IN101 - TP1 1112v1
Sujet

Durée : 1 h

1 OBJECTIFS

- Savoir utiliser les fonctionnalités élémentaires de l'environnement de développement BlueJ ; savoir éditer et compiler une classe, et exécuter une méthode.
- **Savoir lire un énoncé et respecter les consignes.**
- Découvrir les concepts de classe, objet, attribut, méthode, constructeur.
- Comprendre le concept de référence (flèche) et le principe de l'interaction entre objets .
- *Si vous n'êtes pas sûr de la réponse à une question posée dans ce sujet, appelez un intervenant.*
- Savoir utiliser et modifier des méthodes.
- Savoir définir et accéder à des champs de type simple.
- Comprendre l'intérêt des commentaires de documentation (javadoc).
- ***Afficher ce sujet dans un navigateur pour pouvoir bénéficier des liens.***

2 MODIFICATION DE MÉTHODES

Attention ! Un exercice peut perdre tout son intérêt pédagogique si les consignes pour cet exercice ou pour les précédents n'ont pas été scrupuleusement respectées, étape par étape.

2.1 Prise en mains de BlueJ

BlueJ est un environnement de développement de programmes Java destiné à l'enseignement initial de la programmation. Il a été conçu et développé par l'équipe BlueJ de l'université Monash de Melbourne, en Australie, et les universités de Southern Denmark, Kent, et Deakin.

Pour plus d'informations se reporter à la page web de l'unité.

Le logiciel BlueJ et sa documentation sont en libre utilisation pour des usages non commerciaux.

Une version française du tutoriel est également accessible.

BlueJ tourne sous Windows, Linux, et sur toute plate-forme fournissant une machine virtuelle Java. Le JDK doit être installé (pas seulement le JRE).

2.1.1 Manipulations préalables

1. Créez sur votre compte un répertoire `In101` qui contiendra jusqu'à la fin de l'année scolaire tous les programmes réalisés dans cette unité.
2. Créez dans ce répertoire, un sous-répertoire `TP1` qui contiendra tous les exercices réalisés pendant ce premier tp.
3. Sauvegardez (bouton droit/Save as) le fichier `formes5.jar` dans ce sous-répertoire.
   Un fichier `.jar` est l'équivalent java d'un fichier `.zip`, c'est-à-dire une archive compressée contenant plusieurs fichiers (ici, des classes java).
4. Essayez dans la suite de toujours éviter de dupliquer des instructions inutilement.

2.1.2 Charger un projet dans BlueJ

1. Démarrer BlueJ soit par la commande `bluej3 &` soit par le menu « Démarrer ». (ESIEE/Apps)
2. Grâce au menu Projet/Ouvrir non-BlueJ…, chargez le projet `formes` que vous venez de sauvegarder.
3. Un diagramme s'affiche dans la fenêtre principale de BlueJ. Il se compose de 4 rectangles : `Canvas`, `Cercle`, `Carre`, `Triangle`. Chacun de ces rectangles représente une classe. Cette ensemble de classes, avec les liens qui les unissent, constitue le projet `formes`.

2.1.3 Créer un objet

1. Pour créer un objet d'une classe donnée, il faut que cette classe soit activable, c'est-à-dire compilée. Sous BlueJ, les classes non compilées apparaissent hachurées. Si donc la classe considérée apparaît hachurée, commencer par la compiler : clic droit sur cette classe puis choisir la fonction Compiler.
2. Compiler la classe `Cercle` : celle-ci n'est plus hachurée. Mais elle n'est pas la seule ! La classe `Canvas` a elle aussi été compilée bien qu'on ne l'ait pas demandé : c'est parce-que la classe `Cercle` utilise certains services offerts par la classe `Canvas` (pour l'affichage graphique) que BlueJ a automatiquement déterminé qu'il fallait aussi compiler la classe `Canvas` ; d'autre part, le fait que la classe `Cercle` utilise la classe `Canvas` est matérialisé par la flèche en pointillé qui relie ces 2 classes.
3. Maintenant, après un clic droit sur la classe `Cercle`, choisir `new Cercle()`. Le système propose un nom d'instance par défaut : `cercle1`. Le valider par OK. Un rectangle rouge apparaît dans le présentoir à objets : il représente l'objet `cercle1` qui vient d'être créé. `cercle1` est une instance de la classe `Cercle`. Le cercle n'apparaît pas encore sur le dessin.
4. *Exercice 2.1.3a : Créer un **carré** (la classe est fournie), puis un deuxième cercle.*

2.1.4 Inspecter un objet

Pour visualiser l'état d'un objet, activer le clic droit sur cet objet et choisir la fonction Inspecter *(si une valeur apparaît comme une flèche, elle indique une référence vers un autre objet ; la sélectionner puis cliquer sur Inspecter pour en voir le contenu)*. L'inspecteur d'objets ne permet aucune modification.
Pourquoi des guillemets autour de `blue` et pas de `false` ?

