Tutoriel BlueJ

Michael Kölling
Mærsk Institute
University of Southern Denmark

Version 1.4 fr par le groupe Sigma[1]
pour BlueJ Version 1.2.x

Table des matières

[1] `http://www.enseeiht.fr/lima/sigma`, `bluej@enseeiht.fr`

1 Avant-propos

1.1 BlueJ

Ce tutoriel est une introduction à l'utilisation de l'environnement de programmation BlueJ. BlueJ est un environnement de développement Java(tm) destiné à l'enseignement initial à la programmation. Il a été conçu et développé par l'équipe BlueJ de l'université Monash à Melbourne, en Australie, et l'université de Southern Denmark, Odense.

Vous trouverez plus d'information sur BlueJ à l'adresse `http://www.bluej.org`.

1.2 But et public visé

Ce tutoriel est destiné aux utilisateurs qui veulent se familiariser avec l'environnement. Il ne contient ni d'explications sur la conception de l'environnement ni sur les recherches en amont.

Ce tutoriel n'a pas pour but l'apprentissage de Java. Il est conseillé aux débutants de consulter un livre d'introduction à Java ou de suivre un cours de Java.

Ce document n'est pas non plus un manuel de référence exhaustif sur l'environnement. Beaucoup de détails sont omis – l'accent est mis sur la brièveté et la concision plutôt que sur l'analyse complète des fonctionnalités.

Chaque section commence par un résumé qui permettra aux lecteurs qui sont déjà familiers avec certaines parties du système de décider de l'opportunité de lire ou de sauter une partie. La section 10 ne fait que répéter ces résumés et fait office d'index du document.

1.3 Copyright, droit de licence et de distribution

La traduction française a été réalisée par le groupe sigma (`http://www.enseeiht.fr/lima/sigma`).

1.4 Réactions

Les commentaires, questions, corrections, critiques ou toute autre forme de réactions concernant BlueJ ou ce tutoriel sont bienvenus et encouragés. Contacter Michael Kölling (`mik@mip.sdu.dk`) ou le groupe Sigma `bluej@enseeiht.fr` pour la traduction en français.

2 Installation

BlueJ est distribué sous trois formats différents : pour Windows, pour MacOS et pour les autres systèmes. La procédure d'installation est assez simple.

Prérequis

Vous devez disposer de J2SE v1.3 (i.e. JDK 1.3) ou supérieur installé sur votre système avant d'utiliser BlueJ. Si vous n'avez pas un JDK installé vous pouvez en télécharger un à partir du site de Sun à l'adresse `http://java.sun.com/j2se`. Sur MacOS X, une version récente est pré-installée – vous n'avez pas besoin d'en installer une vous-même. Si vous trouvez une page proposant un « JRE » (*Java Runtime Environment*) et un « SDK » (*Software Development Kit*), vous devez télécharger le SDK – le JRE ne suffit pas.

2.1 Installation sous Windows

Le fichier de distribution pour les systèmes Windows s'appelle `bluejsetup-xxx.exe`, où `xxx` représente le numéro de version. Par exemple, la distribution BlueJ version 1.2.0 est contenue dans un fichier nommé `bluejsetup-120.exe`. Vous pouvez trouver ce fichier sur un CD ou vous pouvez le télécharger à partir du site BlueJ à l'adresse `http://www.bluej.org`.

Exécutez ce fichier d'installation. L'exécutable vous permet de choisir un répertoire d'installation. Vous avez aussi la possibilité d'installer un raccourci dans le menu de démarrage et sur le bureau.

Une fois l'installation terminée, le programme `bluej.exe` se trouve dans le répertoire d'installation de BlueJ.

Lors du premier lancement, BlueJ recherche un JDK sur le système. Si plus d'un système Java est trouvé (par exemple, vous avez le JDK 1.3.1 et le JDK 1.4 installés), une boîte de dialogue vous permettra d'en choisir un. Si aucun JDK n'est trouvé, il vous sera demandé de le localiser vous-même (cela peut arriver dans le cas où un JDK est installé mais les entrées correspondantes dans la base de registres ont été supprimées).

Le logiciel d'installation BlueJ installe aussi un programme appelé `vmselect.exe`. Celui-ci vous permettra de modifier ultérieurement la version Java utilisée par BlueJ. Exécuter `vmselect` pour faire démarrer BlueJ avec une version différente de Java.

Le choix du JDK est stocké pour chaque version BlueJ. Si vous avez plusieurs versions de BlueJ installées, il est possible d'utiliser une version de BlueJ avec le JDK 1.3.1 et une autre avec le JDK 1.4. Le changement de version Java pour BlueJ changera la configuration de toutes les installations de la même version BlueJ pour le même utilisateur.

2.2 Installation sous Macintosh

Veuillez remarquer que BlueJ tourne sous MacOS X seulement.

Le fichier de la distribution pour MacOS s'appelle `BlueJ-xxx.sit`, où `xxx` est un numéro de version. Par exemple, la distribution BlueJ version 1.2.0 s'appelle `BlueJ-120.sit`.

Vous trouverez ce fichier sur un CD ou vous pourrez le télécharger à partir de `http://www.bluej.org`.

Ce fichier peut être décompressé avec StuffIt Expander, et beaucoup de navigateurs le feront pour vous. Sinon vous pouvez le décompresser en double-cliquant sur le fichier dans le Finder.

Après la décompression, vous aurez un répertoire appelé `BlueJ-xxx`. Déplacez-le dans votre répertoire Applications (ou à n'importe quel endroit de votre choix). Il n'y a rien d'autre à faire pour l'installation.

2.3 Installation sous Linux/Unix et autres systèmes

Le fichier d'installation est un fichier jar exécutable. Son nom est `bluej-xxx.jar`, où `xxx` est le numéro de version. Par exemple, la distribution BlueJ version 1.2.0 est contenue dans un fichier nommé `bluej-120.jar`. Vous pouvez trouver ce fichier sur votre disque ou vous pouvez le télécharger à partir du site BlueJ à l'adresse `http://www.bluej.org`.

Lancez l'installation en exécutant la commande suivante :

```
<jdk-path>/bin/java -jar bluej-120.jar
```

où `<jdk-path>` est le répertoire où le JDK est installé.

> NB : *Pour cet exemple, la distribution* `bluej-120.jar` *est utilisée. Vous devez indiquer le nom du fichier correspondant à votre distribution (avec le bon numéro de version).*

Une fenêtre apparaît et vous permet de choisir le répertoire d'installation de BlueJ ainsi que la version de JDK pour exécuter BlueJ. Important : le chemin d'accès vers BlueJ (y compris les répertoires parents) ne doit pas contenir d'espaces.

Appuyez sur Install pour finir l'installation.

2.4 Problèmes d'installation

Si vous avez des problèmes, consultez d'abord les foires aux questions (FAQ) sur le site BlueJ (`http://www.bluej.org/help/faq.html`) et lisez la section *How To Ask For Help* (`http://www.bluej.org/help/ask-help.html`).

3 Pour débuter – édition / compilation / exécution

3.1 Lancement de BlueJ

Sous Windows et MacOS, un programme nommé `BlueJ` est installé. Lancez-le. Sous les systèmes Unix, le programme d'installation place un script nommé `bluej` dans le répertoire d'installation. A partir d'une interface graphique, le lancement s'opère par un double clic sur le nom du fichier. Depuis une ligne de commande, vous pouvez lancer BlueJ avec ou sans nom de projet en argument :

```
$ bluej
```

ou

```
$ bluej examples/people
```

3.2 Ouverture d'un projet

Résumé : *Pour ouvrir un projet, sélectionner Open dans le menu Project.*

Les projets BlueJ, comme les paquetages Java standard, sont des répertoires contenant les fichiers associés.

Après avoir lancé BlueJ, utilisez la commande Project > Open pour choisir et ouvrir un projet.

Quelques exemples de projets sont proposés dans le répertoire `examples` de la distribution standard de BlueJ.

Pour cette partie du tutoriel, ouvrez le projet `people` qui est inclus dans ce répertoire. Vous pourrez trouver le répertoire `examples` dans le répertoire principal de BlueJ. Après avoir ouvert le projet vous devez obtenir une fenêtre semblable à celle de la figure 1. La fenêtre peut ne pas être exactement la même selon votre système, mais les différences seront mineures.

3.3 Création d'objets

Résumé : *Pour créer un objet, sélectionner un constructeur dans le menu contextuel de la classe.*

Une des caractéristiques fondamentales de BlueJ est que, en plus de l'exécution d'applications complètes, vous pouvez également interargir directement avec des objets de n'importe quelle classe et exécuter leurs méthodes publiques. Une exécution sous BlueJ est souvent faite en créant un objet puis en invoquant une de ses méthodes. Ceci est extrêmement utile pendant la phase de développement d'une application – vous pouvez tester les classes une par une dès qu'elles ont été écrites. Il n'est donc pas nécessaire de commencer par écrire toute l'application.

