TP de Programmation Orientée Objet
ILOG1 & Licence 3ème année*

Semaine du 28 novembre 2005

Ce sujet de TP reprend très largement le premier chapitre du livre “Conception objet en Java avec BlueJ” de David Barnes et Michael Kölling (Editeur Pearson Education).

1 Création d’objets

Faites une copie du répertoire `/usr/local/bluej/examples/shapes` (et de son contenu) dans votre répertoire personnel (par exemple dans votre répertoire `Documents`) à l’aide la commande :
`cp -R /usr/local/bluej/examples/shapes Documents`
puis placez vous dans le répertoire nouvellement créé et nommé `shapes` à l’aide de la commande `cd Documents/shapes`.

Lancez BlueJ à l’aide de la commande :
`/usr/local/bluej/bluej`
et ouvrez l’exemple shapes que vous venez de récupérer. La fenêtre de la figure 1 apparait.

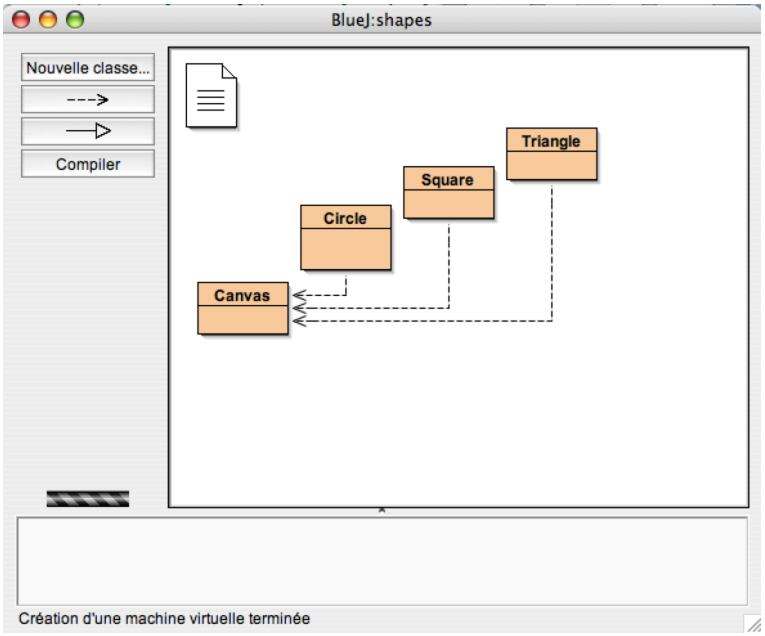

FIG. 1 – Le projet shapes dans BlueJ (sous Mac OS X)

La fenêtre contient un diagramme. Chacun des rectangles colorés représente une classe du projet : `Circle`, `Square`, `Triangle` et `Canvas`.

Cliquez du bouton droit sur la classe `Circle`. Dans le menu déroulant qui s’affiche, choisissez :

`new Circle()`

Pour attribuer à l’instance le nom qui s’inscrit par défaut, cliquez su OK. Un reactangle rouge apparait au bas de l’écran, il est intitulé “circle_1” (cf. figure 2).

*M. Champesme – Département d’Informatique – Institut Galilée

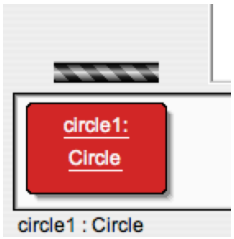

FIG. 2 – Un objet sur le bureau des objets.

Vous venez de créer votre premier objet. L'icône rectangulaire de la figure 1, intitulée "Circle", représente la classe `Circle` ; `circle_1` représente un objet créé à partir de cette classe. La zone située au bas de l'écran et contenant l'objet s'intitule le *bureau des objets* (ou "object bench" en anglais).

Créez un autre cercle. Puis créez un carré.

2 Appel des méthodes

Cliquez du bouton droit sur l'un des objets cercle (et non sur la classe !). Un menu contextuel s'affiche, contenant diverses options. Choisissez `makeVisible` pour dessiner une représentation de ce cercle dans une nouvelle fenêtre (cf. figure 3).

