E

Apprendre Java et C++ avec NetBeans

Généralités

Est-ce le titre d'un nouvel ouvrage ? Pourquoi pas !

NetBeans fait son apparition dans cette édition car un module, permettant d'éditer et de compiler du code C++ en plus de Java, a été récemment intégré dans son environnement. Par ailleurs, l'environnement MinGW, utilisé dans cet ouvrage, est directement intégré et reconnu.

NetBeans (`http://www.netbeans.org/`), tout comme Eclipse (`http://www.eclipse.org/`), est un outil de développement Open Source très puissant et particulièrement bien construit et adapté pour créer des applications ou encore des produits complexes.

Pour plus d'informations sur NetBeans en français, nous recommandons le le site Web suivant :

`http://www.netbeans.org/index_fr.html`

L'installation de NetBeans 6.1 a été vérifiée sous Windows XP et Vista. Les développeurs de NetBeans étant très actifs, de nouvelles versions apparaissent régulièrement. Cependant, avant de passer à une version plus récente (voire bêta), l'auteur conseille au lecteur de se familiariser tout d'abord avec cette version 6.1. Les différences entre versions peuvent être significatives et ne plus correspondre à la description et aux configurations décrites dans cette annexe.

NetBeans est relativement gourmand en ressources. Il peut fonctionner sur un processeur Intel traditionnel avec 512 Mo de mémoire, mais cela reste frustrant pour le développeur.

Pour un investissement minimal, il est recommandé de doubler la mémoire. Utiliser NetBeans sur un processeur Quad Core d'Intel avec 3 Go de mémoire est un vrai plaisir. Un écran de très grande définition (typiquement 1 600 × 1 200 pixels) permettra également de bien visualiser le code source des gros projets.

Linux

Pour un ordinateur aux performances limitées (mémoire et processeur), nous conseillons au lecteur d'utiliser NetBeans dans un environnement Linux, par exemple, Ubuntu (voir annexe F, « Apprendre Java et C++ avec Linux »).

L'installation et l'utilisation de NetBeans sous Linux et sous Windows sont équivalentes. Les différences de configuration sont indiquées dans l'annexe F.

Téléchargement de nouvelles versions

Le site Web de NetBeans (`http://www.netbeans.org/`) permettra aux lecteurs de vérifier si de nouvelles versions existent et de les télécharger le cas échéant afin de les installer (complètement ou uniquement certains composants, comme Java). Les versions Linux sont aussi disponibles sur ce site (voir annexe F).

Documentations et didacticiels

Ce même site Web fera découvrir aux lecteurs un nombre impressionnant de documents, d'exemples, de didacticiels ou encore de vidéos de présentation.

Installation à partir du CD-Rom

L'espace disque requis est au minimum de 140 Mo. Si d'autres composants sont installés, nous pouvons atteindre plus de 350 Mo (voir les options et paquets supplémentaires ci-dessous).

Nous pouvons évidemment installer NetBeans 6.1 séparément, mais il nous faudra au moins une machine virtuelle Java avant de commencer l'installation. Pour utiliser la partie C/C++ de NetBeans, les outils MinGW et MSYS doivent être préalablement installés. Il est donc préférable de procéder tout d'abord aux installations décrites dans l'annexe B (installation des outils incluant Java) et aux vérifications nécessaires.

NetBeans est un outil de développement permettant d'éditer du code et de le compiler. Des débogueurs y sont intégrés (Java et C++), lesquels permettent de faire du pas à pas dans le code source pendant l'exécution afin de s'assurer du bon déroulement du programme ou encore d'examiner le contenu des variables. Nous en donnerons d'ailleurs quelques exemples dans les sections « Naviguer et déboguer » et « Déboguer un projet C++ avec NetBeans » de cette annexe.

La version NetBeans choisie pour cet ouvrage est la version complète 6.1. Elle intègre non seulement le développement de programmes Java, mais aussi celui de programmes C++. Le lecteur pourra de plus s'intéresser par la suite à d'autres aspects et d'autres outils, comme les diagrammes UML ou l'écriture de programmes pour téléphones portables ou le Web (servlets).

La procédure d'installation détaillée ici a été réalisée sous Windows XP mais elle est identique sous Windows Vista. Pour lancer l'installation de NetBeans, nous double-cliquerons sur le programme `netbeans-6.1-ml-windows.exe` situé dans le répertoire `INSTALL` du CD-Rom.

Figure E-1
NetBeans 6.1 – Démarrage de l'installation

Nous cliquerons ensuite sur le bouton Next > afin de poursuivre l'installation (figure E-2) :

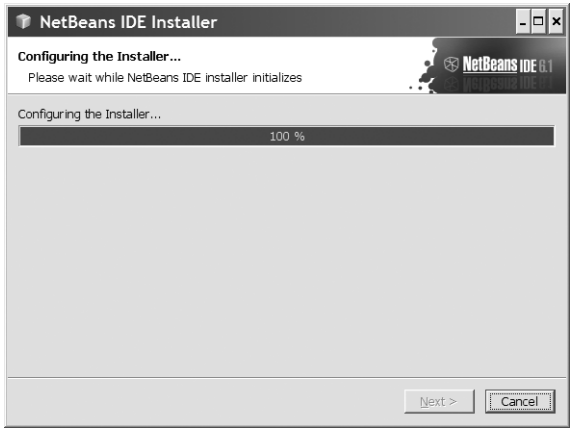

Figure E-2
Installation et initialisation de NetBeans 6.1

La fenêtre de la figure E-3 apparaîtra alors :

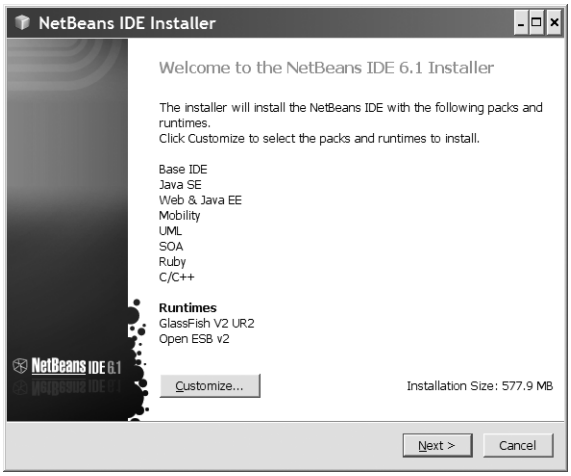

Figure E-3

Présentation des composants optionnels

Comme l'installation de tous les paquets demande plus de 577 Mo, le lecteur pourra décider de n'en garder que quelques-uns, suivant l'espace disque dont il dispose. Pour sélectionner les composants à installer, il faudra cliquer sur le bouton Customize…, ce qui aura pour effet d'ouvrir la fenêtre de la figure E-4.

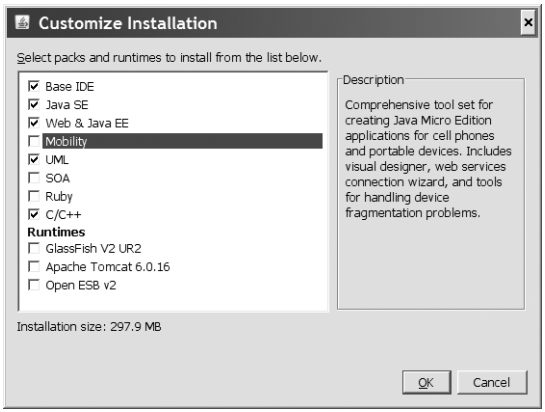

Figure E-4

Sélection des composants à installer

Ici, nous avons conservé les composants Web & Java EE ainsi que UML. Ils ne sont pas vraiment nécessaires pour cet ouvrage mais nous y reviendrons par la suite. Avec ces

deux composants, l'espace disque requis est d'environ 300 Mo (il serait de 140 Mo seulement si Web & Java EE et UML n'étaient pas sélectionnés). Nous cliquerons ensuite sur OK et la fenêtre de la figure E-5 apparaîtra alors, récapitulant les composants à installer et l'espace disque nécessaire. À ce stade, nous pourrons à nouveau cliquer sur le bouton Customize… pour ajouter ou éliminer un composant.

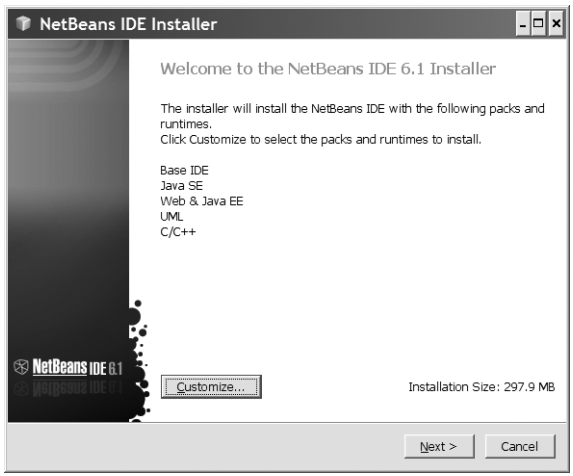

Figure E-5

Liste des composants à installer

Nous cliquerons ensuite sur Next > pour lancer la dernière partie de l'installation. La fenêtre de la figure E-6 s'affichera alors à l'écran :

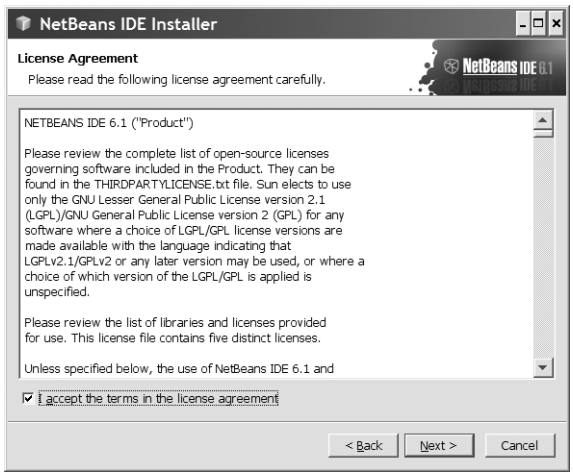

Figure E-6

Termes du contrat de licence de NetBeans 6.1

Nous accepterons les termes du contrat et nous cliquerons sur Next >. Dans la fenêtre suivante (figure E-7), nous choisirons les répertoires d'installation de NetBeans :

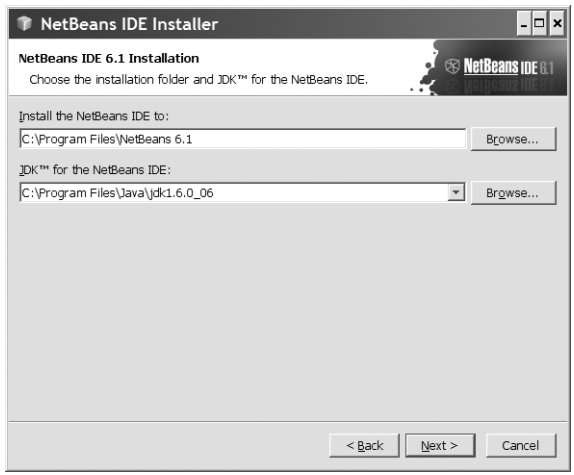

Figure E-7
Choix des répertoires d'installation de NetBeans

Nous conserverons les deux répertoires spécifiés par défaut. Nous constaterons au passage que le second, JDK, a été identifié automatiquement (comme installé dans l'annexe B). Si ce n'était pas le cas, il faudrait le définir manuellement au moyen du bouton Browse… Si l'espace disponible sur le disque `C:` est insuffisant, le lecteur pourra choisir d'installer NetBeans et ses composants sur une autre partition. Une fois les chemins d'accès aux répertoires d'installation renseignés, nous cliquerons alors sur le bouton Next > pour continuer. La fenêtre suivante (figure E-8) indiquera alors le répertoire d'installation choisi et l'espace disque nécessaire.