> NB : *Les méthodes statiques peuvent être exécutées directement sans avoir besoin de commencer par créer un objet. La méthode* `main` *des applications Java est statique, il est donc possible de lancer une application sim-*

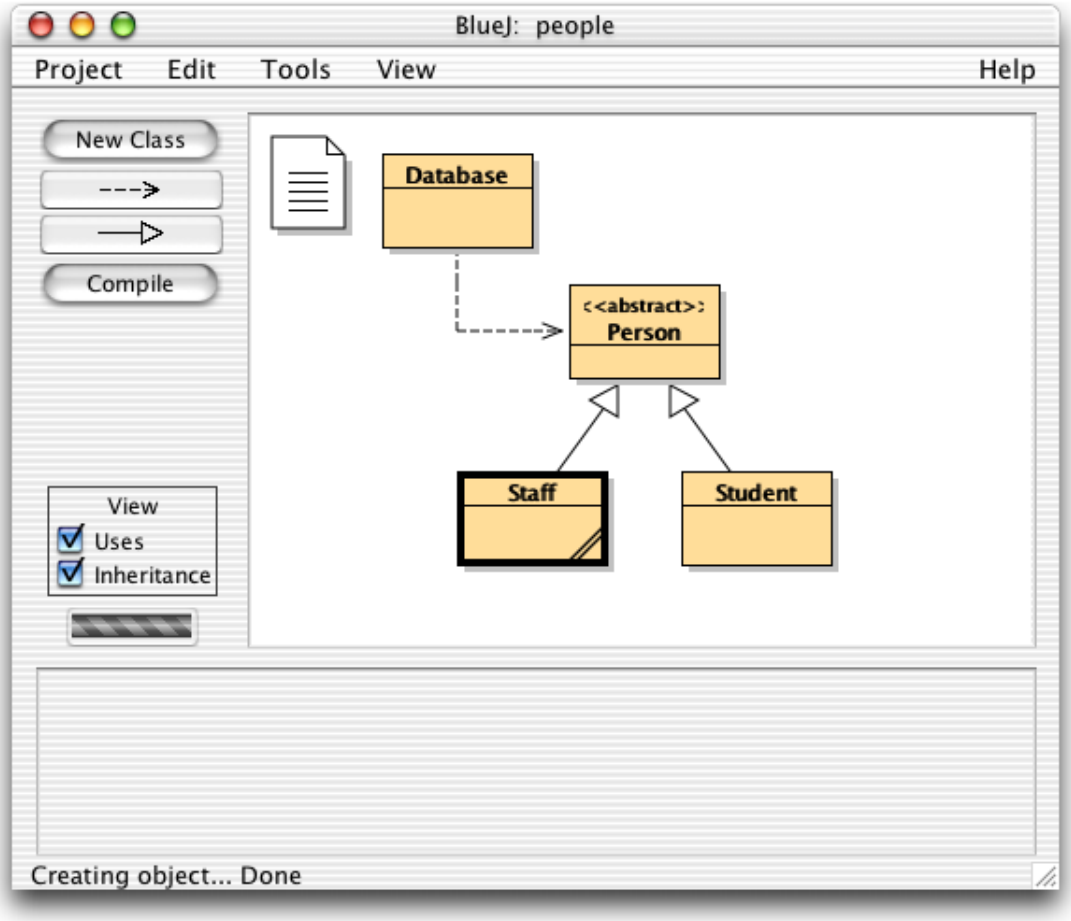

FIG. 1 – La fenêtre principale de BLUEJ

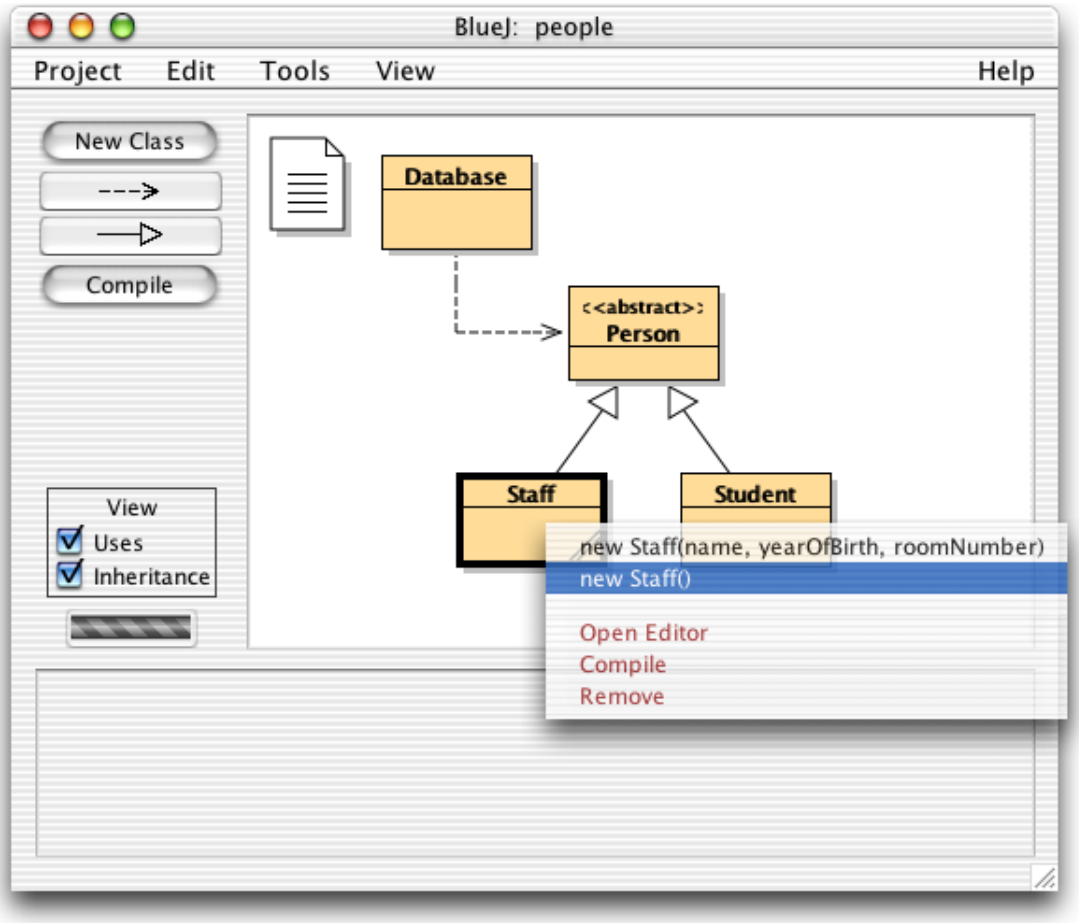

FIG. 2 – Opérations sur les Classes (menu contextuel)

plement en exécutant sa méthode `main`. *Nous reviendrons sur cet aspect ultérieurement. Mais d'abord nous allons faire quelques petites choses plus intéressantes, qu'on ne peut pas faire dans d'autres environnements Java.*

Les rectangles que vous voyez dans la partie centrale de la fenêtre principale (étiquetés `Database`, `Person`, `Staff` et `Student`) sont des icônes représentant les classes impliquées dans cette application. Vous pouvez obtenir un menu avec les méthodes applicables à chaque classe en cliquant sur l'icône de la classe avec le bouton droit de la souris[2](figure 2). Les opérations présentées sont les nouvelles opérations associées à chaque constructeur défini pour cette classe immédiatement suivies par d'éventuelles opérations fournies par l'environnement.

Pour créer un objet `Staff`, il faut cliquer avec le bouton droit sur l'icône Staff (ce qui fait apparaître le menu de la figure 2). Le menu indique deux constructeurs pour créer un objet `Staff`, l'un avec paramètre et l'autre sans. Sélectionnez pour commencer le constructeur

[2]Chaque fois que nous mentionnons un clic droit, les utilisateurs de Macintosh doivent lire Ctrl-clic.

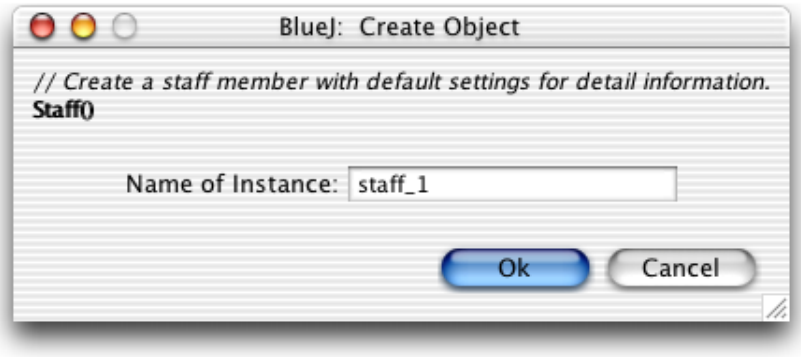

FIG. 3 – Création d'un objet (constructeur sans paramètre)

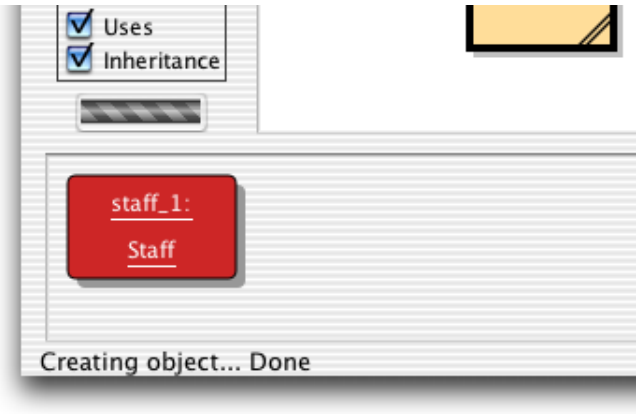

FIG. 4 – Un objet dans le présentoir à objets.

sans paramètre. Le dialogue indiqué figure 3 apparaît alors.

Un nom d'objet est demandé dans la fenêtre de dialogue. Un nom par défaut (`staff_1`) est proposé. Ce nom par défaut suffit dans le cas présent, donc cliquez OK. Un objet `Staff` est alors créé. Une fois l'objet créé, il est placé sur le présentoir à objets (figure 4). La création d'objet se résume donc à ceci : sélectionner un constructeur dans le menu des classes, l'exécuter et obtenir l'objet sur le présentoir.