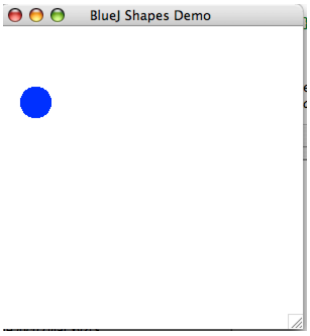

FIG. 3 – Le dessin d'un cercle.

Le menu du cercle contient plusieurs autres opérations. Vous pouvez tester `moveRight` et `moveDown` à plusieurs reprises pour rapprocher le cercle du centre de la fenêtre. Essayez également `makeInvisible` et `makeVisible` pour masquer et afficher le cercle.

Les choix du menu du cercle permettent de manipuler celui-ci. En langage Java, on les désigne sous le nom de *méthodes*. On dit communément que l'on *appelle* ou que l'on *invoque* ces méthodes.

3 Paramètres

Invoquez maintenant la méthode `moveHorizontal`. Une boîte de dialogue vous demande de saisir certaines informations (cf. figure 4). Tapez 50 et cliquez sur OK. Le cercle se déplacera de 50 pixels sur la droite.

Invoquez les méthodes `moveVertical`, `slowMoveVertical` et `changeSize`. Essayez de déplacer le cercle de 70 pixels sur la gauche en utilisant `moveHorizontal`.

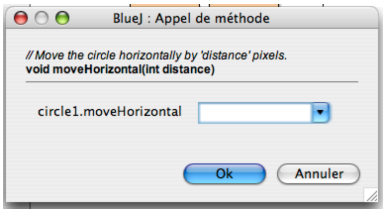

FIG. 4 – Boîte de dialogue pour l'appel d'une méthode avec un paramètre.

4 Instances multiples

Créez plusieurs objets cercle sur votre bureau des objets. Pour cela, sélectionnez `new Circle()` dans le menu déroulant de la classe `Circle`. Affichez les, puis déplacez les à l'écran à l'aide des méthodes `move`. Augmentez la taille de l'un d'entre eux, que vous transformerez en jaune, réduisez en un autre, que vous rendrez vert. Essayez également les autres formes : créez quelques triangles et quelques carrés. Modifiez leur positions leurs tailles et leurs couleurs.

Une fois la classe créée, vous pouvez construire autant d'objets (ou d'instances) de cette classe que vous le souhaitez. A partir de la classe `Circle` vous pouvez créez de nombreux cercles. Idem avec les carrés à partir de la classe `Square`.

Chacun de ces objets possède sa propre position, sa couleur, sa taille. Pour modifier l'attribut d'un objet (comme sa taille), appelez une méthode sur cet objet. Cela affectera l'objet concerné et non les autres.

5 Etat

L'ensemble des valeurs de tous les attributs qui définissent un objet (comme la position x, la position y, la couleur, le diamètre et l'état de visibilité d'un cercle) est également appelé l'*état* d'un objet.

Dans BlueJ, pour connaître l'état d'un objet, sélectionnez la fonction "Inspecter" le menu déroulant de cet objet. Lorsqu'on inspecte un objet, une fenêtre semblable à celle présentée à la figure 5 s'affiche. C'est ce que l'on appelle l'*inspecteur d'objet*.

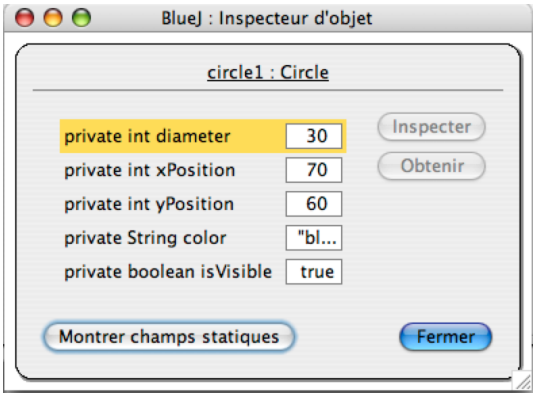

FIG. 5 – L'inspecteur d'objets.

Assurez-vous que le bureau des objets contient plusieurs objets, puis étudiez les tour à tour. Essayez de modifier l'état de l'un d'entre eux (par exemple en appelant la méthode `moveLeft`) lorsque l'inspecteur d'objets est ouvert. Remarquez que les valeurs changent dans l'inspecteur.