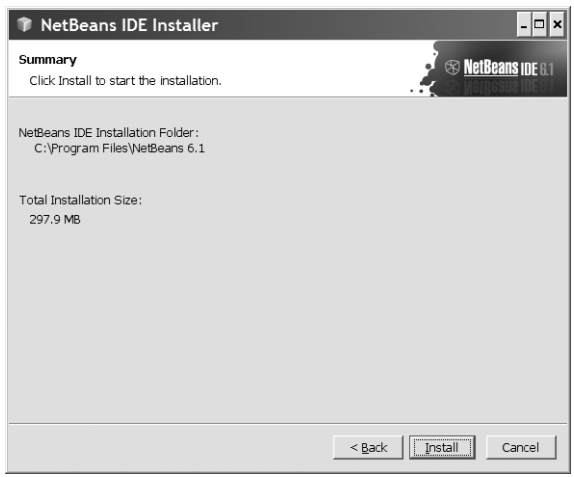

Figure E-8
Installation finale de NetBeans

Si tout est correct, nous pourrons cliquer sur le bouton Install.

Plusieurs messages apparaîtront ensuite dans une nouvelle fenêtre, indiquant l'état d'avancement de l'installation (figure E-9) :

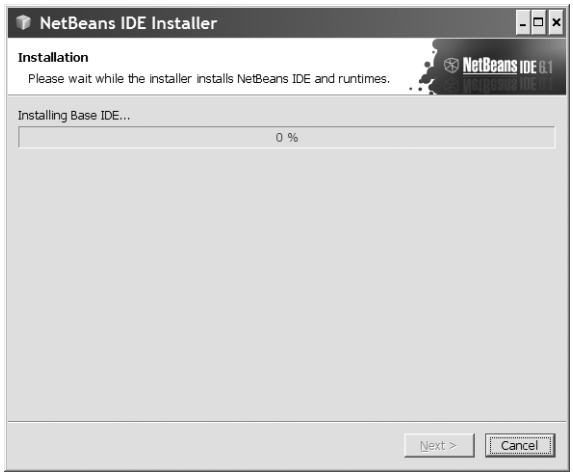

Figure E-9
Progression de l'installation

La durée de l'installation dépendra de la capacité de la machine (mémoire, CPU et disque dur). Avant la fin de l'installation, la fenêtre suivante s'affichera (figure E-10) :

Figure E-10
Fin de l'installation de NetBeans 6.1

Nous activerons l'option Register the NetBeans IDE after finishing the installer, si nous désirons nous enregistrer gratuitement afin de recevoir les informations de mise à jour pour ce produit. Pour terminer l'installation, nous cliquerons enfin sur Finish.

L'icône représentée à la figure E-11 devrait être alors apparaître sur le Bureau (dans le cas contraire, nous pourrons facilement créer un raccourci vers NetBeans). Pour démarrer NetBeans, il suffira alors de double-cliquer dessus.

Figure E-11
NetBeans – Icône de démarrage

Au premier lancement de NetBeans, nous pourrions rencontrer le message de la figure E-12 si nous avions déjà installé NetBeans ou dans le cas d'une réinstallation ou d'une mise à jour :

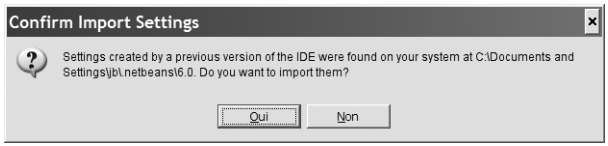

Figure E-12
Reprise d'anciennes configurations

Ici, nous conseillerons de cliquer sur le bouton Non afin d'avoir un nouvel environnement vierge. Si nous cliquons sur Oui, tous nos anciens projets seront importés et il faudra sans doute revoir chacun d'entre eux si l'ancienne version de NetBeans est trop différente par rapport à la nouvelle.

La fenêtre de démarrage de NetBeans s'affichera alors (figure E-13) :

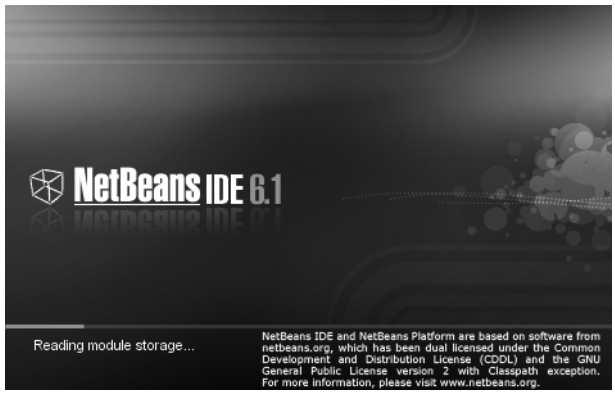

Figure E-13
Démarrage de NetBeans 6.1

Le premier démarrage sera un peu plus lent que les suivants car NetBeans configurera en arrière-plan l'environnement. L'interface de travail par défaut apparaîtra ensuite (figure E-14) :

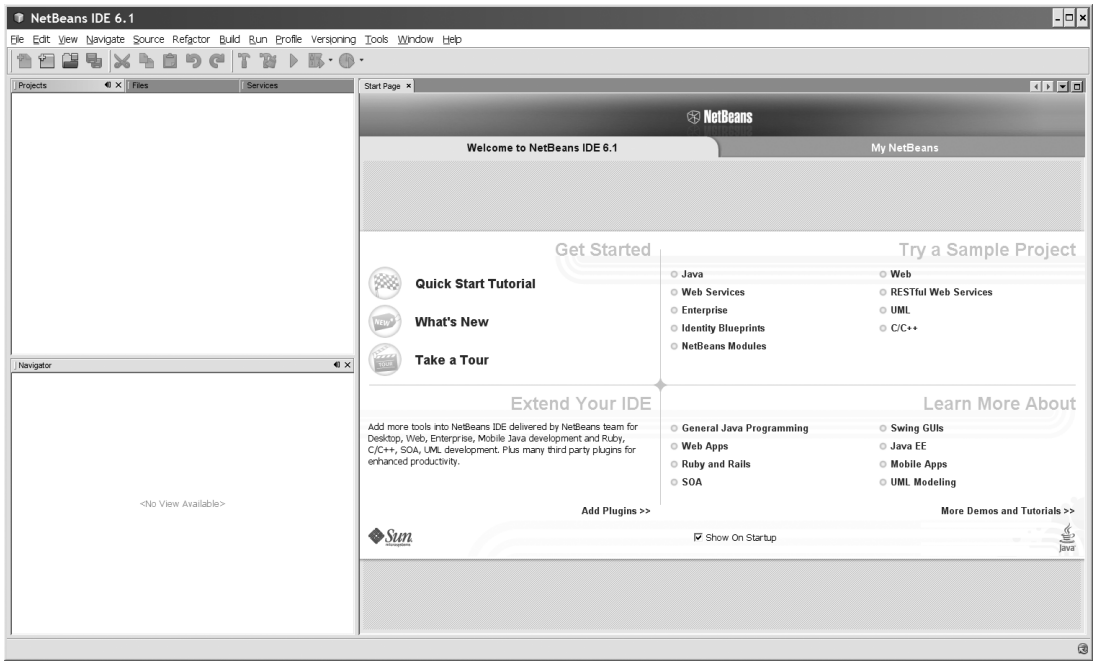

Figure E-14

Interface de travail par défaut de NetBeans

La rapidité d'apparition de cette fenêtre dépendra des ressources de l'ordinateur. À noter qu'un écran de grande définition pourrait aider considérablement à la bonne visualisation des programmes et au travail quotidien.

Une petite icône située en bas à droite de l'interface de travail nous indique que des mises à jour sont disponibles (figure E-15).

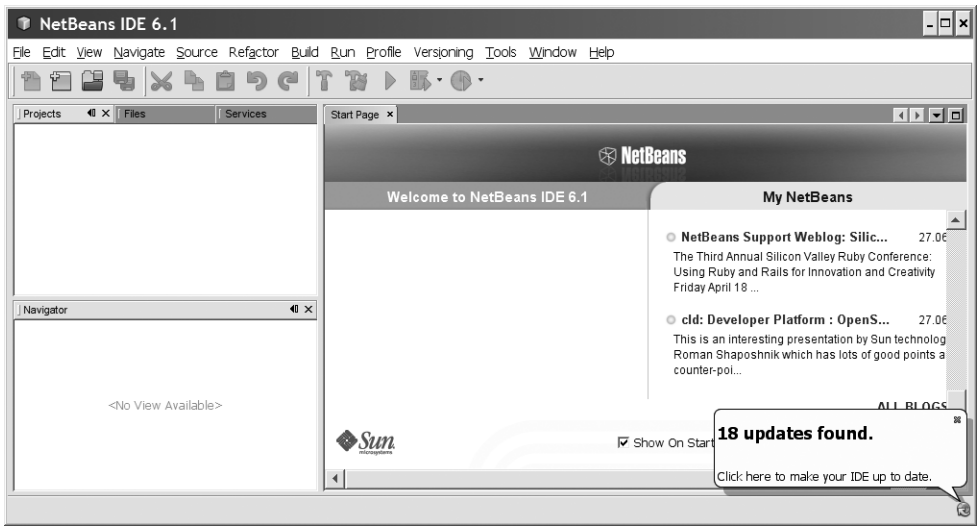

Figure E-15

Icône indiquant que des mises à jour sont disponibles.