Vous avez pu remarquer que la classe `Person` est étiquetée `abstract` (c'est une classe abstraite). Vous noterez (si vous essayez) que vous ne pouvez pas créer d'objet de classe abstraite (conformément aux spécifications du langage Java).

3.4 Exécution

Résumé : *Pour exécuter une méthode, il suffit de la sélectionner dans le menu contextuel de l'objet.*

Une fois l'objet créé, vous pouvez exécuter ses opérations publiques (*méthodes* dans la terminologie Java). Cliquez avec le bouton droit sur l'objet ; un menu contextuel apparaît

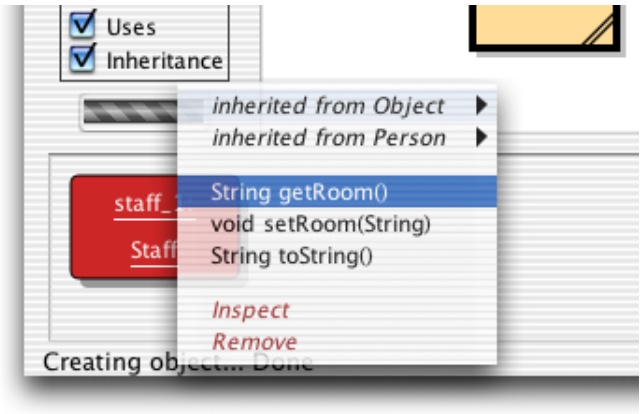

FIG. 5 – Le menu objet.

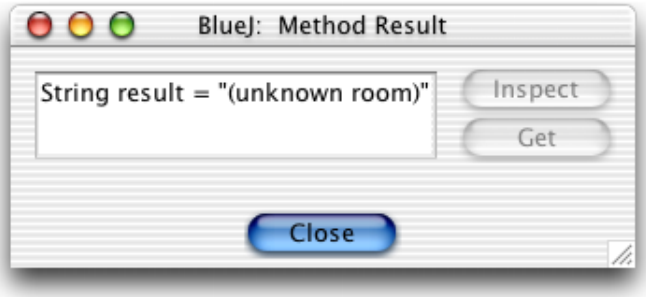

FIG. 6 – Affichage d'un résultat de fonction.

présentant les opérations (figure 5). Le menu énumère les méthodes applicables à cet objet ainsi que deux opérations spéciales fournies par l'environnement (Inspect et Remove). Nous en discuterons plus tard. Focalisons nous d'abord sur les méthodes.

Vous voyez qu'il existe deux méthodes : `setRoom` fixe et `getRoom` renvoie le numéro de bureau pour ce membre de `Staff`. Essayez d'appeler `getRoom`. Sélectionnez tout simplement cette méthode à partir du menu objet et elle sera exécutée. Une fenêtre de dialogue apparaîtra montrant le résultat de l'appel (figure 6). Dans ce cas, vous obtiendrez `(unknown room)` parce qu'aucun nom de bureau n'a été donné pour cette personne.

Les méthodes héritées d'une superclasse sont disponibles grâce à un sous-menu. En tête du menu contextuel objet il y a deux sous-menus, un pour les méthodes héritées de `Object` et un pour celles héritées de `Person` (figure 5). Vous pouvez appeler les méthodes de `Person` (telles que `getName`) en les sélectionnant dans le sous-menu. Essayez de le faire. Vous constaterez que la réponse est également imprécise : `"(unknown name)"` s'affiche car aucun nom de personne n'a été donné.

Maintenant, essayons de donner un numéro de bureau. Ceci permettra de voir comment faire un appel avec paramètres. Les appels à `getRoom` et `getName` ont des valeurs de retour, mais aucun paramètre. Appelez la fonction `setRoom` à partir du menu. Un paramètre vous est alors demandé dans une fenêtre de dialogue (figure 7).

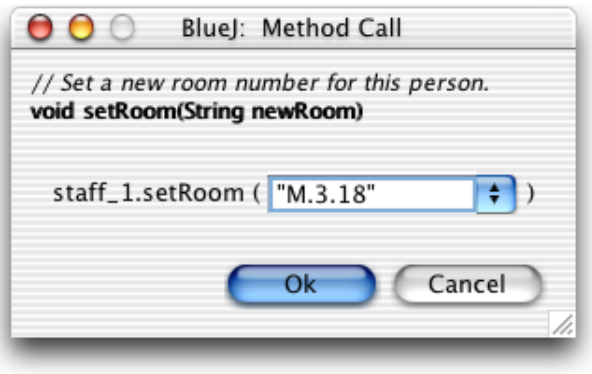

FIG. 7 – Dialogue d'appel de fonctions avec paramètres.

En tête, on trouve l'interface de la méthode à appeler (avec le commentaire et la signature). Ensuite, il y a une zone d'édition où vous pouvez taper le(s) paramètre(s). La signature indiquée en haut précise qu'un paramètre de type `String` est attendu. Entrez le nom du nouveau bureau comme une chaîne (entre des guillemets) dans le champ texte et cliquez sur OK. C'est tout – puisque cette méthode ne renvoie pas de paramètre, il n'y a pas de fenêtre de dialogue pour le résultat. Appelez de nouveau `getRoom` pour vérifier que le bureau a effectivement changé. Exercez-vous à la création d'objets et à l'appel de méthodes. Essayez d'appeler un constructeur avec arguments et appelez quelques méthodes supplémentaires jusqu'à ce que vous soyez à l'aise avec ces manipulations.

3.5 Édition d'une classe

Résumé : *Pour éditer le code source d'une classe, double-cliquer sur son icône.*

Pour l'instant, nous n'avons considéré que l'interface des objets. Il est temps désormais d'en voir le contenu. Vous pouvez voir l'implentation d'une classe en sélectionnant Open Editor à partir des opérations de classe. (Rappelez-vous : un clic droit sur une icône de classe indique les opérations.) Un double clic gauche sur une icône de classe fait la même chose. L'éditeur n'est pas décrit complètement dans ce tutoriel mais il devrait être suffisamment facile à utiliser. Des précisions sur l'éditeur seront fournies ultérieurement. Maintenant, ouvrons l'implantation de la classe `Staff`. Trouvez l'implantation de la méthode `getRoom`. Comme son nom le laisse deviner, elle renvoie le numéro de bureau du membre `staff`. Changeons la méthode en ajoutant le préfixe `"room"` au résultat de la fonction (pour que la méthode renvoie, par exemple, `"room M.3.18"` à la place de `"M.3.18"`). Nous pouvons faire cela en modifiant la ligne

```
return room ;
```

en

```
return "room " + room ;
```

BLUEJ acceptant du Java standard, il n'y a aucune contrainte particulière dans la façon d'implanter les classes.

3.6 Compilation

Résumé : *Pour compiler une classe, cliquer sur le bouton* Compile *de l'éditeur. Pour compiler un projet, cliquer sur le bouton* Compile *dans la fenêtre projet.*

Après avoir modifié le texte (et avant de faire quoi que ce soit d'autre), ayez une vue d'ensemble du projet (grâce à la fenêtre principale). Vous constaterez que l'icône de la classe `Staff` a changé : elle est désormais hachurée. Cette apparence hachurée signale toutes les classes qui n'ont pas été compilées depuis leur dernière modification. Revenons maintenant à l'éditeur.

> NB : *Vous pouvez vous étonner de ne pas avoir trouvé de classes hachurées lorsque vous avez ouvert ce projet pour la première fois. C'est tout simplement parce que les classes du projet* `people` *ont été compilées avant leur distribution. Mais la plupart du temps, les projets de* BlueJ *sont distribués non compilés et, par conséquent, attendez-vous à trouver la plupart des icônes de classes hachurées lorsque vous ouvrirez pour la première fois un projet.*

Dans la barre d'outils de l'éditeur on trouve plusieurs fonctions souvent utilisées. L'une d'elles est Compile. Cette fonction vous permet de compiler directement la classe en cours d'édition. Appuyez maintenant sur le bouton Compile. Si vous n'avez fait aucune erreur, un message indiquant que la classe a été compilée doit apparaître en bas de la fenêtre d'édition. Si vous avez fait une erreur de syntaxe, la ligne erronée apparaît en surbrillance et un message d'erreur s'affiche dans la zone information. (si la compilation marche du premier coup, essayez d'introduire volontairement une erreur de syntaxe, par exemple en omettant un point-virgule, pour voir comment elle est rapportée). Après avoir compilé une classe avec succès, fermez l'éditeur.

> NB : *Il n'est pas nécessaire de sauver explicitement une classe. Le code source d'une classe est en effet automatiquement sauvegardé quand c'est utile (c'est-à-dire quand on ferme l'éditeur ou juste avant de compiler une classe). Vous pouvez néanmoins sauver explicitement (il y a une fonction pour cela dans le menu* class *de l'éditeur), mais cela ne se justifie que lorsque votre système est instable, sujet à des réinitialisations fréquentes, et que l'éventuelle perte de votre travail vous inquiète.*

La barre d'outils de la fenêtre projet possède également un bouton Compile. Cette compilation entraîne la compilation de tout le projet. (En fait, seules les classes modifiées sont concernées et recompilées, dans le bon ordre.) Essayez ceci en modifiant une ou deux classes (de sorte qu'au moins deux classes hachurées apparaissent dans le diagramme de classes) puis cliquez sur le bouton Compile. Si une erreur est détectée dans l'une des classes compilées, l'éditeur s'ouvre et indique l'emplacement et la nature de l'erreur. Vous pourrez constater que le présentoir à objet est à nouveau vide. Les objets sont en effet retirés après tout changement d'implantation.