6 Définition d'un objet

En étudiant les différents objets, on remarque que les objets d'une même classe disposent tous des mêmes champs. En fait le nombre, le type et les noms des champs sont identiques, mais la valeur effective d'un champ

donné peut différer pour chaque objet. Au contraire, les objets de classes différentes peuvent avoir des champs différents. Un cercle, par exemple, dispose d'un champ "diamètre" (i.e. `diameter`) tandis qu'un triangle possède des champs de largeur (`width`) et de hauteur (`height`).

En fait, les types et les noms des champs se définissent dans une classe, et non dans un objet. Ainsi, la classe `Circle` impose que chaque objet cercle (i.e. chaque instance de la classe `Circle`) dispose de cinq champs intitulés `diameter`, `xPosition`, `yPosition`, `color` et `isVisible`. Elle sous-entend également les types de ces champs. En effet, les trois premiers sont du type `int`, tandis que la couleur est du type `String` et que la balise `isVisible` est du type `boolean`.

Lorsqu'on crée un objet d'une classe Circle, il contient automatiquement ces champs. Leurs valeurs sont alors stockées dans l'objet. Ainsi, chaque cercle disposera d'une couleur différente pour chacun (cf. figure 6).

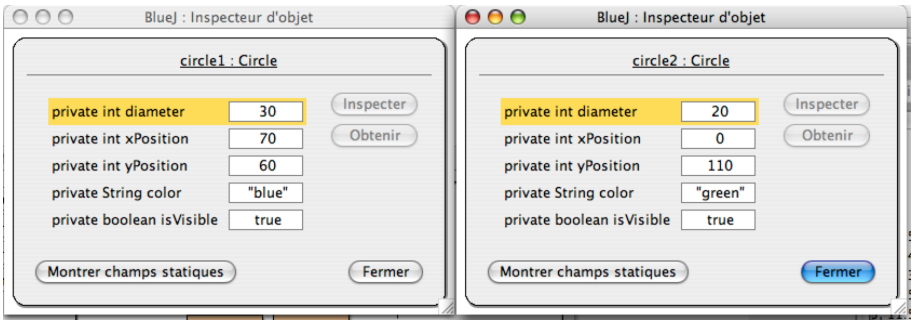

FIG. 6 – Champs de deux instances d'une même classe (la classe `Circle`).

La situation est identique pour les méthodes. Elles sont définies dans la classe de l'objet. En conséquence, tous les objets d'une classe donnée disposent des mêmes méthodes. Toutefois, les méthodes sont invoquées sur les objets. Cela permet de connaître avec certitude l'objet modifié par un appel à la méthode `moveRight`.

Utilisez les formes du projet shapes pour dessiner une maison et un soleil, à la manière de ceux qui sont représentés à la figure 7. Ecrivez en même temps ce que vous devez faire pour y parvenir.

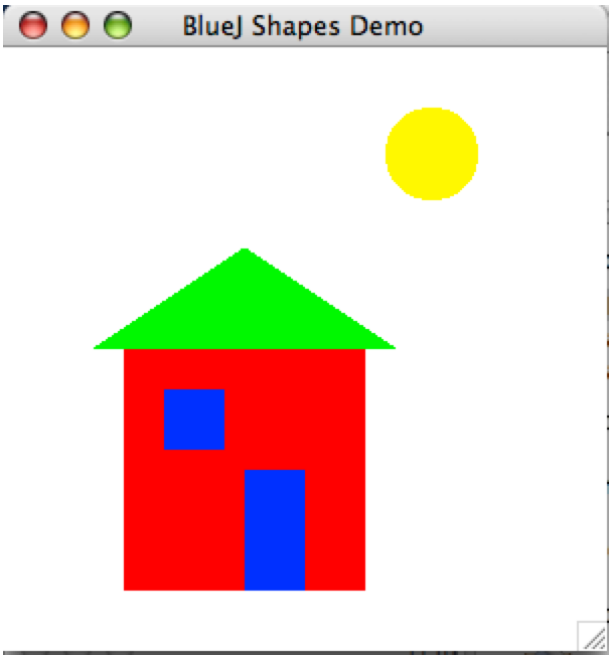

FIG. 7 – Une image créée à partir d'un ensembles d'objets formes.