2.1.5 Exécuter une méthode

1. Après un clic droit sur l'objet `cercle1`, choisir la méthode `rendVisible`. Le cercle (ou plutôt le disque) s'affiche dans une fenêtre séparée (regardez éventuellement dans la barre de tâches). **A partir de maintenant, les 2 fenêtres doivent être visibles simultanément.**
2. Faire de même avec les 2 autres objets créés au **2.1.3.4**. ***Pourquoi ne voit-on pas les 3 formes dessinées ?*** *(`vaBas()` sur `cercle1` peut vous aider à comprendre)*
3. *Exercice 2.1.5a : Invoquer (appeler) les méthodes `vaDroite` et `vaGauche` un certain nombre de fois pour déplacer* `cercle1` *vers le centre de la fenêtre graphique (ces méthodes n'ayant aucun paramètre, on ne peut pas choisir la longueur du déplacement souhaité).*
4. Invoquer la méthode `depHorizontal`, spécifier comme paramètre la valeur `50`, et valider par OK. Le cercle se déplace de 50 pixels vers la droite.
5. *Exercice 2.1.5b : Utiliser la méthode `depHorizontal` pour déplacer le cercle de 70 pixels* **vers la gauche**.
6. La méthode `depHorizontal` a un paramètre de type `int`, c'est-à-dire de type nombre entier relatif. D'autres méthodes peuvent avoir des paramètres d'autres types. Par exemple, la méthode `changeCouleur` a un paramètre de type `String`, c'est-à-dire de type chaîne de caractères (c-à-d mot ou phrase).

7. *Exercice 2.1.5c : Invoquer la méthode* `changeCouleur` *avec le paramère* `"red"` *(guillemets compris!!). Que se passe-t-il ? Que se passe-t-il si le paramètre est fourni sans ses guillemets ? Que se passe-t-il si on spécifie une couleur inconnue, par exemple* `"rouge"` *?*

2.1.6 Explorer

1. Lors du clic droit sur un objet, beaucoup de méthodes peuvent être disponibles. Pour savoir exactement ce que fait telle ou telle méthode, ou bien quel est le rôle de tel ou tel paramètre, il est possible de consulter une documentation extraite automatiquement du programme `.java` (si les bons commentaires y figurent, (voir cours 1.2 au I.1). On appelle cette documentation l'interface de la classe, et le programme `.java` l'implémentation.
2. *Exercice 2.1.6a : Double-cliquer sur la classe* **`Cercle`** *et utiliser la liste déroulante en haut tout à fait à droite (**texte rouge**) pour choisir **Documentation** plutôt que **Source Code**. Lire la documentation et essayer de retrouver les correspondances avec le texte du programme* `.java`*. Poser des questions aux intervenants sur ce que vous ne comprenez pas.*
3. *Exercice 2.1.6b : Essayer toutes les méthodes fournies par les classes* `Cercle`*,* `Carre`*, et* `Triangle` *(sauf les méthodes « héritées de … »). **Attention !** Le fait de supprimer un objet n'efface pas son dessin dans l'image, il faut le rendre invisible AVANT de le supprimer. Si vous hésitez avant de saisir des paramètres, lisez le commentaire en italique au-dessus du champ de saisie.*
   La méthode `efface()` *n'est pas publique pour rester cohérent avec la méthode* `dessine()`*.*

2.1.7 Editer une classe

Pour consulter ou modifier le code source d'une classe, activer le clic droit sur cette classe (**évitez la class `Canvas` pour le moment**) et choisir la fonction Éditer (ou bien double-cliquer sur la classe).

Dès que le code source d'une classe est modifié :

- l'icône de cette classe dans la fenêtre principale BlueJ apparaît hachurée
- si une autre classe utilisait la classe modifiée, elle apparaît aussi hachurée.
- toutes les instances de cette classe sont supprimées du présentoir à objets dès la compilation

Parcourir cette classe en observant notamment « l'indentation » (décalages vers la droite pour faire apparaître la structure du programme), les commentaires sur les accolades fermantes (voir cours 1.2 au VI.1), et les mots `private` et `public` pour indiquer ce qui accessible ou pas hors de la classe. On voit donc des méthodes qui n'apparaissaient pas lorsqu'on cliquait sur le bouton droit. ***Notamment, pourquoi les méthodes* `efface()` *et* `dessine()` *ne sont-elles pas publiques ?***