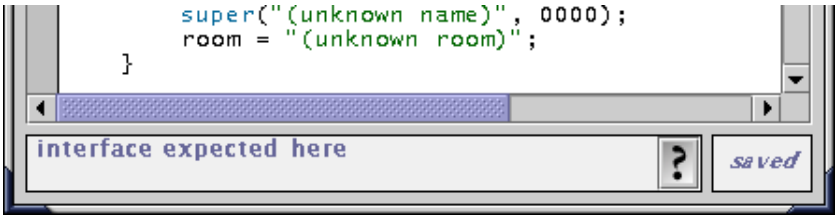

FIG. 8 – Une erreur de compilation et le bouton d'aide.

3.7 Aide pour les erreurs de compilation

Résumé : *Pour obtenir de l'aide au sujet d'une erreur de compilation, cliquer sur le point d'interrogation situé à proximité du message d'erreur.*

Très fréquemment, les étudiants débutants ont des difficultés avec les messages engendrés par le compilateur. Nous essayons d'apporter une aide.

Ouvrez à nouveau l'éditeur, introduisez une erreur dans le code source, puis compilez. Un message d'erreur doit apparaître dans la zone d'information de l'éditeur. A l'extrémité droite de cette zone d'information il y a un point d'interrogation sur lequel vous pouvez cliquer pour obtenir des précisions sur la nature de l'erreur (figure 8).

Au stade actuel de développement de BLUEJ, tous les messages d'erreur n'ont pas un texte d'aide associé. Certains restent à écrire, mais de nombreuses erreurs sont déjà explicitées. Les autres seront décrites et incluses dans une future distribution de BLUEJ.

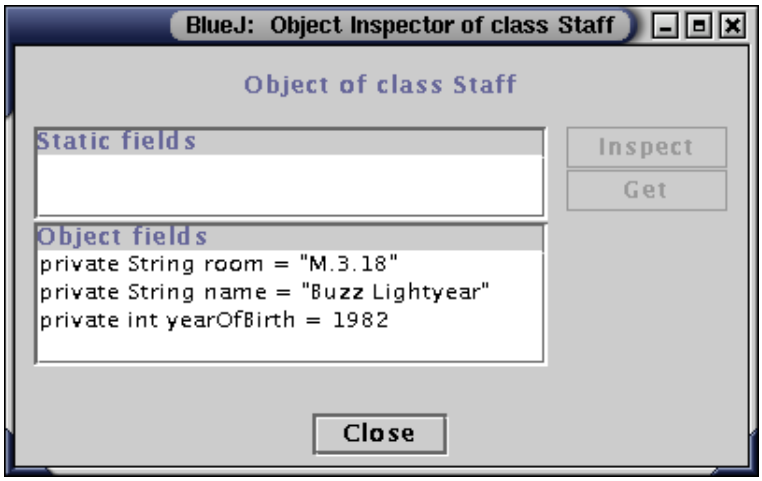

FIG. 9 – La fenêtre d'inspection

4 Pour aller un peu plus loin...

Dans cette section, nous allons voir quelques possibilités supplémentaires de l'environnement. Même si ces possibilités ne sont pas essentielles, elles sont très fréquemment utilisées.

4.1 Inspection

Résumé : *L'inspection d'objet permet un débogage simple en montrant l'état interne d'un objet.*

Quand vous avez exécuté une méthode d'un objet, vous avez peut-être remarqué l'opération Inspect qui est disponible sur les objets en plus des méthodes définies par l'utilisateur (figure 5). Cette opération permet de vérifier l'état des variables d'instance (les champs) des objets. Essayez de créer un objet avec une valeur quelconque (par exemple, un objet `Staff` avec le constructeur qui prend des paramètres). Ensuite, sélectionnez Inspect depuis le menu de l'objet. Une fenêtre apparaît ; elle affiche les champs de l'objet, leur type et leur valeur (figure 9).

L'inspection est utile pour vérifier rapidement si une méthode modifiant l'état d'un objet s'est exécutée correctement. L'inspection est également un outil simple pour le débogage.

Dans l'exemple `Staff`, tous les champs ont des types simples (soit des types scalaires primitifs, soit des chaînes de caractères). La valeur de ces types est affichée directement. Vous pouvez immédiatement voir si le constructeur a réalisé les bonnes affectations.

Pour des classes plus compliquées, la valeur des champs peut être une référence vers un autre objet défini par l'utilisateur. Nous allons utiliser un autre projet pour illustrer ce cas. Ouvrez le projet `people2` qui est aussi inclus dans la distribution BLUEJ. Le présentoir de `people2` est donné par la figure 10. Comme vous pouvez le voir, ce deuxième exemple a une classe `Address` en plus des classes vues précédemment. L'un des champs de la classe

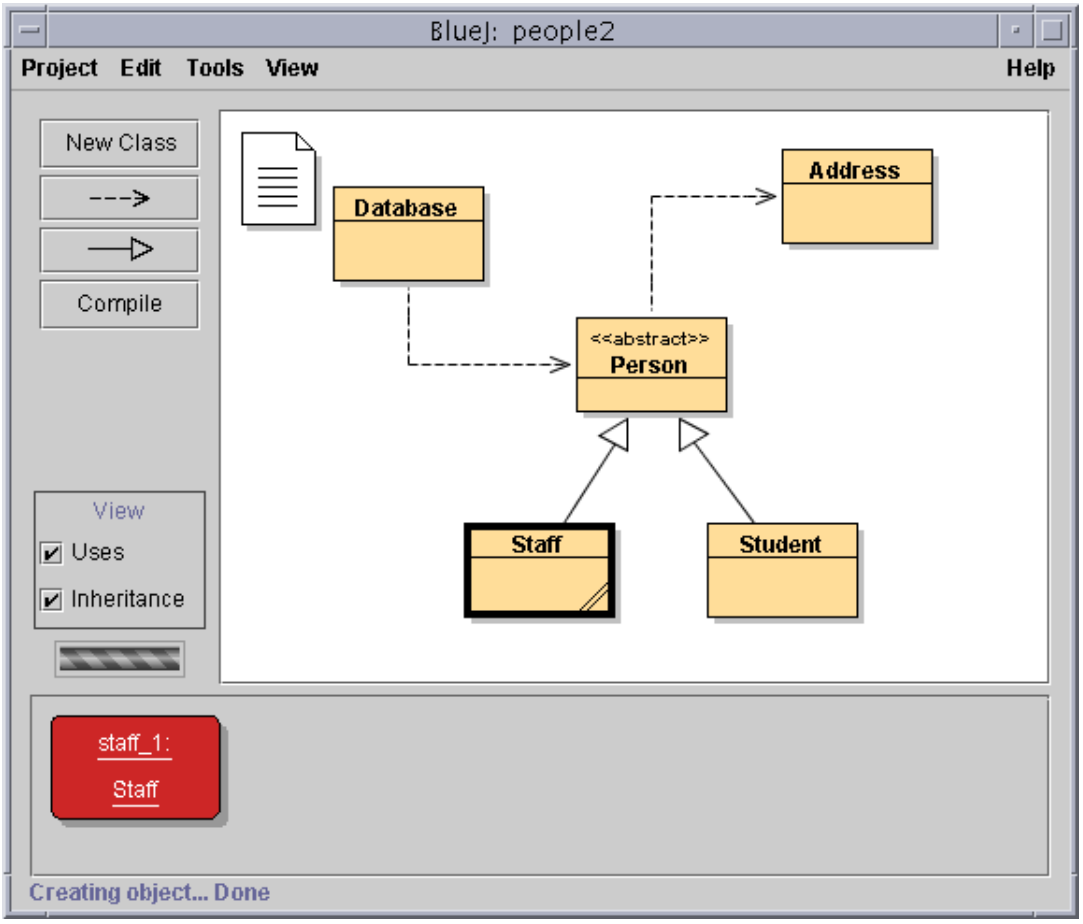

FIG. 10 – La fenêtre du projet `people2`

`Person` est du type `Address`, type défini par l'utilisateur.

Pour tester l'inspection avec des champs contenant des objets, créez un objet `Staff` et exécutez la méthode `setAddress` de cet objet (vous la trouverez dans le sous-menu Person). Entrez une adresse. En interne, le code de `Staff` crée un objet de la classe `Address` et le stocke dans son champ `address`.

Maintenant, inspectez l'objet `Staff`. Le résultat de cette inspection est la fenêtre de la figure 11. Les champs de l'objet `Staff` incluent maintenant l'adresse. Comme vous pouvez le constater, sa valeur est affichée comme `"<object reference>"` (référence d'un objet) ; parce que c'est un type complexe, défini par l'utilisateur, sa valeur ne peut pas être affichée directement dans la liste. Pour examiner la représentation de cet objet `Address`, sélectionnez le champ address dans la liste et cliquez le bouton Inspect dans la fenêtre. (Vous pouvez également double-cliquer sur le champ `address`). Une nouvelle fenêtre d'inspection est ouverte, elle montre la représentation de l'objet `Address` (figure 12).

