

TD1

Environnement de développement

1 Préalable

L'objectif principal des ateliers logiciels de deuxième industrielle est d'offrir une initiation à la programmation réseau. L'étudiant y apprend à mettre en œuvre le paradigme client-serveur en exploitant les services des sockets de la suite de protocoles TCP/IP. Cette compétence peut être vue comme une application pratique du cours « Modèles et Protocoles de Réseaux » de 1^e année option industrielle.

Pour des raisons de productivité et de fiabilité, le choix des outils de développement s'est porté sur le langage Java². et l'environnement de développement intégré, EDI (*Integrated Development Environment*, IDE) NetBeans (cf. <http://www.netbeans.org>)³. C'est pourquoi ce cours peut également être vu comme une suite à l'apprentissage de Java de 1^e année et un complément à l'apprentissage des langages orientés objets (C++ et Java) en ateliers logiciels.

En fin de semestre, l'étudiant doit être capable de concevoir et développer une application client-serveur simple interagissant avec ses utilisateurs. Pour ce faire, l'apprentissage comprend les étapes suivantes :

TD1 Découverte et prise en main de l'EDI NetBeans. Réactivation des compétences Java.

TD2 et TD3 Familiarisation avec les concepts de l'interface utilisateur graphique Swing (GUI) et de la programmation événementielle associée.

TD4 Compréhension de l'utilité des threads et leur exploitation en programmation réseau.

¹Le texte de ce TD s'inspire très largement de ceux des ateliers logiciels écrits jadis par différents collègues. Parmi les TDs que comprend ce cours, *MCD*, *NVS*, *RFS*, *SMB* et *VAK* ont contribué de manière diverses. Certains ont écrits entièrement un TD, d'autres ont contribué ou coécrits un TD, d'autres encore ont fait du travail de relecture. 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 6

TD5 Manipulations des classes entrées et sorties de flux de type fichier et réseau.

TD6 Initiation aux applications client-serveur exploitant les sockets TCP/IP.

Chaque TD contient ses concepts théoriques, ses exemples de codage, ses directives d'apprentissage et ses exercices pratiques. Le rythme du cours est d'environ une séance et demie par TD mais dépend également de la motivation des étudiants de chaque groupe, de leur travail à domicile et de leurs interactions avec leur professeur. Par ailleurs, la pratique a démontré qu'un projet « fil rouge » pouvait faciliter l'apprentissage et contribuer à une intégration des compétences acquises étape par étape. Les étudiants peuvent, s'ils le désirent, convenir avec leur professeur⁴ des spécifications pour un projet de développement personnel d'une application client-serveur, voire d'« égal-à-égal » (*peer-to-peer*).

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...

2.1 Directives d'apprentissage

On trouve **sur le site de Netbeans**, un guide de prise en main de NetBeans. Nous vous encourageons vivement à suivre ce tutoriel⁵ afin d'être en confiance avec cet environnement.

Le suivi pas à pas de ce tutoriel devrait permettre à l'étudiant de se sentir 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 de 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.

⁴Renseignez-vous auprès de vos professeurs respectifs

⁵ Accessible depuis le menu de l'EDI : Menu Help >> Option Quick Start Guide.

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 recourons 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 ;
- ↪ sauvegarder 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* 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 méthode de classe qui crée un triangle de Pascal de n lignes puis une méthode de classe qui l'affiche. Écrivez enfin une méthode 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 :

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

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

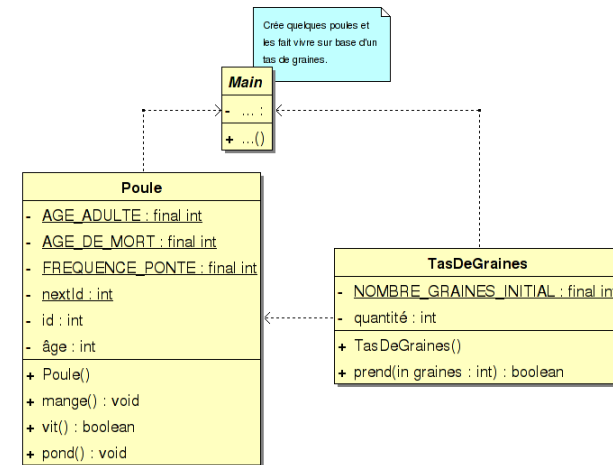


FIG. 1 – Diagramme UML

Exercice 2

Récupérez dans `eDistri/nvs` le programme « Répertoire », c'est-à-dire les trois fichiers `Personne.java`, `Repertoire.java` et `TestRepertoire.java`, et servez-vous en pour créer un projet NetBeans⁶.

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 « poulailler », manipulant des poules. Vous écrirez une classe (`Poule` qui représentera une poule⁷, une classe `TasDeGraines` et une classe `Main` permettant de tout mettre en oeuvre (voir FIG 1).

⁶Vous savez maintenant qu'il est possible de créer un projet NetBeans à partir de sources existantes

⁷Cet animal qui dandine de la tête et qui pond des oeufs.

La classe Poule représente une poule et possède les attributs ⁸,

- ↪ `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`).

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 adulte. 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 « poulailler » sur base d'une liste de poules et d'un tas de graines.

⁸ Je ne parle pas des constantes que vous définirez comme renseignées sur le diagramme.