Université de Nice - Sophia Antipolis
Faculté des Sciences

DEUG MIAS MP1

Programmation 2001-02

1. PREMIER RENDEZ-VOUS AVEC DRJAVA

A. Le langage de programmation JAVA et l'environnement DRJAVA

Java est un langage de programmation normalisé. Le mot *Java* est une marque déposée par la firme *Sun Microsystems* [`http://java.sun.com`] qui contrôle le développement du langage, avec l'aide des principaux acteurs du marché [IBM, Hewlett-Packard, etc]. Ce que l'on nomme *Java 2* débute en réalité avec la version 1.2 du langage [vous travaillez au MIPS avec la version 1.3].

Les outils de développement Java [compilateur, run-time] sont inclus dans le JDK [*Java Development Kit*] distribué par Sun pour Windows ou Linux. Sur Macintosh sous Mac OS-X [un Unix avec interface graphique], Java 2 est fourni en standard par Apple avec tous les outils Unix.

DrJava est – comme *RealJ* - un *environnement de programmation* [en anglais IDE : *Integrated Development Environment*]. Il permet de programmer en Java, gratuitement, et de manière très simple. Nous vous renvoyons au document le concernant qui vous a été distribué par ailleurs. Cette feuille va vous mettre cet IDE en main.

B. Le travail en JAVA au second semestre MP1

Seules des séances en salles-machines sont prévues, avec des feuilles d'auto-formation contenant des compléments sur le langage, sur les techniques de programmation par objets, et des exercices associés. Tous les exercices doivent avoir été compris et programmés pour les examens. Ne prenez pas de retard, vos enseignants attendent de vous un travail régulier, correspondant au niveau que vous souhaitez atteindre. Sachez qu'à l'aube de ce nouveau millénaire, avec une informatisation toujours plus grande de la société, un[e] jeune scientifique ne sachant pas ou mal programmer sera *un[e] handicapé[e]*, ne vous faites aucune illusion là-dessus ! Enfin, comme chaque année, nombre d'entre vous souhaiteront bifurquer vers l'informatique à la fin de la seconde année [filière universitaire ou école d'ingénieur]. Ne négligez pas cette possibilité, laissez-vous le maximum de portes ouvertes, beaucoup de vocations sont tardives et le marché de l'emploi a ses réalités...

Au premier semestre, vous avez appris les rudiments du langage et les premiers algorithmes, notamment sur les tableaux. Au second semestre, vous allez approfondir ce travail préliminaire vers les objets, et le compléter par un apprentissage des interfaces graphiques et de types de données plus sophistiqués.

La page Internet associée à ce cours se trouve sur `http://deptinfo.unice.fr/~roy`, bouton `MP1`. Consultez-la régulièrement, vous y trouverez comment installer *DrJava* chez vous le cas échéant [Windows, Macintosh, Linux], ainsi que les feuilles de TP et des corrections. La consultation de ces dernières fait partie intégrante de cet enseignement. Votre travail ne se limitera donc pas aux séances de TP : les feuilles doivent être préparées, si possible sur machine, *avant* les séances, ou prolongées *après* les séances. Il faut qu'il soit bien clair que le seul travail en séance de TP ne donnera que peu de résultats en l'absence d'un cours en amphi, et l'on n'apprend pas à programmer en dilettante…

C. Organisation du travail

Sur votre disquette[1], créez un répertoire `A:\Java\MP1` de manière à ne pas vous mélanger avec les fichiers du Tronc Commun qui sont dans `A:\Java`. Durant le travail en salles–machines, tous vos fichiers seront placés dans le dossier `A:\Java\MP1`. Pour compiler plus vite, il est aussi possible de travailler en créant un nouveau répertoire dans `E:\Etudiant` sans oublier à la fin de la séance de copier tous les fichiers de ce répertoire dans `A:\Java\MP1`.