Nous pouvons alors choisir de les installer en cliquant dessus (il y aura sans doute plus de 18 mises à jour, figure E-16) ou de les ignorer (voir le commentaire situé après la figure E-17) :

Figure E-16

Mise à jour des paquets

Une fois les mises à jour téléchargées et installées, la fenêtre de la figure E-17 apparaîtra, suite à l'installation du plug-in UML :

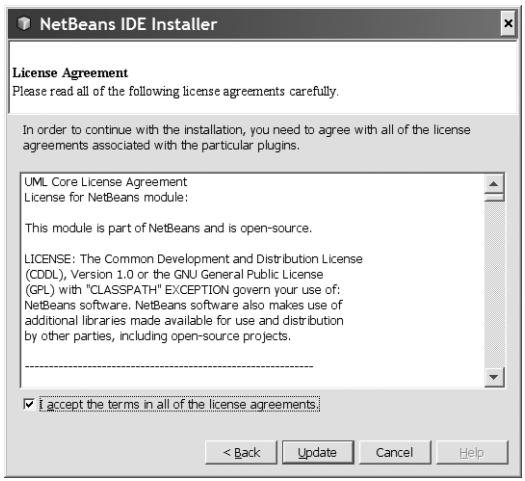

Figure E-17

Contrat de licence pour le paquet UML

Nous cliquerons ici sur le bouton Cancel afin de nous assurer que la version utilisée dans cette annexe correspond bien à celle décrite dans ce livre. Dès que l'utilisateur sera familier avec l'outil, il est recommandé de procéder à cette mise à jour (l'équipe de développement de NetBeans en propose très régulièrement, il est conseillé de s'enregistrer pour recevoir des e-mails d'information). Si nous acceptons la mise à jour (bouton Update), le lecteur pourrait s'attendre à de petites variations sur les fonctions décrites dans la suite de cette annexe.

Dans la fenêtre principale de NetBeans (onglet Start Page), nous désactiverons ensuite l'option Show On Startup. De cette manière, l'écran de bienvenue ne s'affichera plus lors des prochaines ouvertures de NetBeans. Pour que cette modification soit bien prise en compte, nous fermerons NetBeans (menu File > Exit) et le relancerons. Il peut s'avérer intéressant de visiter les liens proposés sur cet écran de bienvenue avant de l'effacer. Nous y trouverons en effet différentes informations sur le produit.

Au prochain démarrage, nous obtenons l'interface de travail présentée à la figure E-18 :

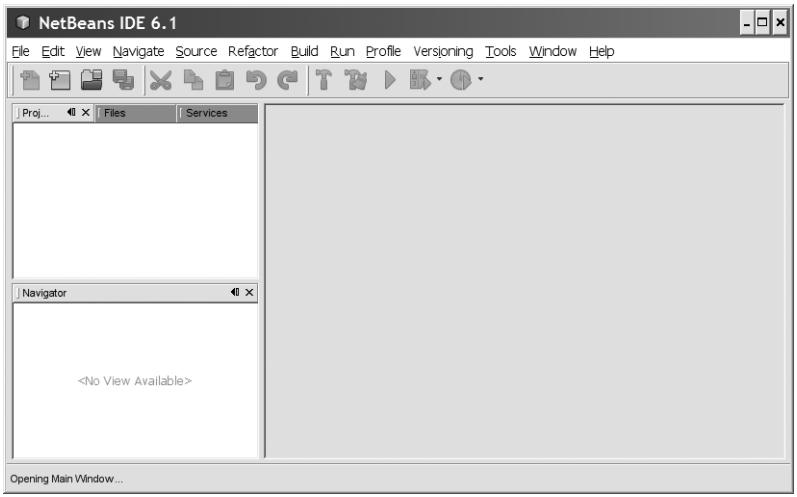

Figure E-18

Interface de travail de NetBeans, vierge au démarrage

L'interface de travail se présentera toujours selon l'aspect, la dimension et la position choisis avant de quitter NetBeans. Suivant la dimension de l'écran, la taille du code source (affiché dans la fenêtre de droite) et les informations indiquées dans les fenêtres Projets, Files ou Navigator, le lecteur pourra redimensionner à souhait chaque fenêtre horizontalement ou verticalement. Pour cela, il faudra placer le curseur de la souris entre les fenêtres, cliquer sur le bouton gauche et déplacer la barre verticale qui apparaîtra alors.

Configuration pour le C++ et le make

Le menu Tools > Options > C/C++ permet de contrôler la configuration de NetBeans et de corriger éventuellement le chemin d'accès du `make` (figure E-19).

Dans l'annexe B, section « Problèmes potentiels avec le make », nous avons mentionné les difficultés susceptibles d'être liées à cet outil, qu'il nous faudrait alors éventuellement remplacer par une autre version, suivant la machine et la version de Windows. Le chemin d'accès indiqué ici semble fonctionner dans tous les cas.

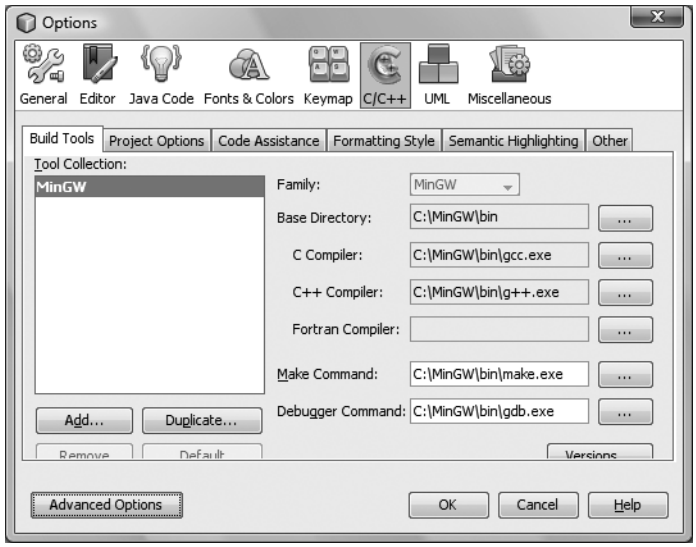

Figure E-19

NetBeans – Configuration du C++ et du make

Présentation de NetBeans

Nous allons ici présenter l'utilisation de NetBeans pour Java et C++ sous Windows Vista (c'est identique sous Windows XP).

Cet ouvrage a été construit avec des petits modules plutôt que des grosses applications ou projets. Il n'a donc pas été envisagé, dans cette édition, d'intégrer tous les chapitres ou tous les exemples dans des projets NetBeans.

En revanche, un programmeur qui acquerra de l'expérience avec Java ou C++ et qui voudra développer une application substantielle y trouvera son bonheur. Il pourra créer un nouveau projet en y intégrant son code ou bien consulter des exemples ou des exercices de cet ouvrage. Pour des applications graphiques Java, le cœur du programme sera sans doute développé avec NetBeans et ensuite intégré avec les classes ou les bibliothèques existantes.

NetBeans et Java

Java et la classe Personne

Pour débuter simplement, nous prendrons l'exemple de la classe `Personne` du chapitre 4.

Nous avons tout d'abord créé un nouveau répertoire `C:\JavaCpp\NetBeans\Chap04a` et copié les fichiers `Personne.java` et `TestPersonne.java` (figure E-20) :

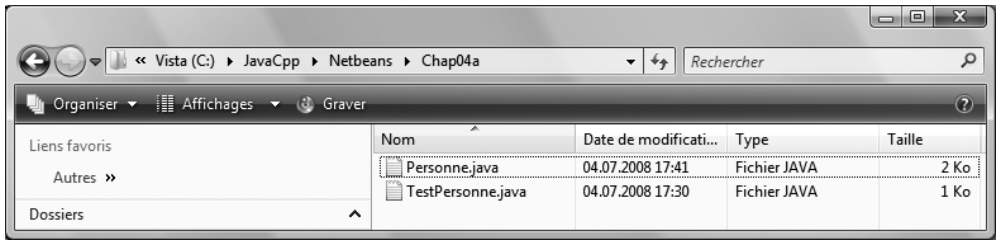

Figure E-20

Préparation d'un projet NetBeans

Lorsque nous éditerons par la suite ces fichiers depuis NetBeans, ils seront modifiés à cet endroit.

Le fichier `Personne.java`, dans ce cas précis, contient du texte avec des accents utilisant malheureusement l'encodage ANSI. Nous conseillons aux lecteurs d'employer l'encodage UTF-8 dans NetBeans. Pour vérifier quel type d'encodage est utilisé, il faut effectuer un clic droit sur le nom du projet, ici `JavaProject4a`, et sélectionner Properties (figure E-21) :

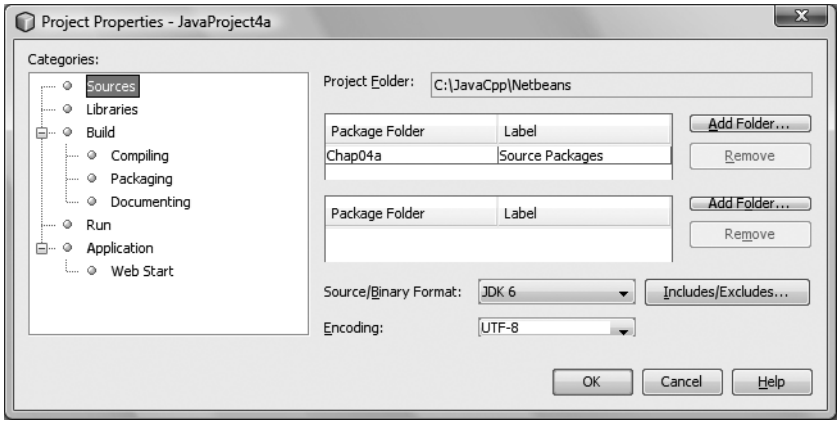

Figure E-21

Configuration de l'encodage UTF-8 dans NetBeans

L'encodage UTF-8 (champ Encoding) est préférable si nous désirons par la suite inclure des textes comportant des caractères particuliers comme c'est le cas pour l'alphabet cyrillique ou arabe. Ces textes peuvent aussi faire partie de la documentation ou de l'exécution du code.

Nous avons deux choix possibles : convertir les fichiers ANSI qui contiennent des accents en UTF-8 ou les conserver tels quels et procéder ensuite à des corrections manuelles dans l'éditeur de NetBeans (voir ci-dessous). Pour une conversion préalable, nous pouvons ouvrir le fichier `Personne.java` avec le Bloc-notes et sélectionner le menu Enregistrer sous en spécifiant le même nom de fichier mais l'encodage UTF-8 (figure E-22) :

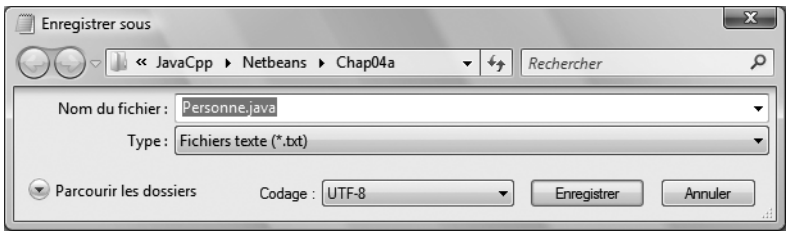

Figure E-22

Sauvegarde du fichier Personne.java avec l'encodage UTF-8

Comme Windows possède son propre type de fichier, il faudra encore apporter une petite correction dans la source et depuis NetBeans. Nous y reviendrons.