Si le champ sélectionné est public alors, au lieu de cliquer Inspect, vous pouvez aussi sélectionner le champ `address` et cliquer le bouton Get. Cette opération place l'ob-

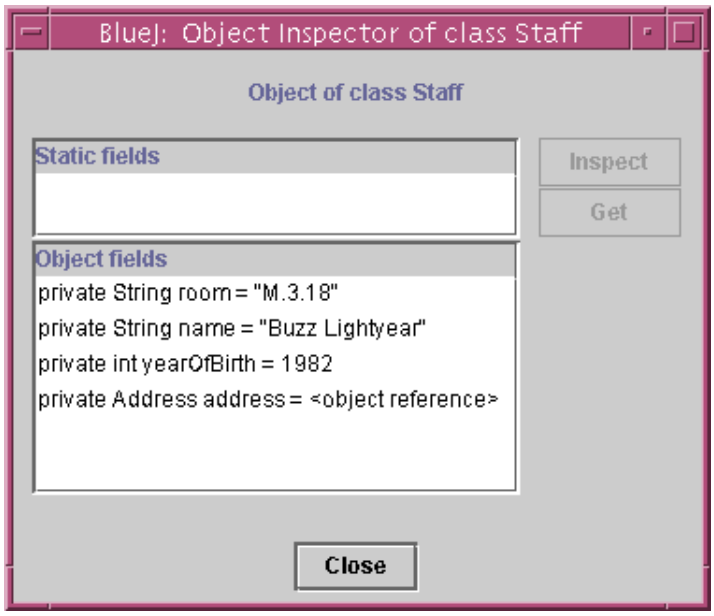

FIG. 11 – *Inspect*ion avec une référence sur un objet

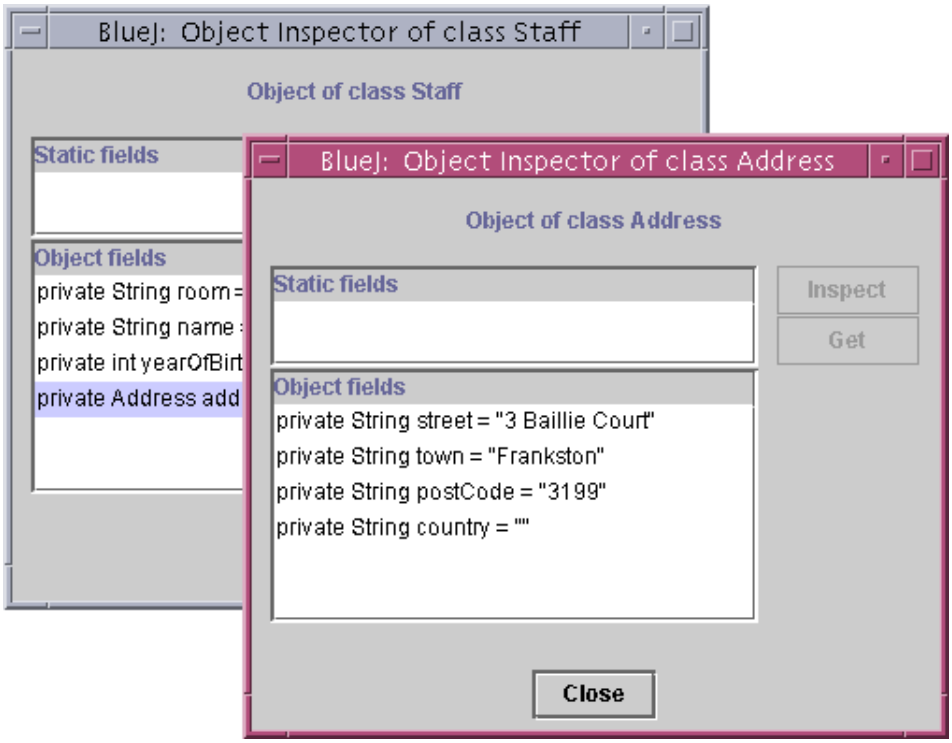

FIG. 12 – Inspection de l'état interne d'un objet

jet sélectionné sur le présentoir à objets. Vous pouvez alors l'examiner en exécutant ses méthodes.

4.2 Passage d'objets en paramètre

Résumé : *Un objet peut être passé en paramètre lors d'un appel de méthode en cliquant sur son icône.*

Les objets peuvent être passés en paramètre de méthodes d'autres objets. Essayons sur un exemple. Créez un objet de la classe `Database`. (Vous noterez que la classe `Database` possède un seul constructeur qui n'a pas de paramètre, aussi la construction d'un objet est directe.) L'objet `Database` permet de stocker une liste de personnes. Il a des méthodes pour ajouter une personne et pour afficher toutes les personnes actuellement stockées. (L'appeler `Database`, c'est-à-dire base de données, est quelque peu exagéré !)

Si vous n'avez pas déjà un objet `Staff` ou `Student` sur le présentoir à objets, créez l'un ou l'autre. Pour la suite, vous avez besoin d'avoir en même temps sur le présentoir un objet `Database`, et soit un objet `Staff`, soit un objet `Student`.

Maintenant, exécutez la méthode `addPerson` sur l'objet `Database`. (Rappel : la classe `Person` est abstraite, aussi il n'y a pas d'objet qui soit directement du type `Person`. Mais, grâce au sous-typage, des objets `Student` et `Staff` peuvent être substitués à des objets `Person`. Ainsi, il est autorisé de passer un `Student` ou un `Staff` là où un objet `Person` est attendu.) Pour passer un objet du présentoir en tant que paramètre lors de l'appel que vous êtes en train de faire, vous pouvez saisir son nom dans le champ paramètre ou, en guise de raccourci, simplement cliquez sur l'objet. Ceci entre son nom dans la zone de saisie de l'appel de méthode. Cliquez OK et l'appel est effectué. Puisqu'il n'y a pas de valeur de retour pour cette méthode, le résultat n'est pas visible immédiatement. Vous pouvez exécuter la méthode `listAll` sur l'objet `Database` pour vérifier que l'opération a réellement été exécutée. La méthode `listAll` affiche les informations de la personne sur la sortie standard. Nous remarquerez que le terminal s'ouvre automatiquement pour afficher le texte.

Essayez ceci de nouveau avec plusieurs personnes entrées dans la « base de données ».

5 Création d'un nouveau projet

Ce chapitre expliquera rapidement comment créer un nouveau projet.

5.1 Création du répertoire de projet

Résumé : *Pour créer un projet, choisir* New *dans le menu* Project.

Pour créer un nouveau projet, choisissez Project > New du menu. Une boîte de sélection s'ouvre et vous permet de spécifier le nom et l'emplacement du nouveau projet. Essayez. Vous pouvez choisir un nom quelconque pour votre projet. Après avoir cliqué OK, un répertoire qui porte le nom choisi sera créé, et la fenêtre principale montrera le nouveau projet vide.

5.2 Création des classes

Résumé : *Pour créer une classe, cliquer le bouton* New Class *et spécifier un nom.*

Vous pouvez maintenant créer vos classes en cliquant le bouton New Class de la barre d'outils du projet. Il vous sera demandé de fournir un nom pour la classe – ce nom doit être un identificateur Java valide. Vous pouvez aussi choisir parmi quatre types de classes : `abstract`, `interface`, `applet` ou « standard ». Le choix détermine quel squelette sera créé pour la classe. Vous pouvez changer le type de classe plus tard en modifiant le code source (par exemple, en ajoutant le mot-clé `abstract` dans le code). Après avoir créé une classe, celle-ci est représentée par une icône dans le diagramme. Si ce n'est pas une classe standard, le type (`interface`, `abstract`, ou `applet`) est indiqué par une icône. Quand vous ouvrez l'éditeur pour une nouvelle classe il affiche le squelette par défaut qui a été créé pour votre classe – ceci vous permettra de démarrer facilement l'écriture du code. Le code par défaut est syntaxiquement correct. Il peut être compilé (mais ne fait rien de significatif). Essayez de créer plusieurs classes et de les compiler.

5.3 Création des dépendances

Résumé : *Pour créer une flèche, cliquer le bouton correspondant et faire glisser la flèche sur le diagramme, ou tout simplement modifier le code dans l'éditeur. Le diagramme de classe montre les dépendances entre les classes sous la forme de flèches.*

Les relations d'héritage (`extends` ou `implements`) sont représentées comme des flèches doubles et les relations *uses* comme des flèches simples. Vous pouvez ajouter des dépendances soit graphiquement (directement sur le diagramme) ou textuellement dans le code source. Si vous ajoutez une flèche graphique, le source est automatiquement mis à jour ; inversement, si vous ajoutez une dépendance dans le source, le diagramme est mis à jour.

Pour ajouter la flèche de manière graphique, cliquez sur le bouton approprié (double flèche pour `extends` ou `implements`, simple flèche pour *uses*) et faites glisser la flèche d'une classe vers l'autre.

L'ajout d'une flèche d'héritage provoque l'insertion d'une définition `extends` ou `implements` dans le code source de la classe (suivant que la cible est une classe ou une interface).

Ajouter une flèche *uses* ne modifie pas immédiatement le code (sauf dans le cas où la classe provient d'un autre paquetage, dans ce cas une déclaration `import` est engendrée, mais nous ne l'avons pas encore vu dans nos exemples). Avoir dans le diagramme une flèche *uses* qui pointe vers une classe qui n'est pas effectivement utilisée dans son code source va générer plus tard un avertissement qui indique qu'une relation *uses* a été déclarée, mais la classe n'a pas été utilisée.

Ajouter une flèche de manière textuelle est facile : tapez simplement le code nécessaire. Dès que le fichier de la classe est enregistré, le diagramme est mis à jour. (Et souvenez-vous que la fermeture de l'éditeur réalise un enregistrement automatique).

5.4 Suppression d'éléments

Résumé : *Pour supprimer une classe, choisir la fonction Remove du menu contextuel. Pour supprimer une flèche, choisir Remove du menu Edit et cliquer sur la flèche.*

Pour supprimer une classe du diagramme, sélectionnez la classe et ensuite Remove class du menu Edit. Vous pouvez aussi sélectionner Remove du menu contextuel de la classe. Pour supprimer une flèche, sélectionnez Remove Arrow du menu et ensuite sélectionnez la flèche que vous voulez supprimer.

