IN101 – TP6 2011/2012 v3

Sujet (à n'imprimer qu'une seule fois par poste de travail)

Durée : 3 h

1 OBJECTIFS

- Maîtriser les classes et méthodes abstraites
- Maîtriser les interfaces
- Maîtriser les exceptions
- Savoir se passer de BlueJ

2 TRAVAIL A REALISER

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

2.1 Créer un répertoire de travail

Si BlueJ n'est pas ouvert, le lancer. **Visualiser le sujet dans un navigateur** pour bénéficier des liens **ET en pdf** pour éviter les problèmes d'affichage de certains caractères ou pour l'imprimer.
Créer un répertoire `tp6` dans `In101` sur votre compte. *C'est dans ce répertoire que devront être stockés tous les programmes Java et exercices relevant de ce tp.*

2.2 Exercice 1 : Projet "proj_formes_pkg" (classe/méthode abstraite, interface)

Cet exercice va consister à transformer le projet `proj_formes` du tp4.

2.2.1 Ouvrir le projet

Télécharger le fichier `proj_formes_pkg.jar` lié à cet énoncé, et l'enregistrer dans le répertoire `tp6` précédemment créé.

Lancer *BlueJ* et ouvrir, le fichier `.jar` sauvegardé ci-dessus. [*Open non-BlueJ* ...].

2.2.2 Découvrir et essayer le projet `proj_formes_pkg`

Ce projet correspond à peu près à ce à quoi vous deviez aboutir à la fin du TP4.

2.2.3 Restructurer ce projet

1. Comme cela a été expliqué en cours, il n'est pas souhaitable d'autoriser l'instanciation de la classe `Forme`. **Faites** ce qu'il faut pour cela et **vérifiez** que cela n'a rien changé ni à la compilation ni à l'exécution (c'est normal, il n'y avait pas d'instanciation de `Forme`).
2. **Ajoutez** une sous-classe `Hexagone` (recopiée sur `Carre` par exemple) mais en supprimant (commentant) `dessineSpec()` car vous ne savez pas comment dessiner un hexagone pour le moment. Ajoutez un `Hexagone` dans `Maison` et rendez-le visible.
   Compilez. **Créez** une `Maison` : le message est cohérent, mais est-il souhaitable ?
3. Comme cela a été expliqué en cours, il n'est pas souhaitable d'une part que les instructions de certaines méthodes soient inutiles, et d'autre part qu'une sous-classe puisse être ajoutée sans qu'elle possède les caractéristiques minimales de ses « sœurs » telles que les méthodes `dessineSpec()` et `changeTailleSpec()`. Faites ce qu'il faut à ces méthodes pour **corriger** cela dans `Forme` et recompilez `Hexagone`; il devrait y avoir une erreur.

4. Pour résoudre ce problème de compilation, il faut maintenant savoir dessiner un hexagone. En s'inspirant de l'attribut de `Carre`, de `dessineSpec` dans `Triangle`, de la javadoc du constructeur de `Polygon`, et de l'aide ci-dessous, écrivez la méthode `dessineSpec` dans `Hexagone`.
Aide : Une façon de dessiner un hexagone est d'utiliser un `Polygon` avec les 6 sommets suivants : (x, y+s), (x+g, y+p), (x+g, y-p), (x, y-s), (x-g, y-p), et (x-g, y+p), avec (x, y) les coordonnées du centre, `s` la taille du côté, p = ½s, et g = ½s√3 *(voir dessin en bas de page)*.
Contraintes : déclarez 2 variables *p* et *g*, puis créez une fonction `public int calcG()` qui calcule *g* en fonction de la taille du côté. La démarche est exposée en détail en bas de la page suivante *(mais à regarder uniquement si vous n'y arrivez pas).*
Vérifiez que l'hexagone dessiné est « régulier » et non pas « allongé ». *(√ = racine carrée)*
Compilez la classe `HexagoneTest` si elle ne l'est pas. [Run Tests]. Tout est vert ?