2.1.8 Modifier une classe

1. Une classe dont l'icône apparaît hachurée dans la fenêtre principale BlueJ n'est pas ou n'est plus opérationnelle. Pour le (re)devenir, il est nécessaire de (re)compiler cette classe. Pour compiler une classe, activer le clic droit sur cette classe et choisir la fonction Compiler :
   - Tous les objets disparaissent du présentoir, ainsi que l'éventuelle fenêtre graphique.
   - Si la compilation s'opère sans erreur, l'apparence hachurée de la classe disparaît.
   - Si une erreur est détectée, l'icône de classe reste hachurée et l'éditeur s'ouvre automatiquement et indique l'emplacement et la nature de l'erreur. **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.**
2. *Exercice 2.1.8a : Editer le code source de la classe* `Cercle` *et y changer la couleur par défaut de* `"blue"` *en* `"red"`*, en suivant les étapes suivantes :*
   - *utiliser l'outil de recherche de l'éditeur [bouton* Chercher*...] pour trouver le mot* `blue`
   - *dans l'instruction* `aCouleur="blue";` *, remplacer* `"blue"` *par* `"red"` *;*
   - *sauvegarder le code source ainsi modifié [menu* Classe*, choix* Enregistrer*] ;*
   - *fermer l'éditeur [bouton Fermer].*

3. On peut remarquer que, la classe `Cercle` ayant été modifiée, son icône dans le diagramme des classes apparaît maintenant hachurée. Recompilez cette classe et vérifiez son bon comportement.

4. *Exercice 2.1.8b : Editer le code source de la classe `Cercle` et, afin de simuler une faute de syntaxe, supprimer un point-virgule quelque part. Recompiler et analyser le message d'erreur. Corriger et recompiler. Faire de même en enlevant une lettre à `aDiametre` par exemple.*
   On peut remarquer dans le répertoire `In101/Tp1` que chaque fichier source `.java` qui a été compilé a donné lieu à un fichier binaire `.class` (**le vérifier**).

5. *Exercice 2.1.8c : Editer le code source de la classe `Cercle` pour ajouter une nouvelle procédure `doubleTaille()` en vous inspirant de `changeTaille()`, qui double le diamètre du cercle.* ***Un paramètre est-il toujours utile ?***
   Compiler. Tester.

3 LA JAVADOC

Nota : le travail demandé doit être terminé, en séance ou, à défaut, hors séance.

3.1 Suite du projet "formes"

3.1.1 Explorer la documentation

1. Editer le code source de la classe **Cercle**. Dans l'éditeur, passer du mode **Source Code** (menu rouge en haut à droite) au mode **Documentation**. *Quelles sont les informations du code source qu'on retrouve (**ou pas**) dans cette documentation automatiquement générée ?*
   Dans quel ordre apparaissent les méthodes ? et les attributs ?
   Aide : `java.lang.String` est équivalent à `String` (cette notation sera expliquée plus tard).

2. Si vous avez parcouru la méthode `dessine()` de la classe Carre par exemple, vous avez peut-être vu l'appel à Rectangle() ; mais à quoi servent donc les 4 paramètres ? La réponse est dans la documentation, mais il faut savoir dans quel paquetage (package) chercher ; ici, on voit au début du fichier que seul le paquetage java.awt est importé. Soit vous cherchez java.awt.Rectangle directement sur le site officiel (http://java.sun.com/javase/6/docs/api/), soit vous utilisez BlueJ, [menu Aide, choix Bibliothèques de classes java (en ligne), cliquer sur Java SE API] ou [menu Outils, choix Utiliser une classe de bibliothèque…] + taper java.awt.Rectangle dans la boîte puis la touche ENTRÉE. Ensuite, cliquer soit sur le [bouton Documentation en ligne], soit sur un des constructeurs pour l'essayer !
   NE PAS UTILISER *google* car on tombe presqu'à chaque fois sur une ancienne version de java.

3.1.2 Sauvegarder

1. Sauvegarder le projet `formes` [menu Projet, choix Enregistrer], le fermer [Projet / Fermer].
2. Regarder le répertoire `In101/Tp1` et y faire éventuellement « le ménage » .

4 Fin du TP

- Avez-vous bien fait TOUS les exercices, dans TOUTES leurs étapes ?
- Avez-vous compris TOUT ce que vous avez vu ? Sinon, demandez à un intervenant.
- Avez-vous visité les liens du menu Help ? Sinon, essayez !
- **Avez-vous expérimenté le « CodePad »** [menu View, choix Show Code Pad] ? Sinon, essayez !
- N'avez-vous pas d'idées d'autres modifications de méthodes que celles proposées ? Expérimentez-les.
- Lisez les compléments de cours (1.0 à 1.2) présents sur la page web de l'unité.
- Si vous avez fini avant l'heure, demandez aux intervenants si vous pouvez partir avant la fin du TP ou bien s'ils vous donnent un autre exercice à faire.
- **Sinon, terminez tout en travail personnel avant le prochain tp.**

Ce sujet a été élaboré par Denis Bureau, d'après un sujet d'Albin Morelle.