionnant FIN_DU_PROGRAMME, la solution apporté est de testé le nœud sélectionné :
-Si l'utilisateur veut déclarer une variable en sélectionnant un nœud autre que «VARIABLES »

```
if( !(arbre.getLastSelectedPathComponent().tostring() ==”VARIABLES”) )
```

Alors une boite de dialogue informatif s'affiche pour inviter l'utilisateur à sélectionner VARIABLES d'abord, voici le code:

```
JOptionPane jop1 = new JoptionPane();
Jop1.showMessageDialogue(null,”Veuillez selectionner VARIABLES”, ”Erreur”,
JOptionPane.ERROR_ERROR) ;
```

-Si le nom du nœud sélectionné est «VARIABLES » alors une boite de dialogue s’affiche avec un champs pour saisir le nom de la variable à déclarer.

```
JOptionPane jop = new JoptionPane();

String NomdeVariable = jop.showInputDialog ("Saisir le nom de variable");
```

Apres que l’utilisateur à choisi le nom de la variable une nouvelle fenêtre s’affiche pour choisir le type de la variable

```
String[ ] TypeDeVariable = {"EST_CHAINE_DE_CARACTERE",
"EST_ENTIER_NATUREL","EST_NOMBRE_REEL"} ;

String TypeDeVariable = (String) jop. showInputDialog(null,

"Veuillez choisir le type de variable","Nouvelle Ligne !",

JOptionPane.QUESTION_MESSAGE, null, TypeDeVariable,
TypeDeVariable[2]) ;
```

Ensuite le nœud est ajouter comme fils au nœud père « VARIABLES », son nom sera le nom de la variable saisi concaténer au type de variable choisi :

```
String NomduNoeud = NomDeVariable+" : " + TypeDeVariable ;

if(NomduNoeud != null && !NomduNoeud.trim().equals("")){

                                        DefaultMutableTreeNode parentNode =
(DefaultMutableTreeNode)arbre.getLastSelectedPathComponent();

                                        DefaultMutableTreeNode childNode = new
DefaultMutableTreeNode(NomduNoeud);

                                        parentNode.add(childNode);

                                        model.insertNodeInto(childNode,
parentNode, parentNode.getChildCount()-1);

                                        model.nodeChanged(parentNode);
```

Enfin notre fait bien le travail de déclaration de variable qui se fait seulement en sélectionnant le nœud **VARIABLES**

4.2.2 Ajouter une instruction :

Dans cette partie déclaration on va programmer le bouton « **AJOUTER UNE NSTRUCTION»**, c'est la même démarche et la même implémentation que pour le bouton précèdent «**DECLARER UNE ARIABLES»**.
La différence est qu'on peut ajouter une instruction on sélectionnant le nœud INSTRUCTIONS ou bien un de ses fils (éventuellement petit fils) donc on est obligé de suivre le chemin du nœud sélectionné jusqu'à la racine de l'arbre pour vérifier que le nœud IINSTRUCTIONS est présent dans le chemin de notre nœud sélectionné vers la racine, les lignes de code pour gérer tout ça sont les suivants :

```
  // on récupère le noeud sélectionné
                                    DefaultMutableTreeNode lenoeud
=(DefaultMutableTreeNode)arbre.getLastSelectedPathComponent();
  // pour afficher le chemin de  ce noeud vers la racine de l'arbre
boolean nouvelleligne= false;
String s = lenoeud .toString();
if(lenoeud .toString()=="INSTRUCTIONS") nouvelleligne= true;

while(!lenoeud.isRoot()) {
lenoeud = (DefaultMutableTreeNode)lenoeud.getParent();
if(lenoeud .toString()== "INSTRUCTIONS")nouvelleligne= true;

                                    }
```

Si le noeud INSTRUCTIONS ne se présente pas dans le chemin du nœud sélectionné vers la racine alors une boite de dialogue préventif s'affiche pour inviter l'utilisateur à aller dans la partie INSTRUCTIONS ;

```
if(!nouvelleligne) {
JOptionPane jop2 = new JOptionPane();
Jop3.showMessageDialog(null, "Aller dans INSTRUCTIONS", "Erreur",
JOptionPane.ERROR_MESSAGE);
                                    }
```

-Si le nom du nœud sélectionné est «INSTRUCTIONS » alors boite de dialogue s'affiche pour choisir le type de l'instruction.

