

TD1₁

Environnement de développement

1 Préalable

Le principal objectif des *ateliers logiciels* de deuxième et troisième est de compléter l'apprentissage du langage *Java*.²

D'autre part, en 1^{re} année, votre apprentissage de *Java* s'est accompli dans un environnement de programmation rudimentaire, dans le but pédagogique, entre autre, de ne rien vous masquer des étapes entre l'édition d'un source et l'exécution du programme résultant. Il existe cependant des environnement de développement intégrés, EDI (*Integrated Development Environment*, IDE) facilitant la vie du développeur.

Un second objectif de ce TD est de vous guider lors de vos premiers pas dans l'EDI choisi, à savoir NetBeans (cf. <http://www.netbeans.org>)³

2 Prise en main de NetBeans

Netbeans est un EDI adapté à *Java*, gratuit et *open source* (initialement développé par SUN). Nous allons l'utiliser dans le cadre des *Ateliers Logiciels* (ALG2ir). Notez cependant tout de suite que nous ne l'emploierons pas dans toute sa puissance. Ainsi, par exemple, lorsque nous développerons des applications à interface graphique, nous ne recourrons pas à l'outil graphique de développement offert par NetBeans ! Nous y reviendrons par la suite...

¹Le texte de ce TD s'inspire très largement de ceux des ateliers logiciels écrits jadis par MCD et VAK auxquels ont collaborés par la suite RFS, SMB et NVS. Je (PBT) laisse mon empreinte et en profite pour remercier tous ceux qui aiment être remerciés.

²Remarquez que la version de *Java* disponible actuellement à l'école sur les machines MS-WINDOWS 2000 est la version 1.6. C'est cette version que nous utiliserons dans le cadre de ce cours.

³Remarquez que la version de *NetBeans* disponible actuellement à l'école est la version 5.5 mais vous pouvez tout aussi bien utiliser la version 6.0.

2.1 Apprentissage

On trouve sur le site de Netbeans, un guide de prise en main de NetBeans 5.5. Nous vous encourageons vivement à suivre ce tutoriel⁴ afin d'être en confiance avec cet environnement.

Notez cependant qu'il est loin de faire le tour de toutes les facilités offertes par l'EDI. N'hésitez pas à consulter la documentation et à essayer les possibilités de débogage, refactoring, etc.

2.2 Avantages et inconvénients du l'utilisation d'un EDI

Comme vous venez de le constater, NetBeans offre de multiples services très intéressants tels que la coloration contextuelle, l'auto-complétion, la recherche dans la documentation Javadoc, le débogage, etc.

Dans le cadre d'un usage pédagogique, certaines des aides proposées par NetBeans vont un peu *trop loin*. Par exemple, la génération de code par NetBeans (hormis les modèles, *templates*) ne sera pas exploitée dans le cadre des ALG2ir. Ainsi, pour mettre en page une application à interface graphique ou pour associer l'exécution d'une méthode à l'occurrence d'un événement, nous ne recourrons pas aux interfaces graphiques et sorciers (*wizards*) de NetBeans, mais coderons tout à la main...

Un autre point faible, pédagogiquement parlant, lors du développement à l'aide de NetBeans, est le masquage du `PATH` et du `CLASSPATH` ainsi que la gestion automatique de la correspondance entre le paquetage (*package*) d'une classe et l'emplacement dans l'arborescence des dossiers du fichier où cette classe est définie. Gardez à l'esprit que vous devez toujours être capable de :

- ↪ définir correctement les variables d'environnement `PATH` et `CLASSPATH` ;
- ↪ sauver vos sources au bon endroit (cf. `CLASSPATH` et paquetage) ;
- ↪ compiler vos sources ;
- ↪ exécuter vos byte codes ;

hors NetBeans, via la ligne de commande, comme on vous l'a appris en 1^{re} année !

3 Exercices

Lors de la résolution de tous les exercices de ce TD et de ceux à venir, veillez à ne pas travailler hors paquetage, mais définissez proprement une arborescence de paquetages personnelle et propre au cours ALG2ir. Normalement vous devriez définir des paquetages de la forme

be.heb.esi.gxxxxx.td01.exercice1 ce qui peut devenir un peu *lourd*

⁴ Accessible depuis le menu de l'EDI : Menu Help >> Option Quick Start Guide.

dans le cadre de ces exercices. Pour ma part vous pouvez vous contenter de quelque chose de la forme `gxxxxx.monbelexercice`.