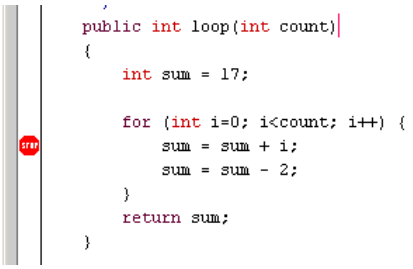

FIG. 13 – Un point d'arrêt

6 Mise au point

Ce chapitre présente les aspects essentiels de l'outil de mise au point de programmes (*debugging*) offert par BLUEJ. En général, les enseignants en programmation considèrent que l'utilisation de ce type d'outil dès l'initiation à la programmation est utile, mais qu'ils ne disposent pas de suffisamment de temps pour présenter un outil compliqué supplémentaire.

L'outil de mise au point doit donc être aussi simple que possible. L'objectif choisi dans le cadre de BLUEJ était que l'outil puisse être présenté en 15 minutes et qu'il puisse être utilisé par les étudiants sans informations supplémentaires. À vous de juger si cet objectif a été atteint.

Les fonctionnalités traditionnelles des outils de mise au point ont donc été réduites aux tâches suivantes :

– positionnement de points d'arrêt ;
– exécution pas à pas du programme ;
– consultation du contenu des variables.

Pour commencer, ouvrez le projet `debugdemo` qui est dans le répertoire d'exemples de la distribution. Ce projet contient quelques classes dont l'objectif est uniquement la présentation des fonctionnalités de l'outil – elles n'ont pas de sens en dehors de cet objectif.

6.1 Positionnement de points d'arrêt

Résumé : *Pour positionner un point d'arrêt, cliquer sur la zone des points d'arrêt à gauche du texte dans l'éditeur (voir figure 13).*

Le positionnement d'un point d'arrêt permet d'interrompre l'exécution à un endroit précis du programme.

Dans l'éditeur, la zone des points d'arrêt se trouve à gauche du texte (voir figure 13)". Vous pouvez positionner un point d'arrêt en cliquant dans cette zone. Un panneau « stop » apparaît alors pour indiquer la position du point d'arrêt.

Essayez ceci. Ouvrez la classe `Demo`, recherchez la méthode `loop`, positionnez un point d'arrêt dans la boucle `for`. Le panneau « stop » doit apparaître dans votre éditeur.

L'exécution s'arrête lorsqu'une ligne de code à laquelle est attaché un point d'arrêt est atteinte.

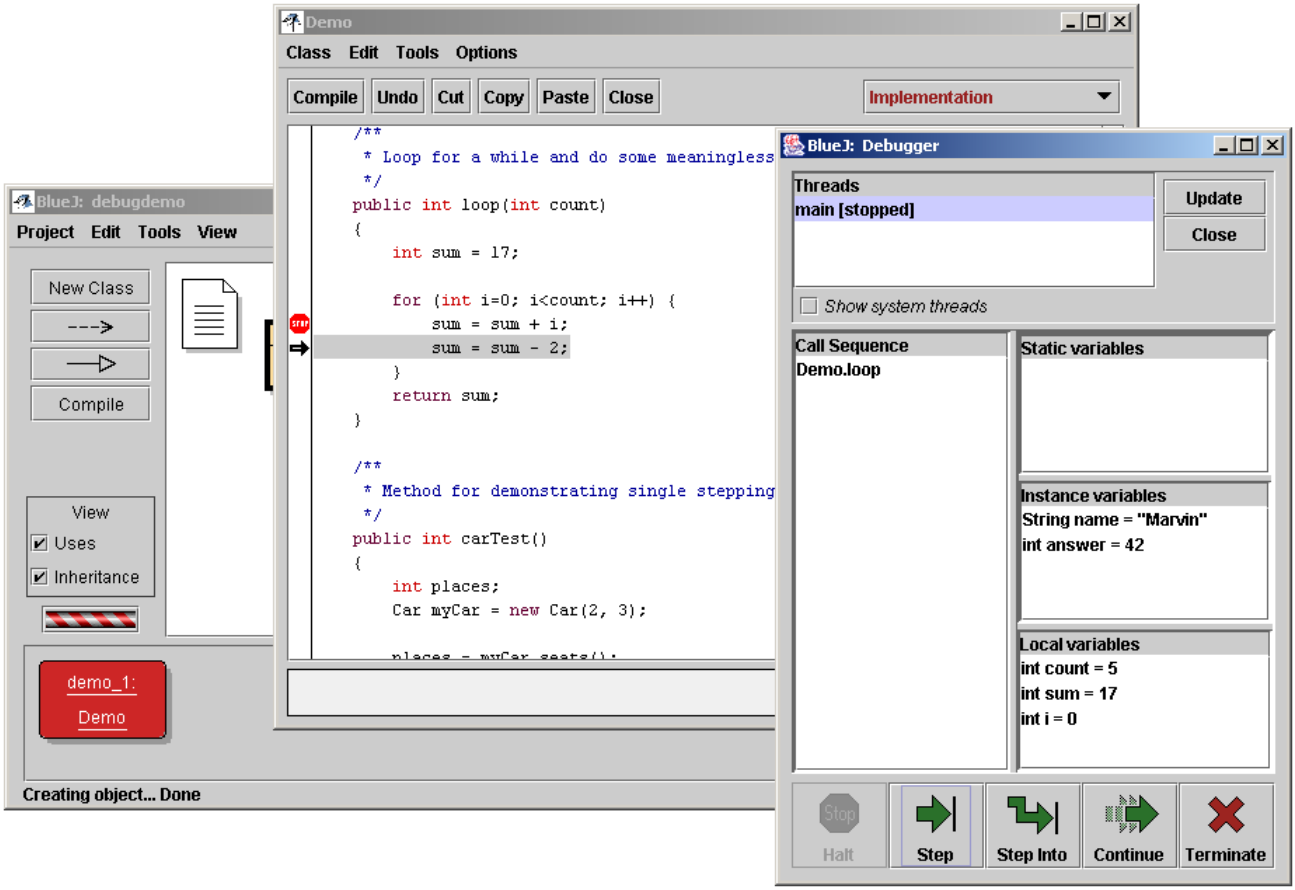

FIG. 14 – Fenêtre de mise au point

Essayez ceci : créez un objet de la classe `Demo` puis appelez la méthode `loop` avec un paramètre de valeur `10` par exemple. Dès que le point d'arrêt est atteint, la fenêtre d'édition contenant la ligne courante apparaît, suivie par la fenêtre de mise au point. Celle-ci est semblable à la figure 14.

La ligne qui doit être exécutée par la suite est mise en évidence dans l'éditeur (en vidéo inverse). L'exécution a été arrêtée juste avant cette ligne.

6.2 Exécution pas à pas

Résumé : *Pour exécuter un programme ligne par ligne, utiliser les boutons* Step *et* Step Into *de l'outil de mise au point.*

Lorsque l'exécution est interrompue (ce qui confirme que la méthode a réellement été exécutée et que ce point du code a réellement été atteint), vous pouvez exécuter le programme pas à pas et suivre le déroulement de l'exécution. Pour cela, cliquez plusieurs fois sur le bouton Step dans la fenêtre de mise au point. Vous remarquerez que la ligne du

programme mise en évidence dans l'éditeur change (la ligne en vidéo inverse se déplace au fur et à mesure de l'exécution). À chaque fois que vous cliquez sur le bouton Step, une seule ligne de code est exécutée puis l'exécution s'interrompt de nouveau.

Essayons sur une autre méthode. Positionnez un point d'arrêt dans la méthode `carTest()` de la classe `Demo` sur la ligne contenant `places = myCar.seats() ;`.

Appelez cette méthode. Quand le point d'arrêt est atteint, le système est prêt à exécuter la ligne contenant l'appel à la méthode `seats()` de la classe `Car`. En cliquant sur le bouton Step, l'ensemble de la ligne est exécutée (sans détailler les instructions composant la méthode). Essayons maintenant le bouton Step Into. Celui-ci permet d'exécuter le corps de la méthode pas à pas (et non en un seul pas comme précédemment). Dans ce cas, vous êtes « transporté » dans la méthode `seats()` de la classe `Car`. Vous pouvez maintenant progresser ligne par ligne jusqu'à la fin de la méthode et retourner à la ligne appelante. Notez comment l'outil de mise au point affiche les changements.

Les commandes Step et Step Into se comportent de la même manière si la ligne courante ne contient pas d'appels de méthodes.

6.3 Inspection des variables

Résumé : *L'accès au contenu des variables est extrèmement simple – il est affiché automatiquement par l'outil de mise au point.*

Lors de la mise au point de vos programmes, il est essentiel de pouvoir accéder au contenu des objets (les paramètres et variables locales des méthodes ou bien dans les attributs d'instances et de classes).

Cela se fait très simplement – vous avez déjà étudié la plupart des mécanismes nécessaires. Vous n'avez pas à utiliser de commandes spécifiques : la valeur des attributs de classe et d'instance de l'objet courant et des paramètres et variables locales de la méthode courante sont toujours affichés et mis à jour systématiquement.

Vous pouvez choisir des méthodes dans la séquence des appels pour afficher les contenus des attributs et variables des autres objets actifs et méthodes. Par exemple, positionnez un autre point d'arrêt dans la méthode `carTest()` puis lancez l'exécution. La séquence d'appel est affichée à la gauche de l'outil de mise au point. Il affiche actuellement : `Car.seats` `Demo.carTest`.

Cela signifie que la méthode `Car.seats` a été appelée par la méthode `Demo.carTest`. Vous pouvez sélectionner la méthode `Demo.carTest` dans cette liste pour consulter le source de cette méthode et afficher le contenu actuel de ses variables.

