Java Learning Machine

un exerciseur pour la programmation Java et python

Martin.Quinson@loria.fr
Maître de conférences à l'université de Lorraine
(Loria, Telecom Nancy, Inria)

Journée EPI ISN de Nancy
27 juin 2013

Motivations pour JLM

Mon expérience d'enseignement :

- 1999-2004 : Introductions à CAML, au C, aux réseaux ; AlgoProg ; CS∅ (en C)
- 2005 : ESIAL : école d'ingénieur formant des spécialistes de l'informatique
  - 3A : Programmation répartie ; AlgoDist avancée (Grilles, P2P)
  - 2A : Programmation système ; OS
  - 1A : CS∅ (en Java) ; Introduction à l'admin système ; C/Shell

Constats

- Niveau de programmation des élèves pas génial en 3A/2A

$\rightsquigarrow$ Les cours d'intro sont plus utiles (et plus intéressants)

- Seuls ceux qui savent avant savent après (problèmes avec *if* en janvier de 1A !)

Comment aider ceux qui en ont besoin ?

- Énormes différences de niveaux en entrée : ceux qui savent trustent la parole
- Expliquer les bases de base : aussi difficile qu'enseigner à faire du vélo
- *Si seulement ils pouvaient bosser par eux-mêmes. . .*
- *. . . si seulement nos TP pouvaient être moins arides et plus attirants*

La génèse de JLM

Printemps 2008

- Introduction 2 semaines bloquées de “remise à niveau info” en entrée
- Forme choisie (avec G. Oster) : travail pseudo-autonome mais tuteuré

Les Buggles (F. Turbak – Wellesley College)

- ☺ Des exercices interactifs simples et relativement ludiques pour CS0
- ☺ Innovations pédagogiques sympa : recursion first, multiples micro-mondes
- ☹ Les élèves complètent le code complet (risque de noyade ?)
- ☹ Pas très modulaire, peu de partage de code entre les exos
- ☹ C’est encore le prof qui valide l’exercice ; ☹ Énoncé pas intégré à l’exercice

Été 2008

- Réimplémentation frénétique de la Java Learning Machine
- Mise en place d’une séquence pédagogique adaptée à notre contexte

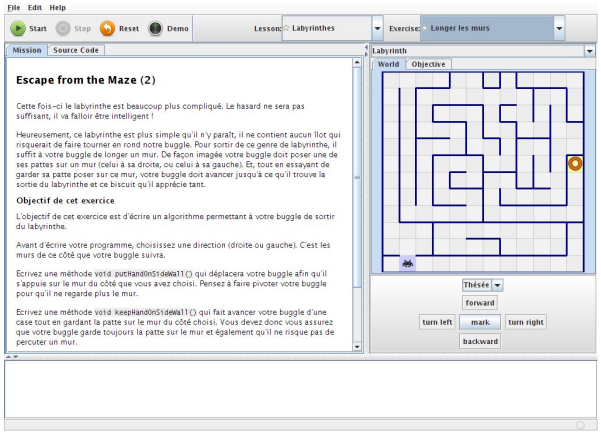
Automne 2008

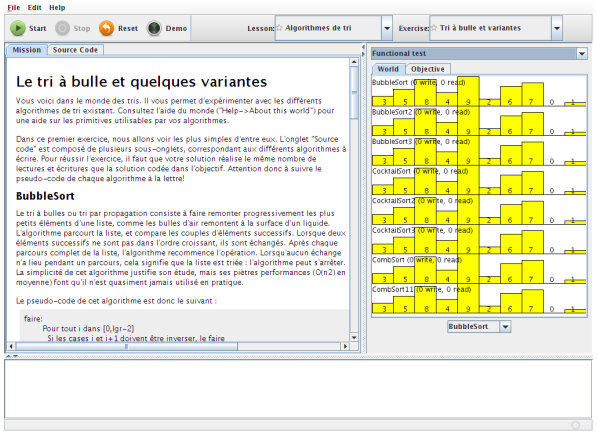
- C’est un succès : plus d’incompréhension douloureuses, au moins
- Refonte complète de JLM pour ajouter d’autres micro-mondes