Nouveau projet avec source existante

Voici le cas qui nous intéresse ici. En effet, nous aimerions créer un nouveau projet avec du code existant depuis notre répertoire `C:\JavaCpp\NetBeans\Chap04a`. Pour cela, nous sélectionnons le menu File > New project de NetBeans ou saisissons le raccourci clavier Ctrl+Shift+N (figure E-23).

Nous choisirons Java Project with Existing Sources, afin d'utiliser le code que nous venons de préparer.

Nous nommerons le projet `JavaProject4a` en souvenir du chapitre 4, le `a` étant ajouté car nous pensons créer d'autres projets pour ce chapitre par la suite. Le répertoire du projet est important et il est conseillé de n'en spécifier qu'un seul, soit `C:\JavaCpp\NetBeans`, dans lequel tous nos projets seront stockés (figure E-24).

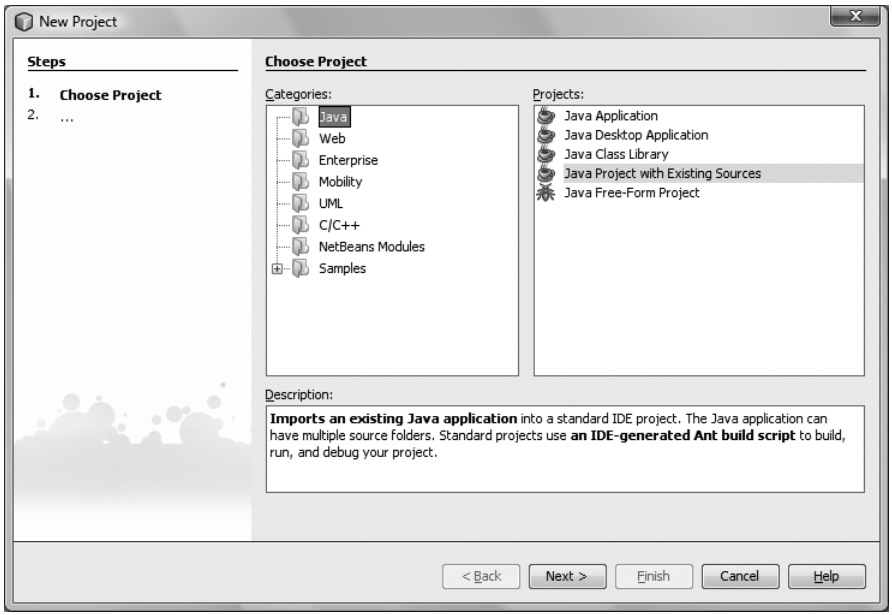

Figure E-23

Création d'un nouveau projet avec source existante dans NetBeans

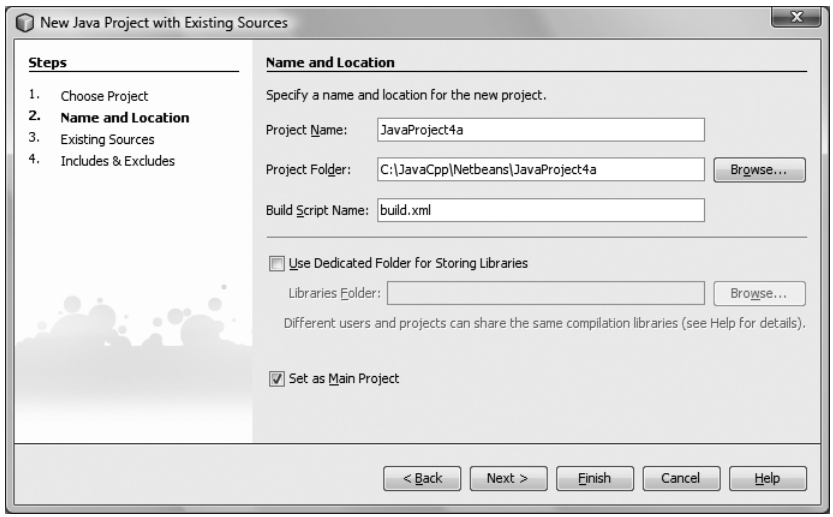

Figure E-24

Spécification du nom et du répertoire du projet NetBeans

Ensuite, nous cliquerons sur le bouton Next > et indiquerons la référence à nos deux fichiers source dans le répertoire créé précédemment (figure E-25) :

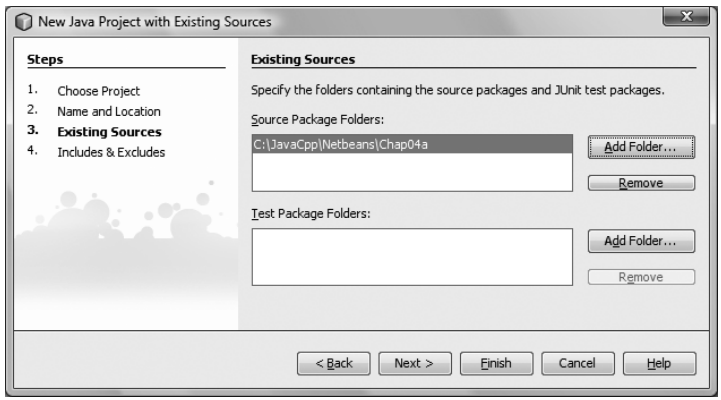

Figure E-25

Spécification des fichiers source du projet

Nous cliquerons à nouveau sur le bouton Next > et indiquerons ensuite quels fichiers source nous désirons employer (figure E-26) :

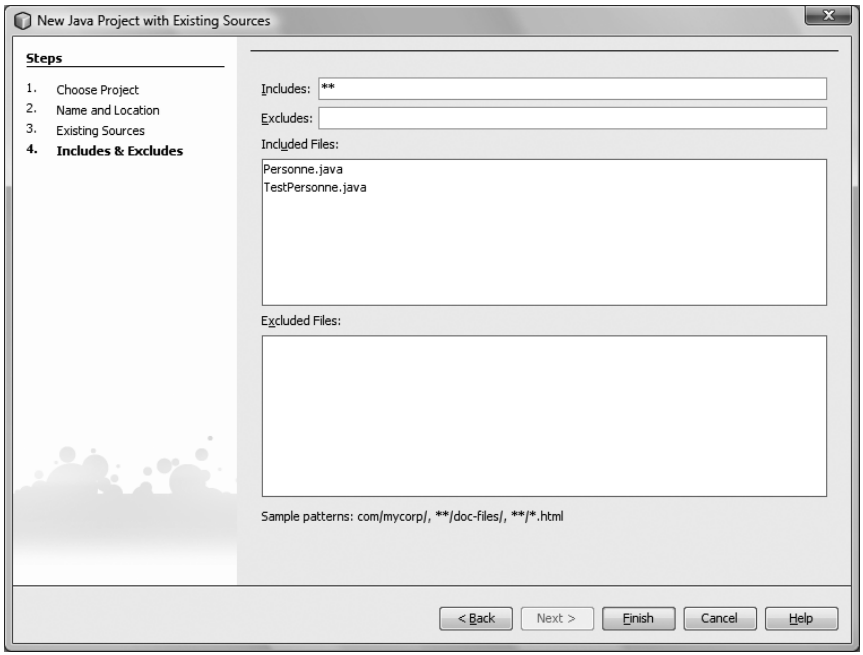

Figure E-26

Récupération de nos deux fichiers Java

Nous cliquerons enfin sur Finish pour terminer.

Pour obtenir cette l'arborescence de la figure E-27, il nous faudra cliquer sur le signe + situé devant <default package>.

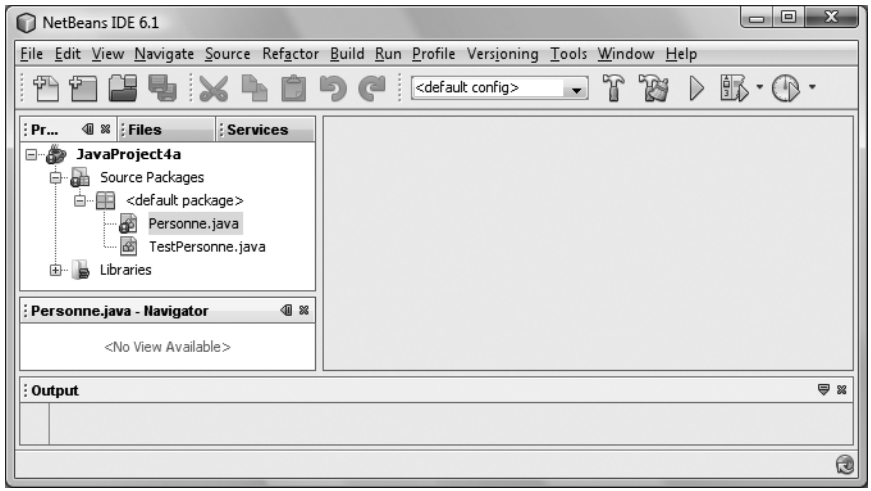

Figure E-27

Projet NetBeans presque prêt

Nous constatons qu'un point d'exclamation rouge est présent devant le fichier `Personne.java`. Ceci indique que le fichier comporte un caractère non reconnu, généré par Windows lors de la sauvegarde sous le Bloc-notes en mode UTF-8. Nous l'effacerons tout simplement.

Nous aurions pu également corriger les caractères accentués non reconnus sans passer par la conversion Bloc-notes. Dans un fichier au codage ANSI, un texte comme « Prénom » apparaîtra sous la forme « Pr□nom » dans NetBeans et il suffira alors de corriger ce caractère non reconnu.

D'ailleurs, nous conseillerons d'utiliser NetBeans pour réaliser la documentation Javadoc et plus particulièrement si celle-ci est en français (figure E-28) (voir le chapitre 4 et la section « Javadoc » de cette annexe).

En cliquant sur la touche F11 du clavier, qui correspond au raccourci du menu Build Main Project, nous allons pouvoir sauvegarder nos éventuelles adaptations de code ou de documentation et construire nos classes Java compilées (figure E-29).