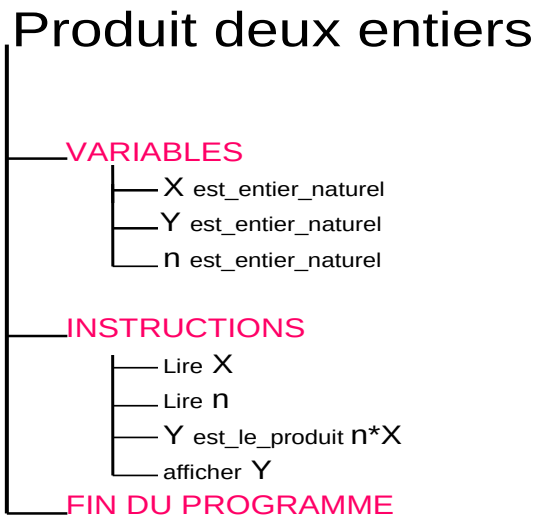
```
JOptionPane jop = new JOptionPane();

String[] action = {"si_alors_sinon", "pour_finpour", "lire_entier", "lire_chaine_char"… };
```

Le traitement du choix diffère D'une instruction choisi à une autre, par exemple si "égale" est choisi une autre fenêtre se lance pour saisir les deux valeurs pour lesquels on va tester l'égalité, pour l'instruction "pour_finpour" deux nœuds fils seront crée pour le nœud sélectionné, le premier pour le début de la boucle pour et l'autre pour la fin de boucle, maintenant on va étudier l'exemple de l'instruction "si_alors_sinon" quand l'utilisateur choisi cette instruction, trois noeuds fils seront créer pour notre nœuds sélectionné (nœud1= "SI", nœud2= "ALORS", nœud3= "SINON") .

```
if(nodeName == "si_alors_sinon"){
DefaultMutableTreeNode childNode1 = new DefaultMutableTreeNode("SI");
parentNode.add(childNode1);
model.insertNodeInto(childNode1, parentNode, parentNode.getChildCount()-1);
model.nodeChanged(parentNode);
DefaultMutableTreeNode childNode2 = new DefaultMutableTreeNode("ALORS");
parentNode.add(childNode2);
model.insertNodeInto(childNode2, parentNode, parentNode.getChildCount()-1);
model.nodeChanged(parentNode);
DefaultMutableTreeNode childNode3 = new DefaultMutableTreeNode("SINON");
parentNode.add(childNode3);
model.insertNodeInto(childNode3, parentNode, parentNode.getChildCount()-1);
model.nodeChanged(parentNode);
                                                    }
```

Voici un exemple de programme construit :



4.2.3 EFFACER UN NŒUD :

Dans cette partie on va programmer le bouton «SUPPRIMER», on doit pas supprimer les trois nœuds principales ("VARIABLES", "INSTRUCTIONS", "FINDUPROGRAMME"), donc on teste si le nœud sélectionne est l'un des trois :

```
 if(((arbre.getLastSelectedPathComponent().toString() ==
"VARIABLES")||(arbre.getLastSelectedPathComponent().toString() ==
"INSTRUCTIONS")||(arbre.getLastSelectedPathComponent().toString() == "FIN
DU PROGRAMME") ))
```

Dans ce cas on affiche une fenêtre avec le message d'erreur "Impossible de supprimer".

```
JOptionPane jop3 = new JOptionPane();
jop3.showMessageDialog(null, "Impossible de supprimer", "Erreur",
JOptionPane.ERROR_MESSAGE);
```