Avancez après la ligne qui contient l'instruction `new Car()`. Notez que la valeur de la variable locale `myCar` est étiquetée en tant que `<object reference>`. Toutes les instances de classes (sauf pour la classe `String`) sont affichées ainsi. Affichez le contenu de l'instance en double-cliquant sur la variable associée. Cela fait apparaître la fenêtre d'inspection du contenu d'un objet qui a déjà été décrite précédemment (voir chapitre 4). Il n'y a aucune différence réelle entre l'accès au contenu des objets dans ce contexte et l'accès au contenu des objets via le présentoir à objets.

6.4 Arrêt et fin

Résumé : *Les boutons* Halt *et* Terminate *peuvent être utilisés pour arrêter l'exécution temporairement ou définitivement.*

L'exécution d'un programme peut être très longue, suffisamment pour que se pose la question d'un fonctionnement correct extrêmement long ou d'un calcul infini. Il est possible d'obtenir une réponse simplement. Appelez la méthode `longloop()` de la classe `Demo`. Celle-ci a un temps d'exécution relativement long.

Ouvrez la fenêtre de mise au point si elle n'est pas déjà affichée (cliquer sur le symbole en rotation qui indique que la machine est active durant l'exécution permet de faire apparaître l'outil de mise au point).

Cliquez ensuite sur le bouton Halt. L'exécution est interrompue exactement comme si elle avait atteint un point d'arrêt. Il est maintenant possible d'exécuter quelques pas, de consulter la valeur des variables pour s'assurer que tout se déroule correctement et qu'il ne manque que du temps pour que le calcul se termine. Vous pouvez poursuivre le calcul (bouton Continue) puis l'arrêter de nouveau (bouton Halt) et ceci plusieurs fois pour mesurer la vitesse du calcul. Si vous ne souhaitez pas poursuivre le calcul (par exemple, après avoir découvert qu'il s'agit vraiment d'une boucle infinie) vous pouvez cliquer sur le bouton Terminate pour arrêter définitivement l'exécution. Cette fonction ne doit être utilisée que dans des cas extrêmes car des objets peuvent se retrouver dans un état incohérent suite à l'interruption en cours d'exécution d'une méthode.

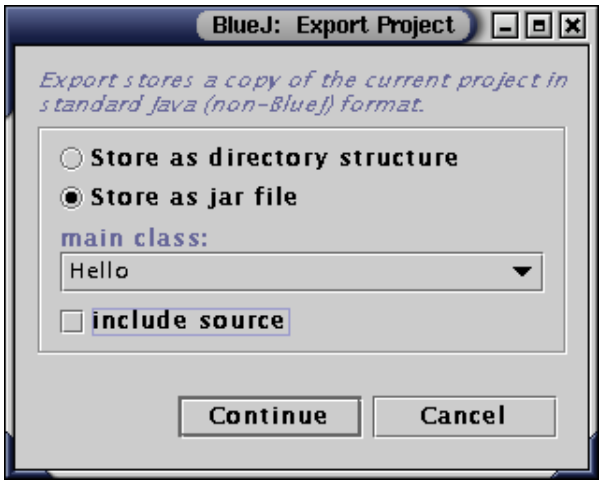

FIG. 15 – La boîte de dialogue *Export*

7 Création d'applications autonomes

Résumé : *Pour créer une application autonome, utiliser Project > Export...*

BLUEJ peut créer des fichiers/archives jar exécutables. Un fichier jar exécutable peut être exécuté en double-cliquant sur le fichier (sur certains systèmes, par exemple Windows et MacOS X), ou en utilisant la commande en ligne `java -jar <file-name>.jar` (invite Unix ou DOS). Essayons cela sur le projet exemple `hello`. Ouvrez le projet (il est dans le répertoire `examples`). Assurez-vous que le projet est compilé. Choisissez la fonction Export du menu Project. Une boîte de dialogue apparaît où vous indiquez le format de sauvegarde (figure 15). Choisissez jar file pour créer un fichier jar exécutable. Pour rendre exécutable le fichier jar, vous devez aussi indiquer une classe principale. Cette classe doit définir une méthode `main` valide (avec la signature `public static void main(String[] args)`).

Dans notre exemple, le choix de la classe principale est facile : il n'y a qu'une seule classe. Choisissez `Hello` dans le menu contextuel. Si vous avez d'autres projets, choisissez la classe qui contient la méthode `main` que vous voulez exécuter. Généralement, on n'inclut pas les sources dans les fichiers exécutables, mais cela est possible, si vous souhaitez distribuer aussi vos sources. Cliquez sur Continue. Ensuite, vous avez une boîte de dialogue pour saisir le nom du fichier jar que vous voulez créer. Tapez `hello` et cliquez sur Create. La création du fichier jar exécutable est terminée. Si l'application utilise une interface graphique, vous pouvez double-cliquer sur le fichier jar. Comme l'exemple utilise une interface textuelle, il doit être lancé depuis la fenêtre d'un terminal. Ouvrez un terminal ou une fenêtre DOS. Allez dans le répertoire où le fichier jar a été sauvé (vous devez voir un fichier `hello.jar`). En supposant que Java est correctement installé sur votre système, vous pouvez démarrer le fichier jar en tapant `java -jar hello.jar`.

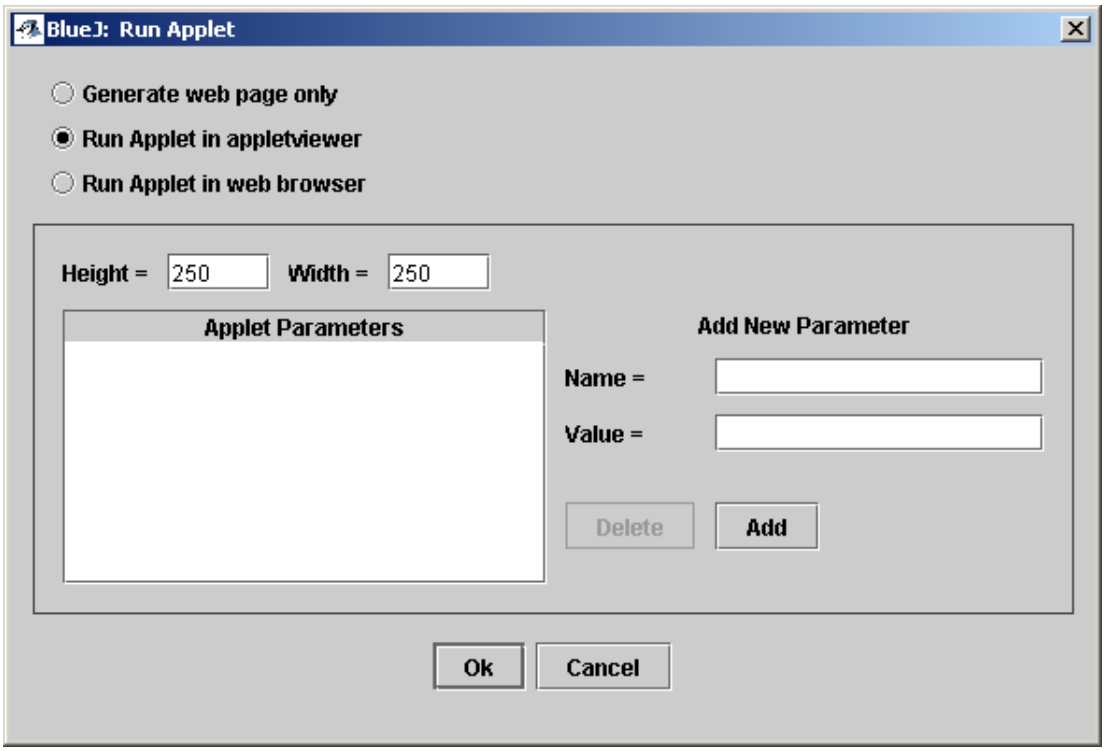

FIG. 16 – La fenêtre de saisie *Run Applet*

8 Création d'appliquettes

8.1 Exécution d'une appliquette

Résumé : *Pour exécuter une appliquette, choisir* Run Applet *dans le menu contextuel de l'appliquette.*

BLUEJ permet de créer et d'exécuter aussi bien des appliquettes que des applications. Nous avons inclus une appliquette dans le répertoire d'exemples de la distribution. Tout d'abord, nous allons tenter d'exécuter une appliquette. Ouvrez le projet `appletdemo` des exemples.

Vous constaterez que ce projet ne possède qu'une unique classe appelée `CaseConverter`. L'icône de la classe est celui d'une appliquette (avec l'étiquette `<<applet>>`). Après avoir compilé, sélectionnez la commande Run Applet du menu contextuel de la classe. Une fenêtre de saisie s'ouvre vous permettant d'effectuer certains choix (Figure 16).

Vous pouvez alors choisir d'exécuter l'appliquette dans un navigateur ou dans un visualisateur d'appliquettes (ou seulement de générer la page web sans l'exécuter). Conservez la configuration par défaut et cliquez sur OK. Après quelques secondes, un visualisateur d'appliquettes devrait s'ouvrir en affichant l'appliquette `CaseConverter`.

Le visualisateur d'appliquettes est installé avec votre JDK ce qui assure d'avoir toujours la même version que celle de votre compilateur Java. Vous rencontrerez ainsi généralement

moins de problèmes avec lui que lors de l'exécution dans des navigateurs. Votre navigateur peut exécuter une version différente de Java et selon la version de ce navigateur, cela peut poser des problèmes. Cela devrait cependant fonctionner avec la plupart des navigateurs habituels.

Sous les systèmes Windows et MacOS, BlueJ utilise le navigateur défini comme navigateur par défaut. Sous Unix, le navigateur est spécifié dans le fichier de configuration de BlueJ.