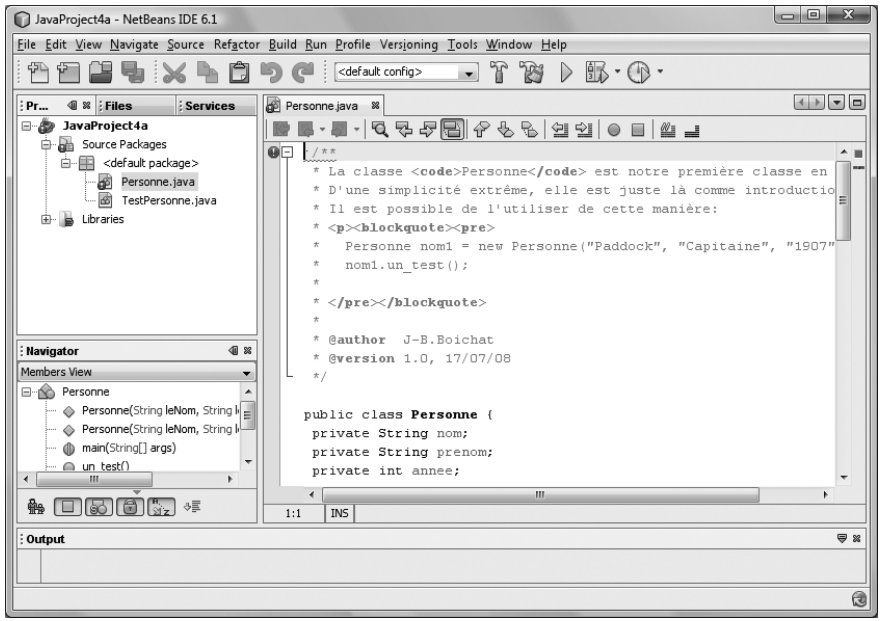

Figure E-28

Les dernières adaptations UTF-8

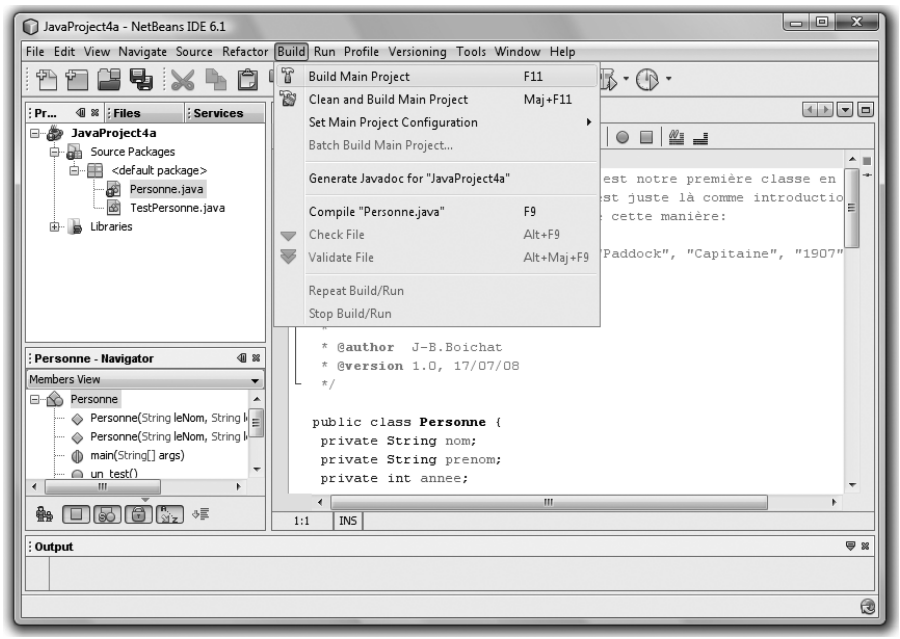

Figure E-29

Le fichier est prêt pour notre première compilation Java.

Après la compilation du projet, si nous naviguons dans l'explorateur de Windows (figure E-30), nous pourrons constater qu'il a été enregistré, ainsi que les fichiers Java, dans le répertoire `JavaProject4a`. Les fichiers compilés sont dans le répertoire `C:\JavaCpp\NetBeans\JavaProject4a\build\classes`.

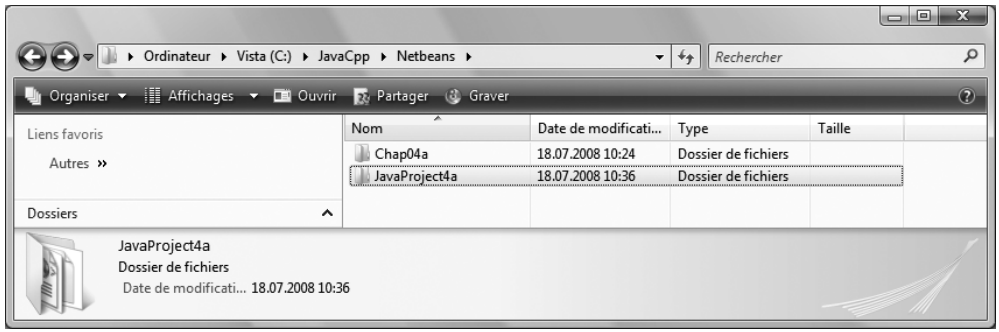

Figure E-30

Nos répertoires de travail sous NetBeans

Le menu Run de NetBeans va nous permettre d'exécuter le projet. Comme nous avons deux classes avec des points d'entrée (`main()`), NetBeans va nous demander depuis quelle entrée du projet nous allons exécuter le programme (figure E-31) :

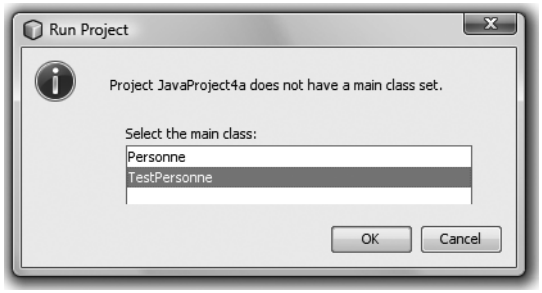

Figure E-31

Sélection de la classe TestPersonne

Nous sélectionnerons `TestPersonne`, notre classe de test :

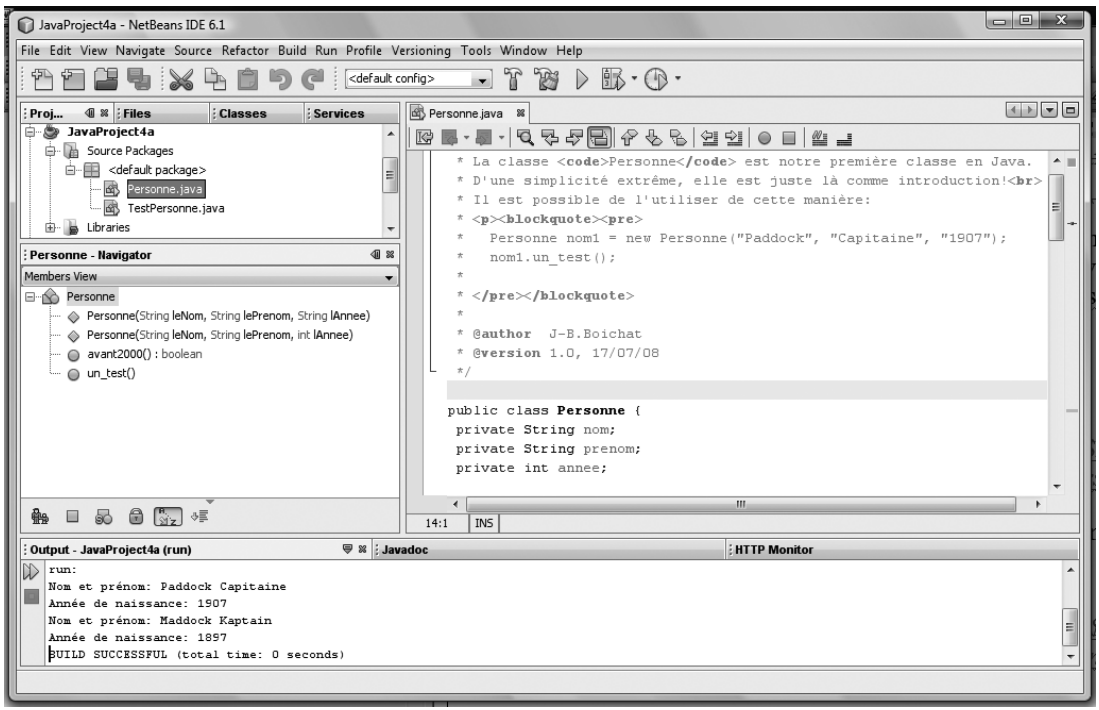

Figure E-32

Première exécution depuis NetBeans

Le résultat de l'exécution de la classe `TestPersonne.java` s'affiche dans la fenêtre Output de NetBeans. Dans le cas présenté, `TestPersonne.java` n'est pas ouvert avec l'éditeur. Ce n'est pas nécessaire et, de plus, tout le code principal de notre projet se trouve dans la classe `Personne`.

Comme il s'agit de la première compilation, les éléments de la fenêtre Navigator sont désormais accessibles et nous allons comprendre rapidement son utilité.

Pour exécuter à nouveau le projet (toujours avec `TestPersonne` comme point d'entrée), nous pouvons sélectionner le menu Run (touche F6) ou cliquer sur l'icône en forme de triangle vert dans la barre d'outils de NetBeans (figure E-32).

Si nous désirons changer le point d'entrée du `main()` pour le projet, il faudra effectuer un clic droit sur le nom du projet dans la fenêtre des projets, en haut à gauche de l'interface de travail, et sélectionner Properties. Nous cliquerons ensuite sur Run, puis sur le bouton Browse… situé à côté du champ MainClass afin de choisir le nouveau point d'entrée (figure E-33).

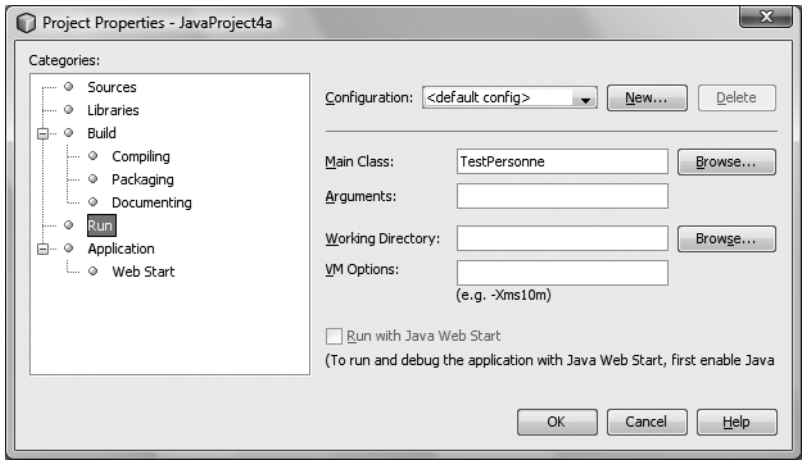

Figure E-33

Modification du main() par défaut

Distribuer nos applications

Une fois le développement de cette application terminé, il pourrait s'avérer intéressant de l'exécuter en dehors de l'environnement de NetBeans et de pouvoir la distribuer à l'extérieur. Nous en avons parlé rapidement à la fin du chapitre 4, section « Nos classes Java dans un paquet .jar » ; NetBeans construit ainsi un paquet .jar et le dépose dans le répertoire de distribution :

```
C:\JavaCpp\NetBeans\JavaProject4a\dist
```

Le fichier est ici nommé `JavaProject4a.jar` et nos amis ou clients pourraient l'exécuter avec :

```
java JavaProject4a.jar
```

à condition qu'une machine virtuelle Java version 1.6 soit installée sur leur PC (une version JRE suffit, voir chapitre 1).