Exercice 1

Écrivez une fonction qui crée un triangle de Pascal de n lignes puis une fonction qui l'affiche. Écrivez enfin une fonction principale pour tester le tout.

Le paramètre n est récupéré par la ligne de commande. Le programme affiche un petit message de type `usage : . . .` si il est appelé avec un mauvais nombre d'arguments ou avec un argument non entier.

Pour rappel, le triangle de Pascal, P , de n lignes est défini par la récurrence :

$$P(i, 1) = P(i, i) = 1 \quad \text{pour } i = 1, 2, \dots, n,$$

$$P(i, j) = P(i - 1, j - 1) + P(i - 1, j) \quad \text{pour } 1 < j < i.$$

Exercice 2

Récupérez dans `eDistri/nvs` le programme « Répertoire », c'est-à-dire les trois fichiers `Personne.java`, `Répertoire.java` et `TestRépertoire.java`, et servez-vous en pour créer un projet NetBeans.

Les sources fournies ont été développés *avant* le JDK 1.5. Avant toute autre chose, mettez-les à jour !

La documentation de la méthode `add` de la classe `Répertoire` indique :

TODO : L'ajout ne se fait que si la personne n'y est pas encore.

Réalisez ce désir.

Ce problème étant réglé, modifiez ce qui doit l'être de sorte à pouvoir associer plusieurs numéros de téléphone à une même personne.

Veillez à utiliser au mieux les possibilités du langage.

Exercice 3

Écrivez une application « poullailler », manipulant des poules. Vous écrirez une classe (`Poule` qui représentera une poule⁵, une classe `TasDeGraines` et une classe `Main` permettant de tout mettre en oeuvre (voir FIG 1).

La classe `Poule` représente une poule et possède les attributs⁶,

- ↪ `nextId`, représentant l'identifiant de la prochaine poule, c'est un attribut de classe,
- ↪ `id`, c'est l'identifiant de la poule,
- ↪ `âge`, c'est l'âge de la poule, il s'incrémente de 1 à chaque tour (on simulera le temps qui passe dans une boucle de la classe `Main`).

⁵ Cet animal qui dandine de la tête et qui pond des oeufs.

⁶ Je ne parle pas des constantes que vous définirez comme renseignées sur le diagramme.

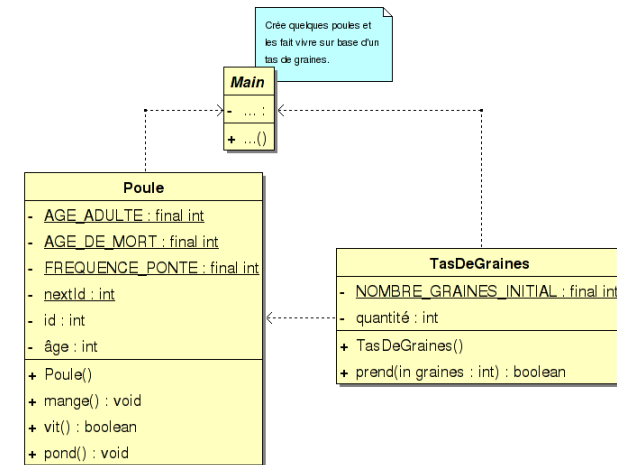


FIG. 1 – Diagramme UML

Hormis son constructeur, elle possède les méthodes,

- ↪ `mange` qui donne quelques graines à la poule. Cette méthode retourne `true` si la poule a mangé le nombre de graines désirés, `false` sinon.
- ↪ `vit` qui précise si la poule est vivante. Une poule reste vivante si elle peut manger le nombre de graines qu'elle veut et que son âge ne dépasse pas son âge de mort, sinon, elle meurt.
- ↪ `pond` qui permet de faire pondre la poule. Dans ce cas, elle retourne une liste de nouvelles poules.

La classe `TasDeGraines` représente le tas de graines que les poules vont manger. Cette classe possède un attribut `quantité` représentant le nombre de graines. Il est initialisé aléatoirement.

Hormis son constructeur, la classe possède la méthode `prend` qui retourne `true` s'il y a suffisamment de graines dans le tas, et le met à jour, et retourne `false`, sinon.

La classe `Main` permet de tester le « poullailler » sur base d'une liste de poules et d'un tas de graines.