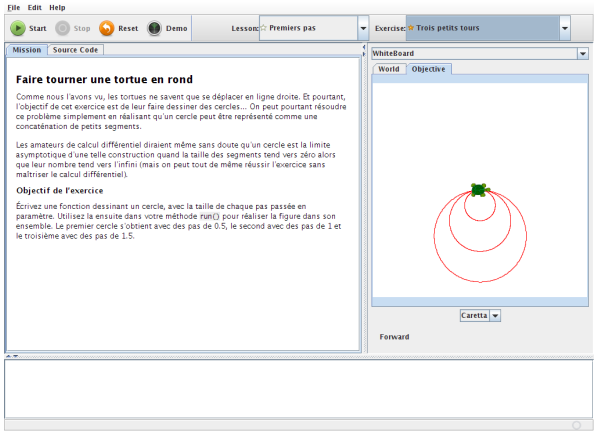
Java Learning Machine aujourd'hui

Exerciseur interactif dédié à la programmation

- C'est en forgeant qu'on devient forgeron (et qu'on apprend à aimer ça)
- Outil interactif et graphique pour apprendre la programmation (Java, Python)







Comment on l'utilise

- On lit la mission à gauche, on compare à droite l'état initial et l'état désiré
- On tape le code, on clique sur un bouton, et *ça s'anime* à droite
- Boucle de feedback très courte (et motivante)
- Mode démo, exécution pas-à-pas, gestion des sessions, etc.