On utilise pour supprimer un nœud la méthode removeNodeFromParent(node), les lignes suivantes expliquent le processus de suppression d'un nœud:

```
DefaultMutableTreeNode node =
(DefaultMutableTreeNode)arbre.getLastSelectedPathComponent();
DefaultMutableTreeNode parentNode =  (DefaultMutableTreeNode)node.getParent();
model.removeNodeFromParent(node);
model.nodeChanged(parentNode);
```

4.2.4 ENREGISTREMENT D'UN ARBRE :

On avait la possibilité d'enregistrer l'arbre sous forme XML en utilisant la classe TreetoXML qui permet d'enregistrer l'arborescence d'un JTree composé de noeud DefaultMutableTreeNode dans un fichier XML grâce aux plus simples et maniables des API pour XML, JDOM.
Aussi on peut utiliser plus simplement XMLEncoder pour sauvegarder et charger les JTree.
Ces deux possibilités ont été écarté puisque notre but n'est pas d'enseigné xml, donc finalement c'était décidé avec les encadreurs d'essayé d'enregistré notre arbre dans un fichier sous forme de code Javascool, pour donner aux élèves la possibilité de voir le code Javascool équivalent aux arbres construites par leurs soins.
La première chose à faire est de permettre à l'utilisateur de naviguer dans le système de fichiers pour choisir l'endroit de sauvegarde du programme et le nom du programme, j'ai utilisé pour ça l'API JFileChooser, ce qui permet de fournir à l'utilisateur (lorsqu'il clique sur le bouton Sauvegardé) une interface graphique pour naviguer dans le système de fichiers, puis de choisir soit un fichier ou répertoire à partir d'une liste ou d'entrer le nom d'un fichier ou d'un répertoire, les lignes de code pour afficher un sélecteur de fichiers sont :

```
JFileChooser save = new JFileChooser();
save.setDialogTitle("Enregister un programme");
save.setDialogType(JFileChooser.SAVE_DIALOG);
save.setApproveButtonText("Enregister");
int value = save.showSaveDialog(null);

if (value == 0) {
 try {
String check = save.getSelectedFile().getPath();
setMainFile(check);
     } catch(Exception e1)
}
```

L'extension de Javascool (.jvs) est ajouter automatiquement, le problème rencontré à ce niveau est que l'extension n'est pas ajouter si on l'ajoute au nom de fichier, donc il faut l'ajouter au path du fichier, aussi le fichier doit être ouvert en écriture par un FileWriter ;

```
BufferedWriter out = new BufferedWriter(new FileWriter(path+".jvs"));
```

On récupère la liste des nœuds avec la method getAllNodes(arbre).
Ensuite on commence à écrire dans notre fichier à la Javascool par la ligne :
void main(){

```
out.write("void main() {");
```

On définie un *Iterator* sur la liste des nœuds pour écrire notre programme suivant les noms des nœuds, par exemple pour les variables on devise notre string suivant les deux points « : » (puisque les variables dans notre arbre sont sous la forme « x : EST_ENTIER_NATUREL ».

```
while(declarationodename!="INSTRUCTIONS"){
String f [] = declarationodename.split (":") ;
String h = f[0];
String g = f[1] ;
if(g=="est-un-entier"){
       out.write("int "+h+"=readInteger()");
       nodesliste.remove(declarationode);
}
if(g=="est-chaine-char"){…}
if(g=="est-un-entier"){…}
```

Pour l'instruction SI_ALORS_SINON le traitement est le suivant :

```
if (nodename=="SI"){
                    out.write("if(");
}
if (nodename=="ALORS"){
                    out.write("){");
}
if (nodename=="SINON"){
                    out.write("}else{");
}
```

Qu'on arrive a la fin du programme on ferme la parenthèse de la partie instructions puis la parenthèse de notre main et on ferme notre fichier ;

```
if (nodename=="FIN-DU-PROGRAMME") {
                                out.write("}");
                                 out.write("}");
                                 out.close();
                        }
```

4.2.5 Ouverture d'un programme

On cliquant sur le bouton "OUVRIR" situer au menu de notre fenêtre, comme dans la partie précédente une interface graphique s'affiche pour naviguer dans le système de fichiers, et choisir le fichier qui doit avoir l'extension d'un programme Javascool (.jvs).