Nous pourrons ensuite ouvrir les fichiers `.jar` avec 7-Zip et consulter leur contenu ainsi que le fichier `MANIFEST.MF` dont nous avons parlé au chapitre 4. Il contient, par exemple, la définition du point d'entrée, ici `Main-Class:TestPersonne`.

Naviguer et déboguer

Nous allons passer maintenant en revue quelques fonctionnalités proposées par NetBeans. La première, sans doute la plus utilisée, consiste à double-cliquer sur le nom du fichier dans la fenêtre Projets, ici `TestPersonne.java`. Le fichier s'ouvre alors pour l'édition et le

fichier précédent, `Personne.java`, reste accessible (il suffit de cliquer sur l'onglet du même nom, figure E-34) :

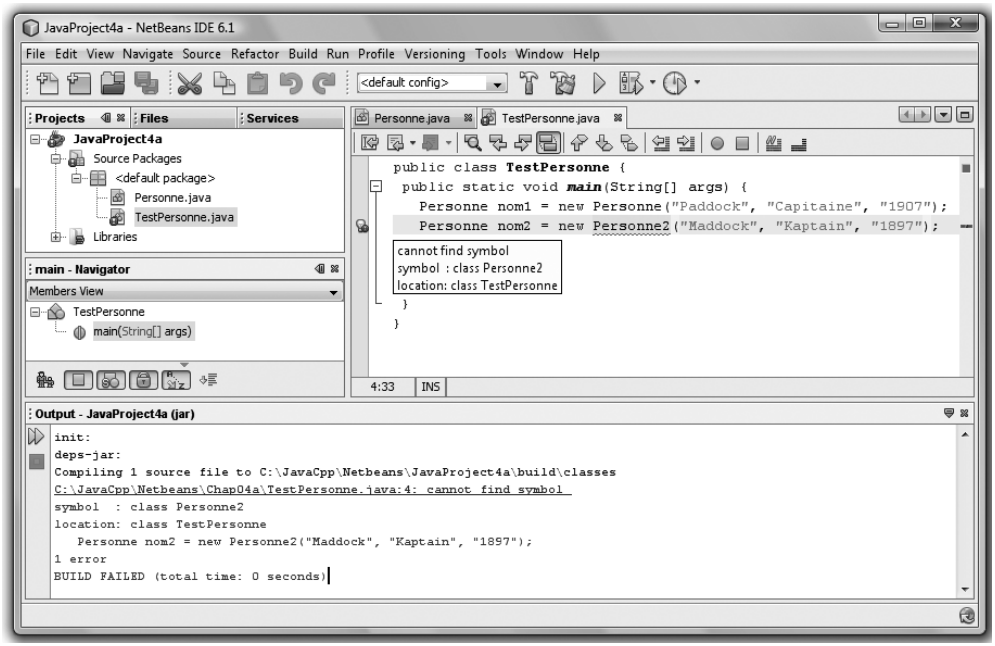

Figure E-34
Naviguer et déboguer

La fenêtre Navigator n'est pas intéressante ici car il n'y a qu'une seule entrée, le `main()`, dans la classe `TestPersonne.java`. Cependant, en fonction du fichier ouvert dans NetBeans, cette fenêtre peut contenir un large choix de méthodes ou d'attributs de classe, comme c'est le cas à la figure E-32. Si nous cliquions sur `un_test()`, la fenêtre de droite, permettant l'édition de `Personne.java`, se positionnerait sur le code de cette méthode de classe. Les différents pictogrammes nous informent des différents types ou attributs (par exemple, `public`, `private` ou autres).

Sur l'exemple de la figure E-34, nous avons ajouté le chiffre `2` après `Personne`, sur la deuxième ligne de code. Une petite ampoule apparaît alors immédiatement devant la ligne : elle nous indique un problème, même sans compiler. En se positionnant sur l'ampoule, ou en compilant le fichier avec le menu Build (le résultat s'affiche alors dans la fenêtre Output), nous verrons rapidement que la classe `Personne2` n'existe pas et que le chiffre `2` est de trop. Nous pouvons également cliquer sur la ligne de l'infobulle `cannot find symbol` : NetBeans se positionnera alors correctement sur la ligne 4 du fichier `TestPersonne.java`. Ceci s'avérera particulièrement pratique lorsque nous aurons beaucoup d'erreurs et dans plusieurs fichiers. Si le fichier comportant les erreurs n'est pas encore ouvert, il le sera automatiquement et le curseur sera positionné sur la ligne posant problème.

Nous supprimons donc le `2` de `Personne` afin de corriger l'erreur et cliquons sur la quatrième ligne de code (`Personne nom2 …`). Nous constatons alors qu'un carré rouge est apparu devant la ligne (figure E-35) :

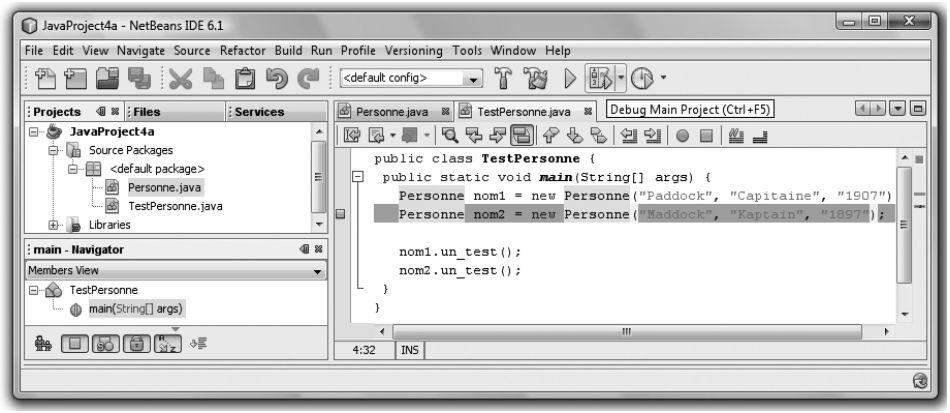

Figure E-35

Déboguer : se positionner au bon endroit

Ce carré rouge indique un point d'arrêt du débogueur. Le programme peut maintenant être lancé avec l'icône Debug (Ctrl+F5), également accessible depuis le menu Run ou en cliquant sur le bouton à droite du triangle vert de la barre d'outils.

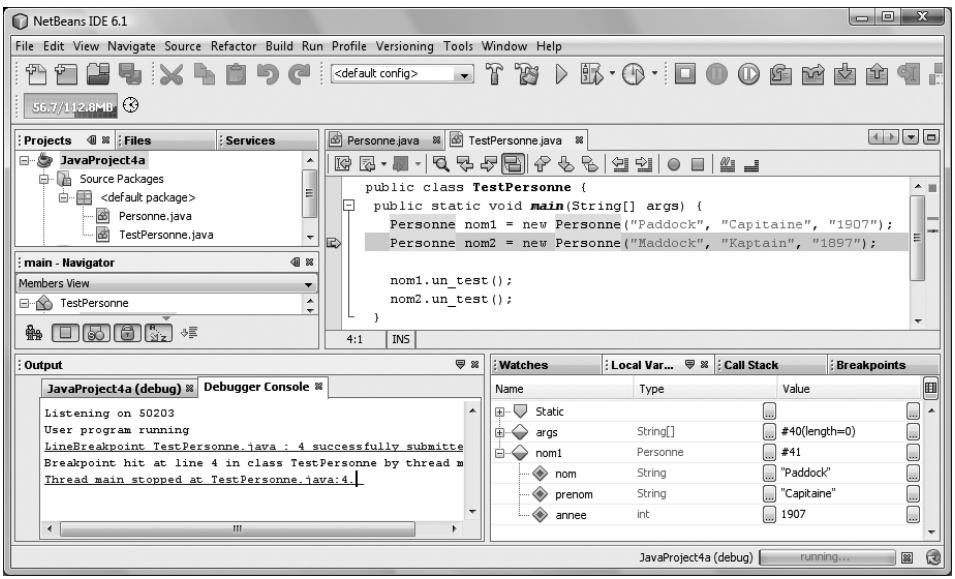

Figure E-36

Nous faisons du pas à pas.

Le programme s'est arrêté sur la ligne spécifiée, mais elle n'a pas encore été exécutée. Dans la fenêtre située en bas à droite de l'interface de NetBeans (figure E-36), nous cliquons sur l'onglet Local Variables afin d'examiner les variables locales déjà initialisées, par exemple l'objet `nom1` de la classe `Personne`. Différentes options sont maintenant possibles, comme le Step Into (F7), symbolisé par une flèche verticale vers le bas (accessible via la quatrième icône en partant de la droite dans la barre d'outils, ou par le menu Run).

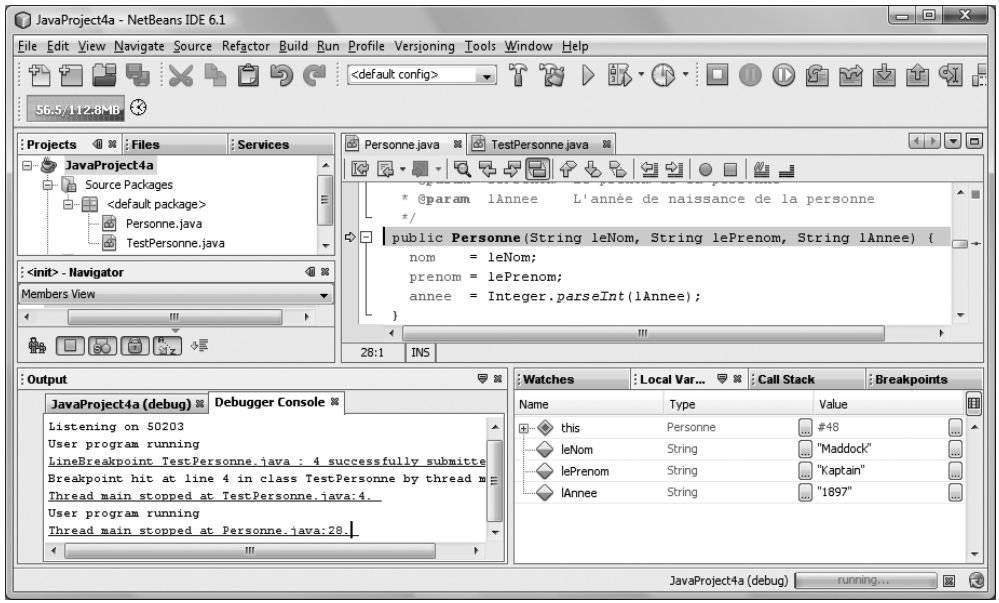

Figure E-37

Pas à pas à l'intérieur d'une classe

L'option Step Into nous a permis d'« entrer dans » le `new Personne("Maddock", "Kaptain", "1897");`, c'est-à-dire le constructeur de la classe `Personne` pour ce nouvel objet `nom2`. La variable `leNom` est bien `"Maddock"` : nous sommes à l'entrée du constructeur.