Première démonstration

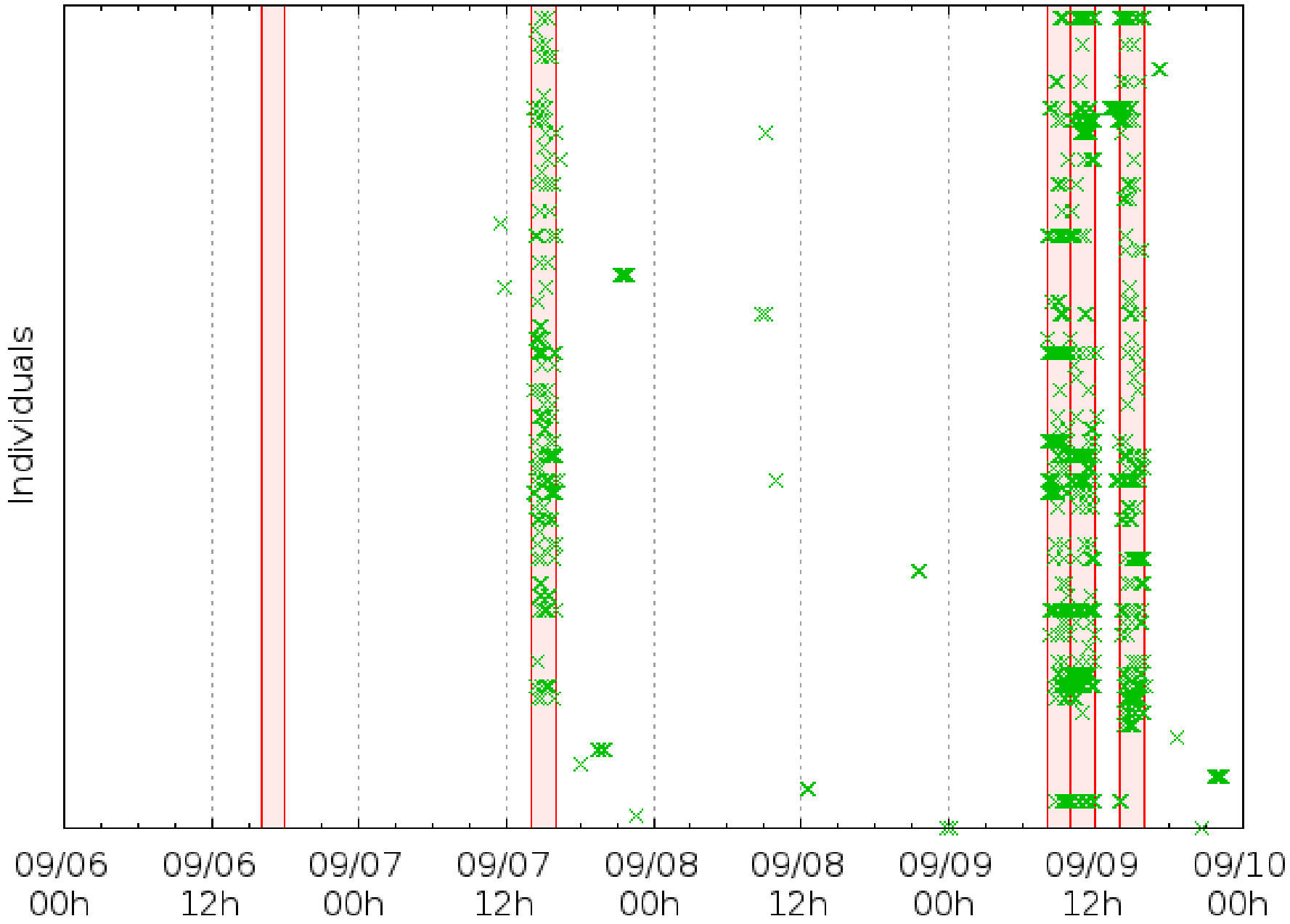
Éléments remarquables

- L'interface est bilingue anglais/français
- Les exercices sont bilingues Java/python
- Vitesse d'animation variable (exo *Méthode/Dessiner/Plus grand*)
- Le même code utilisé pour tous les buggles (exo *Instructions conditionnelles*)
- Parfois il y a plusieurs mondes (exo *Méthodes avec résultats*)

L'univers des buggles

- Bouger : avancer, tourner, reculer. Getter/setter de position et de direction
- Buggle : getter/setter de couleur, de couleur de brosse, lever/baisser
- Interactions : Lire la couleur de la case ; Lire / écrire des chaînes par terre
- Baggles : Des biscuits qu'on trouve par terre, et qu'on peut ramasser
- Murs : tester leur présence ; ils sont infranchissables
- Tout est expliqué dans *Aide/À propos de ce monde*

Nos élèves travaillent par eux-mêmes



Démonstrations des univers

L'univers des tortues

- Moins d'interaction avec le monde, mais virages au degré près
- Avance, recule, tourne, lève/baisse crayon, getter/setter d'angle et position
- C'est le logo classique, toujours efficace pour la récursivité

L'univers des tris

- Un tableau, auquel on accède par les builtins uniquement
- Décompte exact des accès et comparaisons $\rightsquigarrow$ identification de l'algo de tri
- Une vue temporelle pour voir (et comparer) la chronologie de l'algorithme

Les autres univers

- Bat : tests simples de fonctions, à la jUnit
- Hanoï : exemple à suivre pour ceux qui veulent implémenter un univers
- Sciences Manuelles du Numérique : implémenté par un stagiaire 2A en 1 mois
- Lightbot : jeu / casse tête (proof of concept de la généricité de JLM :)

Notre progression pédagogique en CS0

Objectif : tactical programming

- Le B-A-BA de la programmation : écrire des trucs de base sans difficulté
- Que ce ne soit plus un problème quand les algos seront non-triviaux

Implémentation language-agnostic (Java, python, soon Scala)