• Pour lancer DrJava, ouvrez « *Documents de cours* » en haut du menu *Démarrer*, puis ouvrez le dossier `mp1`. Vous lancez DrJava en double-cliquant sur le fichier `drjava.cmd`. Une fois que DrJava est là, vous fermez la fenêtre « *Documents de cours* » ainsi que la vilaine fenêtre noire MS-DOS. Cette installation est - espérons-le - provisoire…

D. Première consultation avec DrJava : le « toplevel »

Vous avez donc lancé DrJava. Une fenêtre surgit, divisée en trois panneaux.

- le panneau 1 contient la liste des fichiers en cours d'édition.
- le panneau 2 est l'éditeur.
- le panneau 3 est le « toplevel » interactif.

Oublions pour l'instant les panneaux 1 et 2. Ce qui fait l'originalité [et même l'exclusivité] de *DrJava* est le panneau 3, le « toplevel » interactif [*DrJava* le nomme « *Interactions* ». Connu aussi sous le nom de « *shell* » en Unix, il s'agit d'un lieu de dialogue entre vous et *DrJava*. Vous allez pouvoir, à l'invite du caractère `'>'`, dire quelque chose au docteur. Ce quelque chose doit tenir en une seule ligne et peut être :

- une *déclaration* terminée par un point-virgule, par exemple :

```
> int x = 10 ;
```

- une *instruction* terminée par un point-virgule, par exemple :

```
> x = x + 1 ;                ou bien          > System.out.println("x = " + x) ;
```

- une *expression* [arithmétique, booléenne, etc] sans point-virgule, par exemple :

```
> Math.sqrt(x)               ou bien          > x == 10
```

Lorsque vous déclarez au toplevel que x est un entier initialisé à 10, ceci est mémorisé par Java pour les lignes suivantes. Vous pouvez modifier la valeur de x, mais pas revenir sur le fait que c'est une variable entière, donc interdit de déclarer deux fois une variable [tout comme dans une méthode d'ailleurs] !

> Pour travailler de manière conséquente au **toplevel**, il faut **l'agrandir** au maximum. Utilisez pour cela les minuscules flèches (notamment celle de gauche) se trouvant à l'extrémité gauche de la frontière horizontale entre la zone 3 et les zones 1-2.

Exercice 1.1 **a)** Mettez-vous au toplevel. Déclarez que x est un *double* initialisé à 0.
b) Sans utiliser aucun « `print` », demandez quelle est la valeur de `x`.
c) Modifiez la valeur de `x` en lui donnant la valeur 10, en terminant par un point-virgule.
d) Vérifiez que `x` a bien changé de valeur.
e) Demandez si `x ≥ 0`.
f) Modifiez la valeur de `x` en la multipliant par 2, sans point-virgule à la fin ! *Vous voyez qu'une affectation est en réalité une expression, et que le point-virgule la transforme en instruction sans résultat.* Néammoins la valeur de `x` a bien changé, vérifiez-le !
g) De manière générale, quelle est la valeur d'une affectation « `x = <expr>` » où `<expr>` est une expression ?…
h) Déclarez que `y` et `z` sont des entiers.
i) Comment comprenez-vous l'instruction « `y = z = 3 ;` » au vu de ce qui précède ?
j) Que pensez-vous de l'instruction « `x + 3 ;` » ? Provoquera-t-elle une erreur à la compilation ?

[1] Tâchez d'avoir une disquette de sauvegarde, et de copier une fois par semaine le contenu de la disquette courante sur la disquette de sauvegarde. Certains lecteurs mangent les disquettes ☺…

MORALE : Bien distinguer en Java une **expression** d'une **instruction** !

Exercice 1.2 **a)** Essayez un « `System.out.println("x vaut " + x)` » avec ou sans point-virgule. Idem avec `print` au lieu de `println`. Regardez bien comment réagit le toplevel à toutes ces variantes.
b) Plutôt qu'exécuter la méthode `System.out.print(…)`, demandez donc la valeur de l'expression « `"x = " + x` »

MORALE : Au toplevel, les instructions d'**entrées/sorties** sont le plus souvent **inutiles** !