8.2 Création d'une appliquette

Résumé : *Pour créer une appliquette, cliquer sur le bouton* New Class *et choisir* Applet *comme type de classe.*

Après avoir examiné comment exécuter une appliquette, nous allons maintenant en créer une.

Créez une nouvelle classe de type `Applet` (vous pouvez sélectionner le type dans le choix New Class). Compilez puis exécutez l'appliquette. C'est gagné ! Ce n'était pas trop dur ?

Les appliquettes (comme les autres classes) sont générées avec un squelette de classe par défaut contenant du code valide. En ce qui concerne les appliquettes, ce code montre une appliquette simpliste affichant deux lignes de texte. Vous pouvez ensuite ouvrir l'éditeur pour insérer votre propre code dans celui de l'appliquette.

Vous y trouverez toutes les méthodes classiques des appliquettes avec un commentaire les caractérisant. Le code exemple est dans la méthode `paint`.

8.3 Test d'une appliquette

Dans certains cas, il peut être utile de créer un objet `Applet` dans le présentoir à objets (comme pour les classes normales). BlueJ offre cette possibilité – le constructeur est présenté dans le menu contextuel des appliquettes. À partir du présentoir, vous ne pouvez pas exécuter l'appliquette complète mais seulement appeler certaines méthodes. Cela peut être utile pour tester des méthodes seules que vous avez écrites comme une partie de l'implantation de votre appliquette.

9 Autres opérations

9.1 Ouverture de paquetages extérieurs à BlueJ

Résumé : *Les paquetages extérieurs à* BlueJ *peuvent être ouverts avec la commande Project > Open Non Bluej*

BlueJ vous permet d'ouvrir des paquetages existants créés à l'extérieur de BlueJ. Pour faire ceci, sélectionnez Projet > Open Non BlueJ à partir du menu. Sélectionnez le répertoire qui contient les fichiers source Java, et cliquez sur le bouton Open in BlueJ. Le système vous demandera de confirmer que vous voulez ouvrir ce répertoire.

9.2 Ajout de classes existantes au projet

Résumé : *Des classes venant de l'extérieur peuvent être copiées dans le projet en utilisant la commande Add Class from File.*

Souvent, vous devez utiliser une classe que vous avez récupérée à un autre endroit dans votre projet BlueJ. Par exemple, un enseignant peut donner une classe Java aux étudiants pour qu'ils l'utilisent dans un projet. Vous pouvez inclure facilement une classe existante dans votre projet en sélectionnant Edit > Add Class from File... à partir du menu. Ceci vous permettra de sélectionner un fichier source Java (i.e. avec l'extension `.java`) à importer.

Quand la classe est importée dans le projet, elle est copiée et enregistrée dans le répertoire du projet courant. L'effet est exactement le même que si vous aviez créé la classe et écrit tout son code source.

Une autre possibilité est d'ajouter le fichier source de la nouvelle classe au répertoire du projet à l'extérieur de BlueJ. À la prochaine ouverture du projet, la classe sera intégrée dans le diagramme du projet.

9.3 Appel de `main` et d'autres méthodes statiques

Résumé : *Les méthodes statiques peuvent être appelées depuis le menu contextuel de la classe.*

Ouvrez le projet `hello` du répertoire d'exemples. La seule classe du projet (la classe `Hello`) définit la méthode standard `main`.

En cliquant avec le bouton droit sur la classe, vous voyez que le menu inclut non seulement le constructeur de la classe, mais aussi la méthode de classe (statique) `main`. Vous pouvez appeler `main` directement depuis le menu (sans créer l'objet au préalable, comme ce serait le cas pour une méthode d'instance (non-statique)).

Toutes les méthodes statiques peuvent être appelées de cette manière. La méthode `main` attend un tableau de chaînes de caractères comme argument. Vous pouvez définir un tableau de chaînes avec la syntaxe Java standard pour les constantes. Par exemple, vous pouvez passer `{"un", "deux", "trois" }` (sans oublier les accolades) à la méthode. Essayez-donc...

NB : *En Java standard, les tableaux constants ne peuvent pas être utilisés directement comme arguments pour un appel de méthode. Ils servent seulement comme valeurs initiales. En* BlueJ, *pour que ce genre d'appel soit possible, les tableaux constants en paramètre sont possibles.*

9.4 Génération de la documentation

Résumé : *Pour engendrer la documentation d'un projet, sélectionner Project Documentation depuis le menu Tools.*

Vous pouvez engendrer la documentation de votre projet dans le format Javadoc standard depuis BlueJ. Pour faire ceci, sélectionnez Tools > Project documentation depuis le menu. Cette fonction engendre la documentation pour toutes les classes d'un projet depuis le code source de la classe, et ouvre un navigateur pour l'afficher.

Vous pouvez aussi engendrer et visualiser la documentation d'une classe donnée directement dans l'éditeur BlueJ. Pour faire ceci, ouvrez l'éditeur et utilisez le menu dans la barre d'outils de l'éditeur. Sélectionnez Interface au lieu de Implementation. Ceci affiche la documentation en javadoc (de l'interface de la classe) dans l'éditeur.

9.5 Travail avec les bibliothèques

Résumé : *Les classes standard de l'API Java peuvent être visualisées en sélectionnant Help > Java Class Libraries*

Souvent, quand vous écrivez un programme Java, vous devez faire référence aux bibliothèques standard Java. Vous pouvez ouvrir un navigateur affichant la documentation de l'API du JDK en sélectionnant Help > Java Standard Classes depuis le menu (si vous êtes connecté à Internet).

La documentation du JDK peut aussi être installée et utilisée localement (hors-ligne). La section d'aide du site de BlueJ explique comment faire.

9.6 Création d'objets à partir des classes de l'API

Résumé : *Pour créer des objets à partir de classes d'une bibliothèque, sélectionner Tools > Use Library Class*

BlueJ propose une fonction pour créer des instances de classes qui ne font pas partie de votre projet, mais sont définies dans la bibliothèque. Vous pouvez, par exemple, créer des objets de classe `String` ou `ArrayList`. Ceci peut être utile pour tester rapidement ces objets.

Vous pouvez créer un objet de la bibliothèque en sélectionnant Tools > Use Library Class... depuis le menu. Une boîte de dialogue apparaîtra pour vous demander un nom de classe complet, comme `java.lang.String` (vous devez obligatoirement mettre le nom complet, c'est-à-dire comprenant le nom des paquetages englobants).

Le menu associé au champ de saisie de texte comprend les classes utilisées récemment. Quand vous avez saisi le nom d'une classe, appuyez sur Entrée pour afficher tous ses

constructeurs et ses méthodes de classe (statiques) dans une liste. Toutes ces méthodes peuvent être invoquées en les sélectionnant dans la liste.

L'invocation se fait de la même manière que pour tout autre constructeur ou méthode.

10 En guise de résumé

Pour débuter – édition / compilation / exécution

1. Pour ouvrir un projet, sélectionner Open dans le menu Project.
2. Pour créer un objet, sélectionner un constructeur dans le menu contextuel de la classe.
3. Pour exécuter une méthode, il suffit de la sélectionner dans le menu contextuel de l'objet.
4. Pour éditer le code source d'une classe, double-cliquer sur son icône.
5. Pour compiler une classe, cliquer sur le bouton Compile de l'éditeur. Pour compiler un projet, cliquer sur le bouton Compile dans la fenêtre projet.
6. Pour obtenir de l'aide au sujet d'une erreur de compilation, cliquer sur le point d'interrogation situé à proximité du message d'erreur.

Pour aller un peu plus loin...

7. L'inspection d'objet permet un débogage simple en montrant l'état interne d'un objet.
8. Un objet peut être passé en paramètre lors d'un appel de méthode en cliquant sur son icône.

Création d'un nouveau projet

9. Pour créer un projet, choisir New dans le menu Project.
10. Pour créer une classe, cliquer le bouton New Class et spécifier un nom.
11. Pour créer une flèche, cliquer le bouton correspondant et faire glisser la flèche sur le diagramme, ou tout simplement modifier le code dans l'éditeur. Le diagramme de classe montre les dépendances entre les classes sous la forme de flèches.
12. Pour supprimer une classe, choisir la fonction Remove du menu contextuel. Pour supprimer une flèche, choisir Remove du menu Edit et cliquer sur la flèche.

Mise au point

13. Pour positionner un point d'arrêt, cliquer sur la zone des points d'arrêt à gauche du texte dans l'éditeur (voir figure 13).
14. Pour exécuter un programme ligne par ligne, utiliser les boutons Step et Step Into de l'outil de mise au point.
15. L'accès au contenu des variables est extrèmement simple – il est affiché automatiquement par l'outil de mise au point.
16. Les boutons Halt et Terminate peuvent être utilisés pour arrêter l'exécution temporairement ou définitivement.

Création d'applications autonomes

17. Pour créer une application autonome, utiliser Project > Export...

Création d'appliquettes

18. Pour exécuter une appliquette, choisir Run Applet dans le menu contextuel de l'appliquette.
19. Pour créer une appliquette, cliquer sur le bouton New Class et choisir Applet comme type de classe.

Autres opérations

20. Les paquetages extérieurs à BlueJ peuvent être ouverts avec la commande Project > Open Non Bluej
21. Des classes venant de l'extérieur peuvent être copiées dans le projet en utilisant la commande Add Class from File.
22. Les méthodes statiques peuvent être appelées depuis le menu contextuel de la classe.
23. Pour engendrer la documentation d'un projet, sélectionner Project Documentation depuis le menu Tools.
24. Les classes standard de l'API Java peuvent être visualisées en sélectionnant Help > Java Class Libraries
25. Pour créer des objets à partir de classes d'une bibliothèque, sélectionner Tools > Use Library Class