Pour construire notre arbre on a utilisé un FileReader pour lire les mots de notre programme et construire notre arbre en ajoutant des noeuds à notre arbre.

4.2.6 APPEL DE FONCTION

On donne la possibilité à l'utilisateur d'appeler une fonction lorsqu'il sélectionne un noeud et clique sur le bouton « APPELLER UNE FONCTION », il choisi le nom de la fonction dans un sélecteur de fichier, la fonction est un fichier d'extension Javascool (.jvs).
Ici on va construire un sous arbre en partant du noeud sélectionné, on lit le code de la fonction et on construit notre sous arbre avec la partie instruction de la fonction, ensuite on donne la possibilité à l'élève de changé les noms et les valeurs des variables utilisé dans le corps de notre sous arbre qui représente la fonction appelé.

4.2.7 COMPILER ET EXECUTER UN PROGRAMME

Reviens à compiler et exécuter le code Javascool enregistré puisque notre outil sera intégré dans le logiciel Javascool.

PERSPECTIVES

Notre outil de programmation peut être amélioré facilement, il est bien documenté, et sera intègre dans le logiciel Javascool pour lequel plusieurs instanciations sont proposées pour travailler en classe selon la configuration informatique :

1. [luxe] Le logiciel Java'sCool dans sa forme complète avec un éditeur coloré, des menus adaptés, un interface Orphy, etc. (nécessite une installation spécifique sur chaque machine utilisée).
2. [facile] L'environnement de programmation simplifié pour utiliser les proglet's d'un simple clic (nécessite une simple installation standard de Java).
3. [sans réseau] L'environnement de programmation simplifié et sa documentation se télécharge sous forme d'un fichier [ZIP] :
   - Il suffit en suite de le copier sur une clé USB puis d'extraire le dossier org.javascool.proglet sur chaque machine,
   - Et d'ouvrir org.javascool.proglet/index.html avec un navigateur.

   (nécessite une installation standard de Java sans accès au réseau pendant le travail en classe).

4. [web] L'environnement de programmation simplifié pour utiliser les proglet's en web service (nécessite une installation standard de Java et un accès au réseau pendant le travail en classe). A n'utiliser que pour des essais.

Remerciement

Mes vives remerciement pour mes deux encadrants et tout particulièrement Mr VIEVILLE Theirry qui été toujours disponible et à l'écoute de mes questions, aussi il m'a aidé tout au long de ce TER avec ses conseil et ses éclaircissement sinon j'aurais pas pu faire tout ce travail avec la réduction de l'effectif.
Je remercie également mes collègues étudiants qui m'on fait part de leurs réflexion et expérience avec les outils de développement que j'ai utilisé pendant mon TER.

Sans oublier bien sure tous les enseignants de lycées pour leur précieuse collaboration.

Références

http://www.fuscia.info

http://interstices.info

http://javascool.gforge.inria.fr/proglet/doc/about-all.htm#parcours

http://javascool.gforge.inria.fr/proglet/doc/about-autres.htm

http://www.unilim.fr/irem/fileadmin/documents/Documents_stages/Manuel_AlgoBox.pdf

http://www.javafr.com/codes/ENREGISTRER-ARBORESCENCE-JTREE-DANS-XML-AVEC-JDOM_49042.aspx

http://www.developpez.net/forums/d528520/java/interfaces-graphiques-java/awt-swing/composants/arbres/jtree-recuperer-noeuds/

http://java.sun.com/docs/books/tutorial/uiswing/components/filechooser.html

http://perso.telecom-paristech.fr/~charon/coursJava/fichiersEtSaisies/index.html

http://www.siteduzero.com/tutoriel-3-65570-les-arbres.html