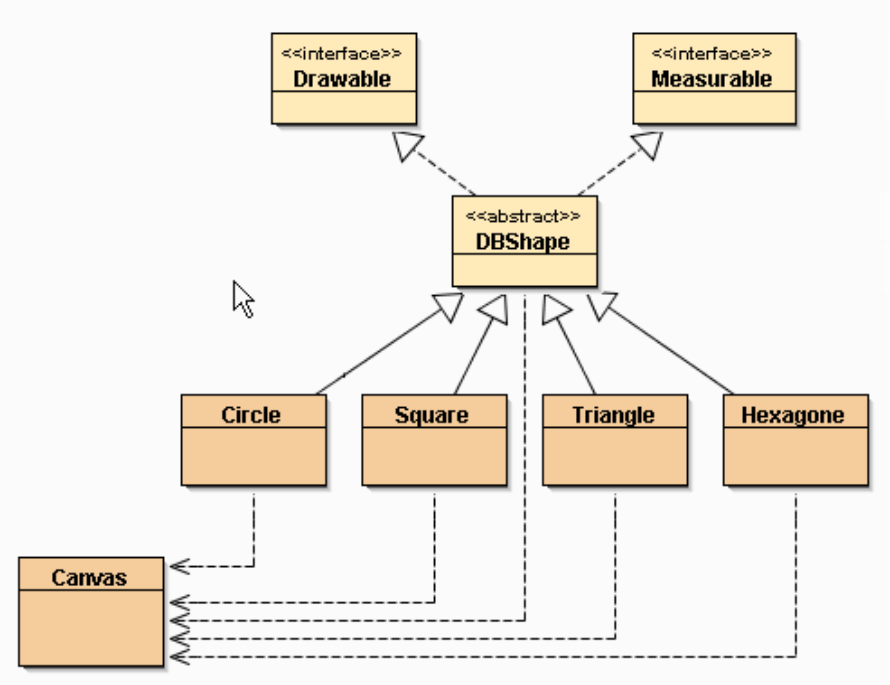
5. On veut maintenant pouvoir « hériter » de plusieurs caractéristiques telles que `Dessinable`, `Mesurable`, … L'héritage multiple étant interdit en java, nous allons donc implanter des interfaces. Créez l'interface `Dessinable` ne comportant que `dessine()`, *(pourquoi pas `dessineSpec()` ?)*, `efface()`, `estVisible()`, `rendVisible()`, et `rendInvisible()`. `Forme` doit maintenant respecter cette interface. Compilez.

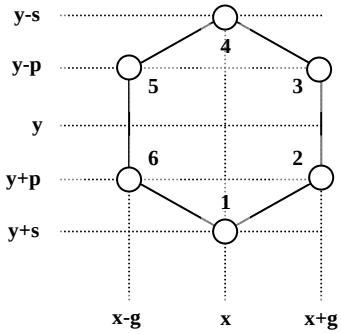
6. Créez une nouvelle interface `Mesurable` imposant les 2 méthodes retournant un réel `surface()` et `perimetre()`. `Forme` doit maintenant respecter aussi cette interface, mais peut-elle implémenter `surface()`et `perimetre()` ? Ajouter ce qu'il faut dans les sous-classes. **Aide :** La surface d'un hexagone vaut 1.5(√3)s². Tous nos triangles sont forcément isocèles, donc `Math.hypot()` peut certainement vous servir *(voir dessin en bas de page).*

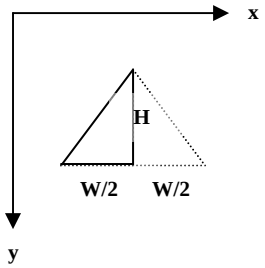
7. Ajoutez dans `Maison` *avant le deuxième* `System.out.println` (respectivement **après le dernier** `System.out.println`) la création d'un tableau de `Mesurable` contenant *les 6* (respectivement **les 3**) formes existant à ce moment du programme.

8. Ajoutez juste après chacun des 2 affichages ci-dessus un appel à la procédure (qu'il vous faudra écrire) `printStats()` qui devra calculer et afficher la somme des surfaces puis la somme des périmètres des formes du tableau qu'on lui passe en paramètre.
Testez ; ça devrait donner environ 27139.08 et 1454.43, puis 6531.36 et 513.36.
Au fait, le nombre d'instances de Forme est-il correct, tout à la fin ?

9. A la fin de l'exercice, on devrait se retrouver dans cette situation :







2.2.4 Savoir développer sans BlueJ

1. Fermez *BlueJ* et ouvrez un terminal dans le répertoire `proj_formes_pkg`.
2. Utilisez un éditeur de texte (*nedit, emacs, KWrite,* …) pour modifier le fichier `pkg_application/Maison.java`.
3. Ajoutez l'affichage d'une ligne d' `*` au début et à la fin de `printStats()`.
   ***Rappel** : vous êtes en train d'écrire des instructions en Java …*
4. Sauvegardez, et quittez l'éditeur.
5. Compilez par : `javac pkg_application/Maison.java`
   (on indique ici le chemin d'accès complet au fichier `.java`)
6. Exécutez par : `java pkg_application.Maison`
   (on indique ici le nom complet de la classe, avec son paquetage)
   une erreur se produit ? Corrigez-la en créant juste un objet `Maison` là où il faut.
   (attention à la signature O-BLI-GA-TOIRE de la méthode à créer – **voir résumé du cours 5**)
7. Recompilez, retestez : ça marche mais c'est bloqué ? `System.exit(0);` permet de terminer le programme (on peut remplacer le 0 par un code d'erreur destiné au système).
8. Recompilez, retestez : ça marche mais on n'a plus le temps de voir ? **On attendra le 2.3.1 pour corriger.**
9. Relancez l'éditeur (en n'oubliant pas le & à la fin de la ligne pour ne pas avoir à quitter l'éditeur de texte) pour modifier les 4 sous-classes de `Forme`. Changez la couleur par défaut du cercle en vert, du carré en jaune, du triangle en bleu, et de l'hexagone en rouge.
10. Sauvegardez chaque fichier sans sortir de l'éditeur.
11. Compilez par : `javac pkg_graphique/*.java`
12. Exécutez par : `java pkg_application.Maison` (ou déboguez avec `jdb`)

S'il y a un problème de **version de java ou de format de classe**, c'est que la commande `java` lance une JVM trop ancienne. Dans ce cas :
1) BlueJ/Help/About BlueJ : noter le chemin du JDK puis fermer.
2) Taper : `alias java6 chemin_du_JDK/bin/java`
3) Remplacer `java` par `java6` dans les commandes ci-dessus.