Exercice 1.3 Vous allez effectuer des calculs plus complets au toplevel.
a) Commencez par « nettoyer » le toplevel en demandant ***Reset interactions*** dans le menu *Edit*. Cette opération non seulement nettoie la fenêtre toplevel, mais fait oublier à *DrJava* toutes les variables que vous avez utilisées jusqu'à présent.
b) Faites afficher au toplevel [*rappel : sans « `print` »*] une valeur approchée de la somme :

$$s = \sum_{n=1}^{+\infty} \frac{1}{n^2}$$

avec une boucle `for`. [Réponse : 1.6439…]. Soyez raisonnable dans votre approche de $+\infty$ …
c) Copiez-collez toutes vos instructions pour faire ce calcul sur une seule ligne, déclarations et affichage du résultat compris !
d) Comparez la valeur trouvée avec celle de $\pi^2/6$. *Pour une démonstration, demandez à votre prof d'Analyse* ☺…

Pour **sauvegarder le contenu du toplevel** [quitte ensuite à supprimer des lignes, ajouter des commentaires, etc], il suffit de le copier-coller dans l'éditeur, puis de demander « *Save As* » dans le menu *File*. N'utilisez pas une extension `.java` puisque ce n'est pas une classe, nous vous proposons de le nommer `calcul-serie.top` [`top` pour toplevel]… Evitez les accents. La sauvegarde effectuée, vous demandez « *Close* » dans le menu *File*.

Exercice 1.4 Nettoyez le toplevel !
a) A l'aide d'une boucle `while`, calculez au toplevel le plus petit diviseur u ≥ 2 de l'entier n = 67898839.
b) Faites le même calcul avec une boucle `for`.
c) Faites le même calcul avec une boucle `do`.
d) Faites le même calcul avec une boucle « infinie » `while (true)...` *[utilisez une instruction `break`].*
e) Le nombre u est-il premier ? Pourquoi ?
f) Que vaut v = n/u ? Prouvez au toplevel que v est premier.
g) Complétez la ligne suivante au toplevel *[tout sur une seule ligne !]* pour définir dynamiquement une nouvelle petite méthode `ppdiv(…)` retournant le plus petit diviseur ≥ 2 d'un entier n ≥ 2.

```
> static int ppdiv(int n) { ......... }
```

Utilisez-la pour retrouver la valeur de u. Si votre méthode fonctionne mal, corrigez-la le cas échéant *[oui, une nouvelle définition écrase l'ancienne !]*. Retrouvez de même la valeur de v.
h) Sauvez le toplevel dans un fichier `ppdiv.top`. Vous enjoliverez après le TP ce fichier sous Microsoft Word® par exemple pour l'imprimer proprement.

Exercice 1.5 Nettoyez le toplevel !
a) Importez la classe `unsa.Turtle` puis définissez deux tortues nommées `euler` et `cauchy`.
b) Disposez les trois fenêtres de la manière suivante : les deux fenêtres `Turtle0` et `Turtle1` en haut et celle de *DrJava* en bas [elle ne contient toujours que le toplevel].
c) Jouez interactivement avec les deux tortues en leur envoyant quelques messages : avance, tourne, dessine un carré [boucle `for`], dessine un petit cercle [boucle `for`], etc.

ATTENTION : si vous nettoyez le toplevel après avoir créées les tortues, il sera bien ré-initialisé, mais les fenêtres tortue resteront vivantes et inaccessibles puisque les variables `euler` et `cauchy` auront été oubliées! Il est possible avec un `System.exit(0)` au toplevel de provoquer une ré-initialisation plus profonde. Il vaut mieux demander « Quitter » dans l'une des fenêtres tortue…

E . Allez, on retourne en classe !

Les exemples qui précèdent montrent l'extraordinaire bénéfice de disposer d'un toplevel pour faire de petits calculs sans avoir à écrire de grosses classes en attendant constamment de fastidieuses compilations. Il se trouve que le travail du programmeur occasionnel [un scientifique en particulier] se satisfait souvent d'un tel travail interactif. Mais si les méthodes à écrire deviennent plus lourdes [plusieurs lignes] et le programme plus complexe, il devient indispensable de programmer une ou plusieurs classes complètes, avec ou sans `main(…)`.