Nous laissons le lecteur faire lui-même d'autres découvertes : il peut ainsi continuer le programme (F5) et le terminer, pendant une pause de débogueur ou à n'importe quel instant si d'autres points d'arrêt n'ont pas été introduits entre-temps. D'autres aspects sont présentés ci-après pour le C++ et suivent les mêmes concepts (par exemple, Build and Clean Main Project du menu Build).

Javadoc

Pour notre classe `Personne` du chapitre 4, nous avons édité la documentation Java avec Crimson. Dans ce même chapitre, nous avons donné quelques indications pour coder correctement les mots-clés (par exemple, `@param` ou `@return`). Nous allons à présent montrer

toute la puissance d'un outil comme NetBeans pour nous aider à faire ce travail correctement et proprement.

La première chose à faire est d'activer la fonction Javadoc via le menu Tools > Options > Java Code > Hints de NetBeans (figure E-38) :

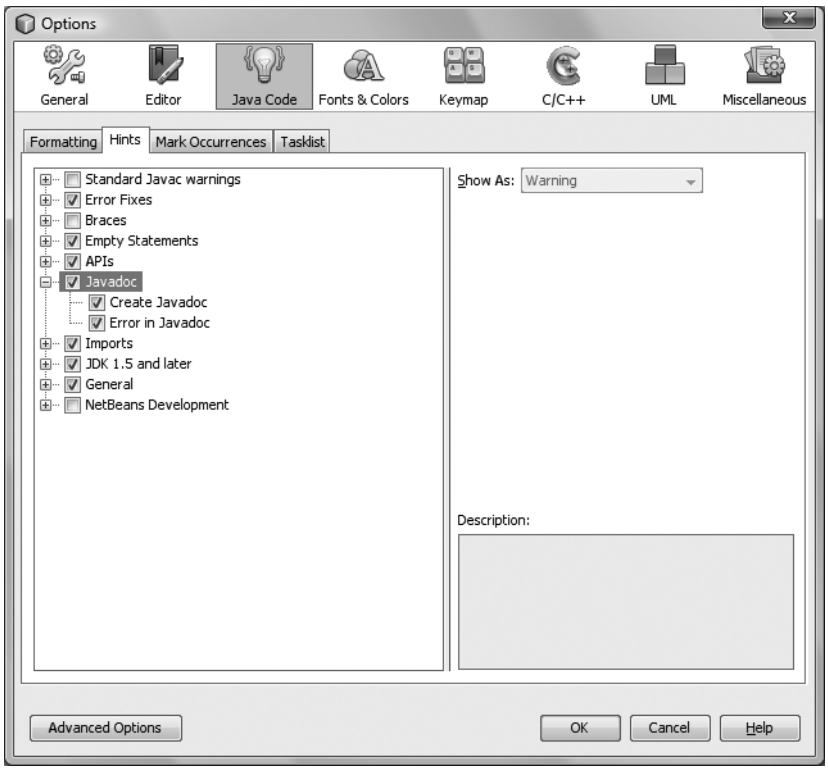

Figure E-38

Activation de la fonction Javadoc

En guise d'exemple, nous allons ajouter une nouvelle méthode appelée `avant2000()` qui va nous retourner un `boolean` (`true` ou `false`, vrai ou faux), nous indiquant si la personne est née avant l'an 2000.

```
public boolean avant2000() {
     if (annee < 2000) {
         return true;
     }
     return false;
 }
```

Lors de l'édition de cette méthode, nous constatons que NetBeans nous aide durant la saisie du texte. Par exemple, si nous définissons la méthode sans son contenu, ce que nous faisons généralement pour nous assurer que les accolades (`{ }`) sont bien présentes, NetBeans va nous indiquer que le retour manque (`missing return statement`) si nous nous positionnons sur le point d'exclamation rouge dans la marge. Nous aurons d'autres indications si le `f` du `if` ou bien une parenthèse est omis. Ceci s'avère très utile, car nous n'avons pas besoin d'attendre la compilation pour corriger la faute.

Lorsque ce code est terminé, nous cliquons sur `avant2000()`(la première ligne de la méthode). Une petite ampoule apparaît alors devant la ligne, nous indiquant que la documentation manque. Il faudra alors cliquer sur la première ampoule puis sur la seconde qui apparaîtra ensuite (figure E-39) :

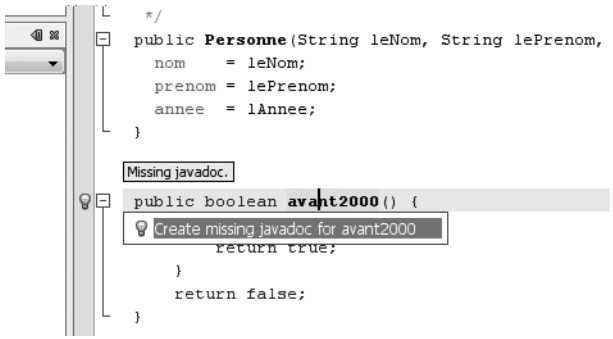

Figure E-39
Générer du Javadoc manquant

afin que le bloc de texte prêt pour l'entrée de la documentation soit généré automatiquement :

```
/**
 *
 * @return
 */
```

Il suffira alors de saisir correctement le texte désiré.

Grâce au menu Window > Other > Javadoc View, nous pouvons afficher la documentation formatée de notre méthode `avant2000()` dans la fenêtre Output (figure E-40) :

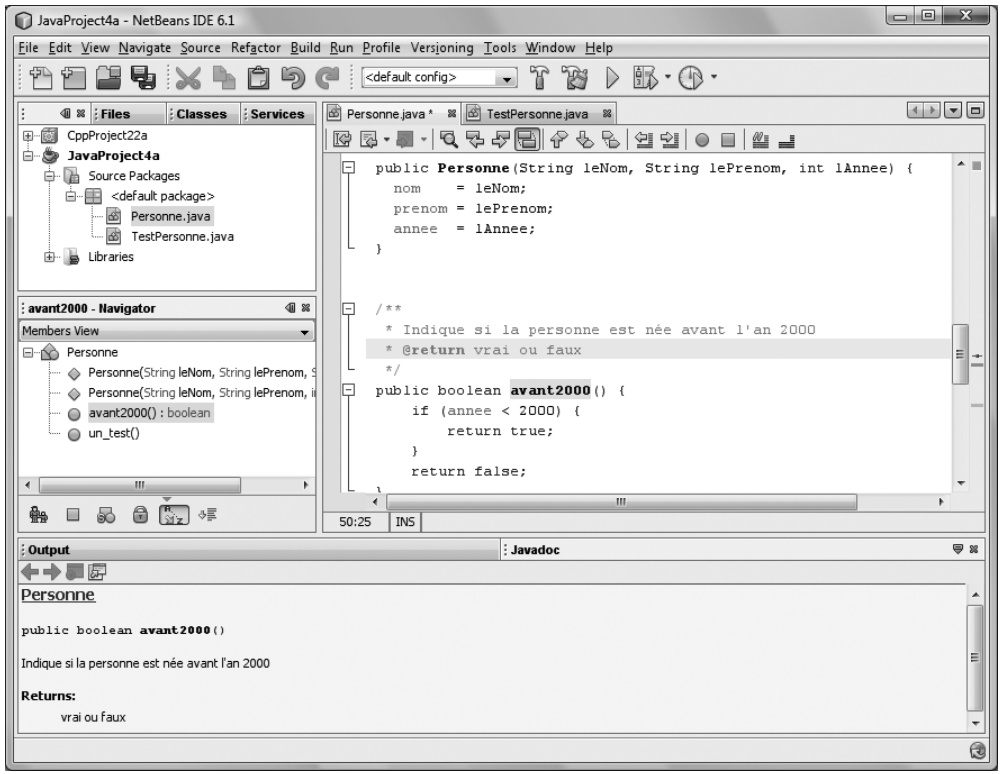

Figure E-40

Saisir et visualiser le Javadoc

En effectuant un clic droit sur le projet `JavaProject4a` dans la fenêtre en haut à gauche, nous pourrons sélectionner l'entrée Generate Javadoc. Si nous cliquons dessus, nous obtiendrons la fenêtre représentée à la figure E-41.

La documentation pour nos méthodes `public`, en particulier la nouvelle méthode `avant2000()`, est ainsi générée et disponible dans le répertoire `C:/JavaCpp/NetBeans/JavaProject4a/dist/javadoc`.

Cette manière de faire est évidemment plus conviviale et efficace que celle décrite au chapitre 4, plus traditionnelle.

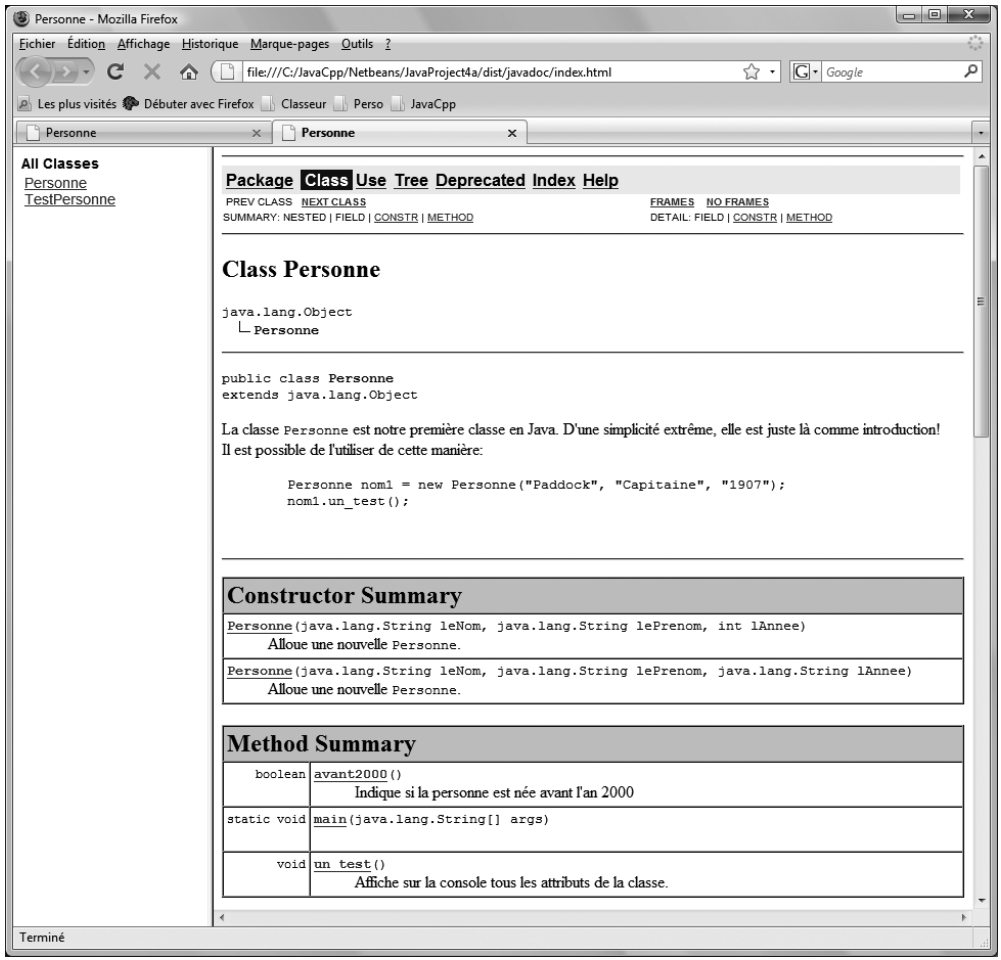

Figure E-41

Javadoc final pour la classe Personne

UML – Diagramme de classe

Nous allons maintenant présenter rapidement un exercice de *Reverse Engineering* : générer des diagrammes UML (*Unified Modeling Language*) et en particulier des diagrammes de classe à partir d'un code existant. Notre classe `Personne` représente un bon exemple pour ce genre d'exercice et sa mise en pratique sera très facile grâce à NetBeans.