- Découverte de JLM
- Instructions (et commentaires)
- Conditionnelles (+40 exercices Bat pour savoir écrire des tests logiques)
- Boucles while ; switch cases (+ 3 exos d'applications)
- Les variables
- Boucles for et do/while
- Méthodes (+5 exos qui forcent à factoriser le code)
- Méthodes avec résultat, avec paramètres
- Tableaux (+12 exos d'applications, certains en Bat)
- Application : parcours de tableau 2D (5 exos) les turmites (turing 2D – 4 exos)

Comment rajouter un univers

Trois classes à dériver

- World : contient les données. Sait se dupliquer et se comparer.
- Entity : interface de contrôle du monde (adapte le code métier du monde) le code des élèves est placé dans une entité pour s'exécuter
- WorldView : représentation graphique du monde
- (WorldPanel : boutons interactifs pour contrôler l'entité dans l'interface)
- (Exercise : séparé normalement, mais parfois dérivé pour adapter JLM)

Complexité des mondes existants

- Buggle : 19 classes, 2000 lignes (dont un tiers pour l'éditeur de mondes)
- Turtle : 4 classes, 700 lignes
- Sort : 8 classes, 900 lignes
- Lightbot : 8 classes, 1300 lignes (dont l'éditeur de code graphique)
- Bat : 5 classes, 500 lignes
- Hanoi : 5 classes, 300 lignes

Écrire un exercice

Deux classes, et un fichier HTML

- *Exercise* : Instancie et peuple tous les mondes-test
- *Exercise*.html : Explique ce qu'il faut faire
- *Exercise*Entity : Fait ce qu'il faut faire
- (*Exercise*.fr.html : Explique ce qu'il faut faire en français)

C'est très simple

- Rares sont les exos de plus de 50 lignes
- Il n'y a pas grand chose à configurer, mais les noms sont normalisés
- On peut faire plus (exos de Turmites dérivent BuggleWorld pour factoriser du code)

Ça va devenir encore plus simple

- Mode "édition" en travaux, mais pas fini
- L'éditeur graphique de mondes Buggle est déjà fonctionnel

Écrire une leçon est encore plus simple

Une classe lesson.xxx.Main

- Elle contient une seule méthode *void loadExercises()*

```
addExercise(new Environment(this));
Lecture basics = addExercise(new Basics(this));

Lecture conditions = addExercise(new Conditions(this));
Lecture loopWhile = addExercise(new LoopWhile(this));
```

Un fichier HTML décrivant la leçon

- Il devrait expliquer les attendus pédagogiques
- On peut le traduire, bien sûr

Empaquetage du tout

- JLM sait charger des jar files séparés
- Le contenu doit suivre une structure assez simple (et mal documentée)

C'est même documenté dans JLM directement

	Instructions	Comments	Conditionals	While loops	Variables	For loops	Do/While loops	Methods	Switch	Arrays
Welcome in the Buggles' World										
Java Instructions										
Writing more complex programs										
Conditional instructions										
While loops										
Baggle Seeking										
Storing and manipulating data										
For loops										
Do .. while loops										
Methods										
	Instructions	Comments	Conditionals	While loops	Variables	For loops	Do/While loops	Methods	Switch	Arrays
Building methodically										
Methods returning a result										
Methods with parameters										
Methodically drawing										
Methodically drawing (only bigger)										
Drawing bigger and bigger										
Even more pattern to draw										
Buggle Dance Revolution										
Buggle Dance Revolution 2										
Slug Hunting										
Slug Tracking										
	Instructions	Comments	Conditionals	While loops	Variables	For loops	Do/While loops	Methods	Switch	Arrays
Snake World										
Knitting and Arrays										
Knitting, Arrays and modulos										
Traversal by column										
Traversal by line										
Zig-zag traversal										
Diagonal Traversal										

Malheureusement, ceci n'est pas généré automatiquement (pas toujours à jour)