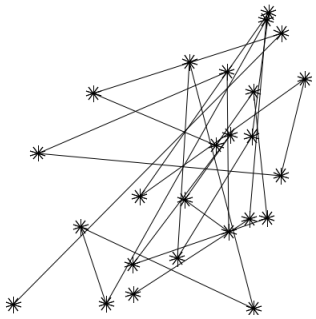
Exercice 1.6 **a)** Avec Explorer, allez sur la page Internet du cours, à la séance n°1. Vous trouverez un lien sur un fichier `Etoile.java`. Récupérez ce fichier sur votre disquette, dans le dossier `A:\Java\MP1` comme il se doit.
b) Ouvrez ce fichier dans *DrJava* en demandant « Open » dans le menu File. Notez que le constructeur de cette classe est pauvrement indenté. Pour le ré-indenter proprement, sélectionnez à la souris les lignes concernées puis pressez la touche de tabulation [à gauche du A]. Il s'agit donc d'une touche d'indentation, elle ne produit plus une tabulation !
c) Dès qu'un fichier est modifié, une étoile apparaît en haut à côté de son nom et le bouton « Compile » est désactivé. Sauvez la version bien indentée, puis compilez le fichier. S'il y a une petite erreur, corrigez-la et recompilez…
d) Last compilation completed successfully. Good ! Mettez-vous au toplevel en cliquant sur l'onglet *Interactions*. Définissez une variable `star` de type `Etoile`, avec 10 rayons de longueur 10. Demandez-lui de se dessiner avec une nouvelle tortue, au point (-50,0) par exemple. Ne cliquez pas sur le bouton « Quitter ».
e) Quelle est la valeur de la variable `star` au toplevel ?
f) Pas très beau, n'est-ce pas ?... Ajoutez à la classe `Etoile` une méthode `toString(…)` pour avoir une valeur plus agréable à regarder.
g) Recompilez ! Hélas, le toplevel est nettoyé et ses variables perdues [c'est d'ailleurs très bien, car sinon ce serait une belle pagaille !]. Il y a toujours le moyen d'avoir un fichier `.top` dans lequel on copie le toplevel pour faire du copier-coller ultérieurement. Mais en réalité, nous sommes mûrs pour un bon et brave `main(…)`.
h) Plutôt que rajouter un `main(…)` à la classe `Etoile`, vous allez écrire une seconde classe `TestEtoile` dans le même fichier [*oui, on peut écrire plusieurs classes dans un seul fichier*]. Cette seconde classe ne contiendra qu'un `main(…)` définissant l'étoile `star`, une tortue `lea`, et demandant à `star` de se dessiner avec `lea` au point `(-50,0)`. Et tant que vous y êtes, faites afficher la valeur de la variable `star`. Recompilez…
i) Ah, il s'agit maintenant d'exécuter le `main(…)` de `TestEtoile`. Pour cela, vous allez utiliser le toplevel en mode « ligne de commande » bien connue des pratiquants d'Unix. La phrase magique[2] est :

```
> java TestEtoile
```

j) Remplacez le dessin unique de la `star` par 25 dessins de cette même demoiselle à des endroits aléatoires de la fenêtre de la tortue `lea`. *Vous n'utiliserez pas `Numerik.randomInt(…)`, d'ailleurs il ne sera plus question de `Numerik`. Retrouvez la bonne formule en utilisant directement `Math.random()` qui retourne un double dans `[0,1[`. On rappelle qu'une fenêtre tortue fait 400 × 400.*

Et on s'arrêtera là pour aujourd'hui…

P.S. Quoi, il vous reste du temps ? Dans la figure précédente, reliez donc les étoiles tracées par un segment :



[2] *Une phrase magique équivalente pour exécuter cette méthode de classe, et plus puriste, serait « TestEtoile.main(null); » où « null » représente une référence inexistante vers un tableau de chaînes. Mais nous aurons l'occasion de reparler de cet objet « null » plus tard peut-être…*