Nous commencerons par effectuer un clic droit sur le projet `JavaProject4a` dans la fenêtre en haut à gauche, afin d'accéder à la fonctionnalité Reverse Engineer… (figure E-42).

Nous conserverons les paramètres par défaut et cliquerons sur le bouton OK.

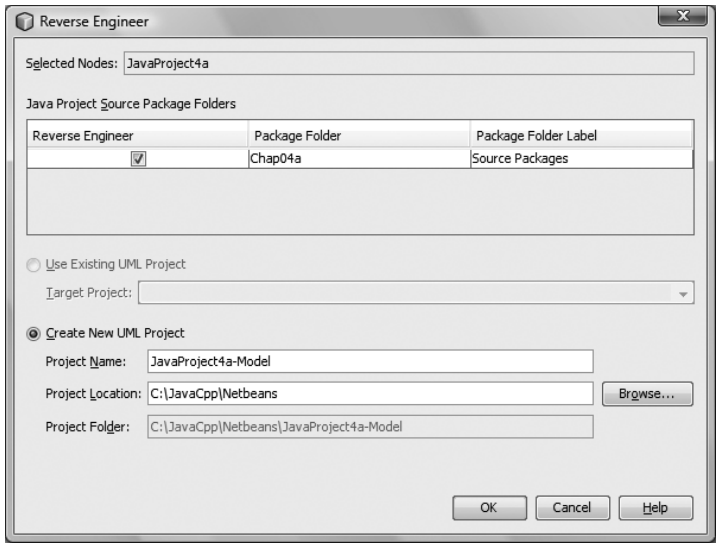

Figure E-42

NetBeans prêt pour du Reverse Engineering

Nous ouvrirons ensuite le projet `JavaProject4a-Model` (créé lors du Reverse Engineering), le `Model` (le répertoire des diagrammes UML), et effectuerons un clic droit sur la classe `Personne` afin de sélectionner Create Diagram from Selected Elements :

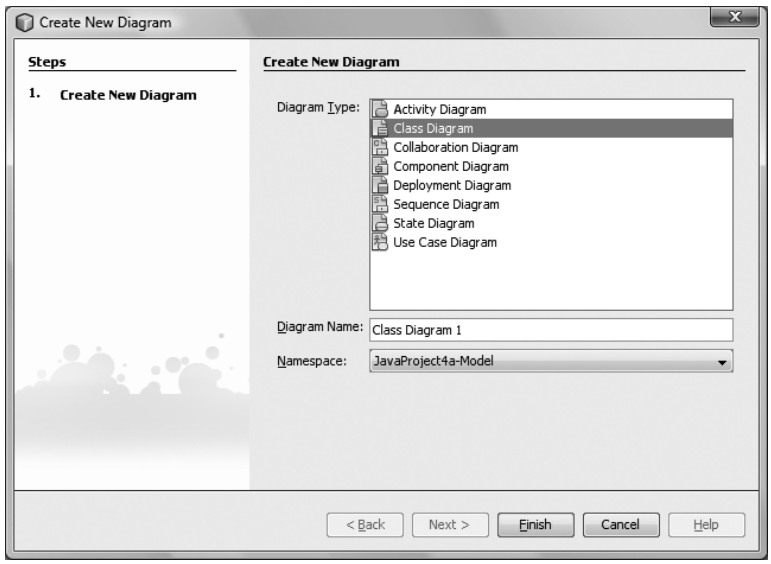

Figure E-43

Diagramme de classe à créer

Nous sélectionnerons Class Diagram et cliquerons sur le bouton Finish. La fenêtre de la figure E-44 s'affichera alors :

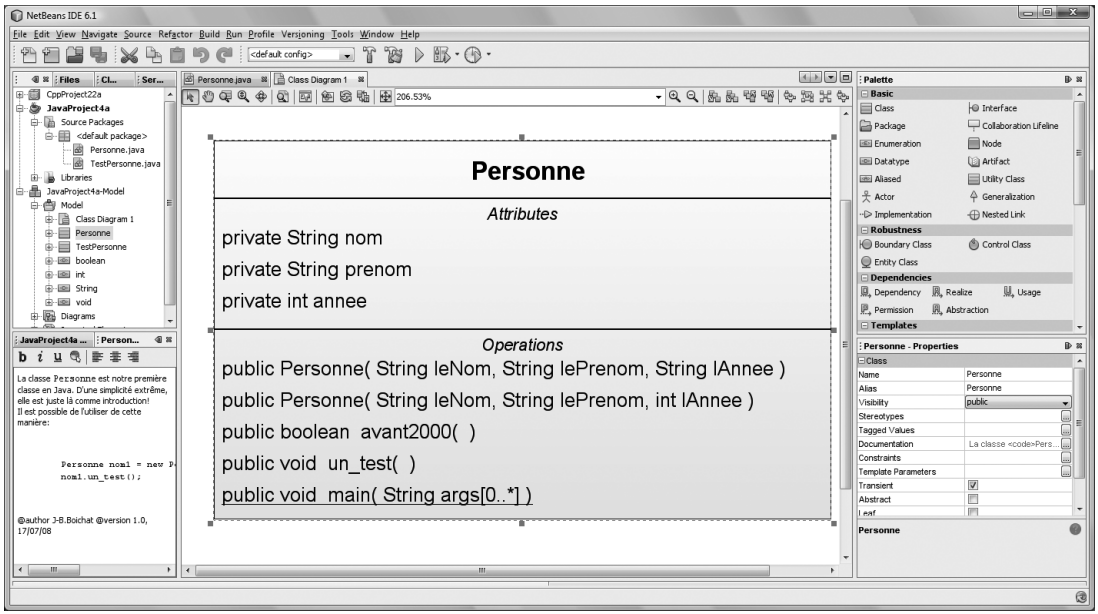

Figure E-44

Diagramme pour la classe Java Personne

Nous laisserons le lecteur affamé de technologie découvrir le monde UML avec cette référence sur le Web :

`http://fr.wikipedia.org/wiki/Unified_Modeling_Language`

L'étape suivante serait évidemment de définir ses propres classes ou d'étendre la classe `Personne` à partir de ces diagrammes UML, et de laisser NetBeans générer le code Java.

NetBeans et C++

Le jeu d'Othello dans NetBeans

Pour le C++, nous allons procéder de la même manière que pour Java mais avec le fichier `othello2.cpp` du chapitre 22. Comme pour l'exercice avec Java, nous allons copier ce fichier dans le répertoire `C:\JavaCpp\NetBeans\Chap22a`.

NetBeans va nous demander un `Makefile` à importer, contrairement à l'exemple détaillé pour Java. Nous allons donc copier ce `Makefile` dans `C:\JavaCpp\NetBeans\Chap22a` à partir des exemples du chapitre 22 et ne garder que la partie concernant le C++ et notre fichier `othello2.cpp` comme ceci :

```
all:  cpp

cpp:  othello2.exe

othello2.exe:   othello2.o
                g++ -g -o othello2.exe othello2.o

othello2.o:     othello2.cpp
                g++ -g -c othello2.cpp
clean:
                rm -rf *.o *.exe
```

Les entrées `all` et `clean` sont requises par un `Makefile` standard, mais absolument par NetBeans.

Par rapport à la version d'origine, nous avons ajouté `-g` après chaque `g++`. Lors de la compilation, ce paramètre va nous permettre un débogage pas à pas avec le débogueur `gdb` que nous avons intégré à notre distribution MinGW. Pour une installation personnalisée depuis Internet (voir annexe B), il faudra sans doute rechercher et installer cet outil séparément.

Nous pouvons très bien charger ce « projet » dans Crimson et vérifier qu'il fonctionne (figure E-45) :

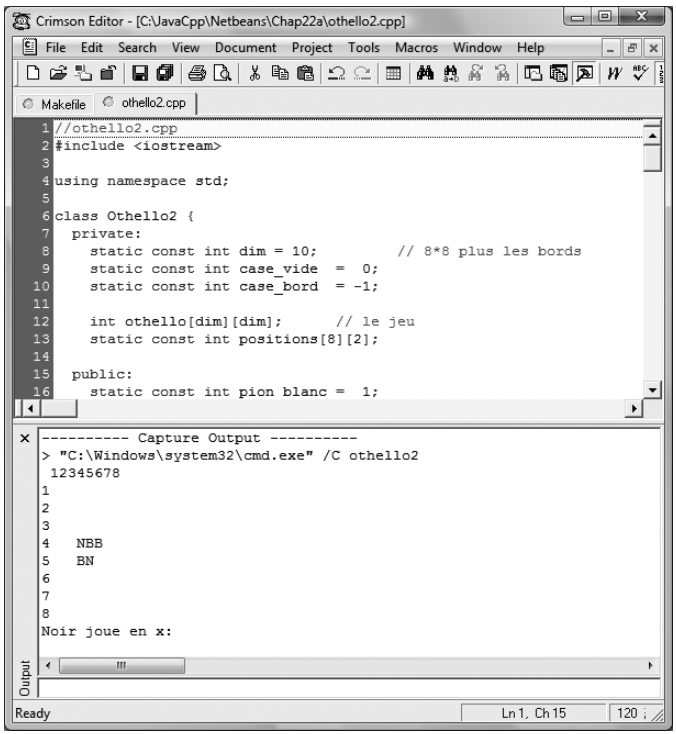

Figure E-45

Vérification avec Crimson