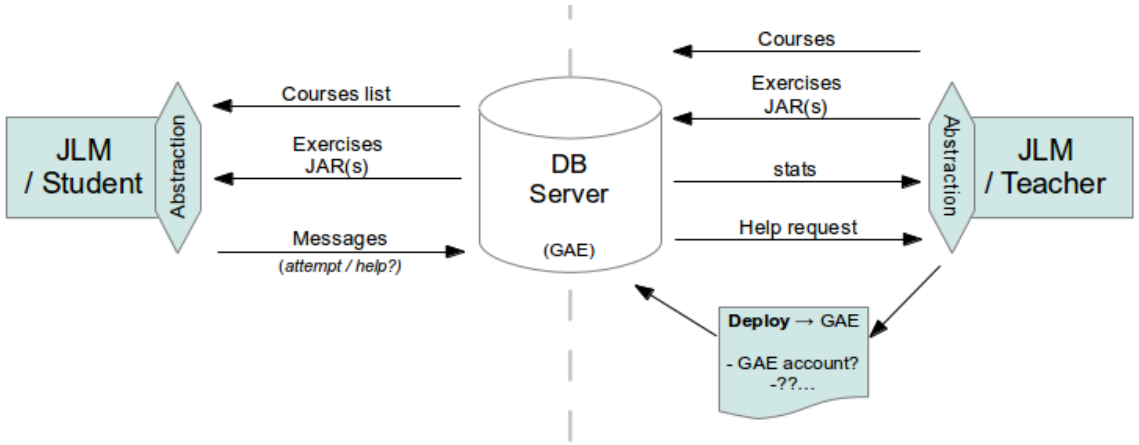
JLM et le métier d'enseignant

«Qui a besoin de moi ? ?»

- JLM libère le temps d'aider ceux en difficulté, mais on ne sait plus qui c'est

JLM espionne vos élèves

- Post identi.ca et twitter
- Les élèves peuvent cliquer sur un bouton pour lever la main
- Chaque tentative et ses résultats peuvent être envoyé vers votre serveur



Une console pour l'enseignant

- Ces informations sont affichées pour choisir qui aller aider
- Je n'ai pas encore testé en production ce code d'étudiants
- Beaucoup de travail restant sur le filtrage pour sélectionner qui aller voir

Point de vue d'enseignants

Usages à Télécom Nancy et ailleurs

- Utilisé dans 2 modules (remise à niveau et AlgoProg), malgré ma délégation
- Utilisé pour le stage LIESSE (profs prepa, 3j)
- 150 exos scénarisés en leçons cohérentes, des dizaines d'heures d'usage

- Beaucoup d'intérêt, quelques usages disparates chez les collègues
- Mines Nancy l'ont utilisé et ont arrêté (pb technique $\rightsquigarrow$ cata ; Java $\rightsquigarrow$ python)

Adaptabilité

- Ajouter des exos assez simple, mais nous sommes les seuls à l'avoir fait :-/
- Un mode "édition" est en travaux (papier à musique des exo de prog)

Avant de l'utiliser chez vous

- Utiliser un tel outil demande un **gros travail de préparation**
  Tester en salle de TP, faire les exos, anticiper les questions
- Un plan B en cellulose est raisonnable la première fois
- Mais une fois que c'est en place, c'est vraiment plaisant

Conclusions

JLM veut aider trois publics

- Élèves : apprennent à leur rythme dans un “jeu” sérieux
- Enseignants : pratique de la programmation perdre leur liberté éditoriale
- Auteurs : réutilisation du code non-fonctionnel, retour par instrumentation

Encore pleins d'idées

- De nouveaux univers : backtracking, POO, système, PatchWorld, Robbozzle
- De nouveaux langages : Scala, tous les JSR231 et même le langage C
- Beaucoup de polishing : visuel, fonctionnel et de documentation
  - une erreur de compilation est un lien vers la ligne indiquée, marquée en rouge
  - Beaucoup d'améliorations possible sur le game-play

Join us !

- Get it ! `http://www.loria.fr/~quinson/Teaching/JLM/`
- Hack it ! `http://github.com/oster/JLM`
- Licence GPL, j'espère que vous allez vous l'approprier et participer