S'il y a un problème pour trouver **`javac` sous windows/dos**, c'est que le chemin du JDK ne se trouve pas dans le *command path*. Dans ce cas :
1) BlueJ/Help/About BlueJ : noter le chemin du JDK puis fermer.
2) Taper : `set path=chemin_du_JDK/bin/;%path%`
3) Il peut être pratique de mémoriser cette commande dans un fichier USEJAVA.BAT

Démarche pour le 2.2.3.4 :

- Après avoir déclaré/initialisé 2 variables `vX` et `vY` pour éviter d'avoir à appeler plusieurs fois les accesseurs, déclarez/initialisez 2 variables entières `vP` et `vG` pour éviter d'avoir à les calculer plusieurs fois, mais attention à effectuer chaque calcul en réel (`double`) avant d'arrondir le résultat à un entier.
- Écrivez la méthode `calcG` puis testez-là à l'aide de la méthode `testcalcG` d'`HexagoneTest`.
- Écrivez les abscisses des 6 points dans l'ordre du dessin de la page précédente dans `vXPoints` et les 6 ordonnées dans le même ordre dans `vYPoints` .
- Recopiez depuis la classe Triangle l'appel à *Canvas.draw(…,…, new Polygon…).*

T. SVP →

Exercice 2 : Les exceptions

2.2.5 Exemple simple de traitement d'exception « silencieux »

1. Pour pouvoir observer le dessin juste avant la suppression des 3 formes, il faut ajouter une pause grâce à `Thread.sleep( 5000 );` Essayez. Une erreur de compilation ?
2. Consultez la documentation de cette méthode pour comprendre ce qui se passe, et ajoutez le try/catch nécessaire, sans rien faire en cas d'exception *(à éviter en général)*. Recompilez et retestez.

2.2.6 Compréhension des mécanismes

Ajoutez dans un des projets BlueJ du jour une nouvelle classe `FinallyDemo` que vous devez copier/coller à partir de cette page. (lire d'abord cette explication de l'instruction `switch`)
Compilez. Exécutez `main()`. Comparez le code java et les affichages dans le terminal.
Posez des questions aux intervenants si vous ne comprenez pas certains affichages.

2.2.7 Création d'une classe d'expérimentation

1. Créez (en dehors des paquetages) une nouvelle classe `Calculs` sans attributs ni constructeurs.
2. Écrivez une première méthode statique `carre` prenant une `String` en paramètre et devant retourner le carré du nombre représenté par la `String` en paramètre. Pour cela, utilisez la méthode `convStringToInt` qui effectue la conversion `String` vers entier. Incorporez le source (volontairement peu lisible) de cette méthode à la fin de votre classe. Compilez et lisez le point suivant.
3. Le compilateur vous dit **soit** de traiter l'exception « vérifiée » susceptible de se produire quand la `String` ne représente pas un nombre (`java.util.zip.DataFormatException`), **soit** de déclarer que la fonction `carre()` est susceptible de la lancer, ce qui forcera tous les programmes qui utiliseront `carre()` à faire de même. Pour éviter cet inconvénient, nous allons traiter l'exception dans `carre()`, mais le traitement consistera à lancer une exception « non vérifiée » (`IllegalArgumentException`) pour ne rien imposer aux utilisateurs de `carre()`. Il semble utile de récupérer le message d'information (grâce à la fonction `getMessage()`) dans l'exception que l'on a interceptée pour le transmettre au constructeur de l'exception que l'on veut lancer. Compilez. Testez avec `"10"` puis avec `"toto"` .
4. On souhaite maintenant que la fonction retourne -1 quand son paramètre vaut `"i"` . Ajoutez ce cas particulier au début de la fonction. Compilez. Testez avec `"i"`, `"10"` et `"toto"` .

2.2.8 Amélioration de la classe d'expérimentation

1. Écrivez une nouvelle méthode testCarre() sans paramètres qui commencera par déclarer un tableau de String contenant "2", "5", "i", "10", "0", "20".
2. Affichez « 1) », calculez la somme des carrés des nombres contenus dans le tableau, puis affichez le résultat. Compilez. Testez. Cela affiche-t-il bien 1) 528 ?
3. Affichez « 2) », copiez/collez les lignes du point 2 ci-dessus après avoir remplacé le contenu de l'avant-dernière case par `"toto"` . Compilez. Testez. Corrigez le problème en traitant l'exception dans la boucle. Recompilez. Retestez. Cela affiche-t-il bien 1) 528 puis 2) 528 ?

2.3 Terminer la séance

Si pas fait antérieurement, **générer** (après l'avoir complétée !) la documentation de ce dernier projet et de proj_formes_pkg, puis sauvegarder les projets ouverts, puis fermer BlueJ. Si besoin, envoyer par mél à votre binôme, en fichiers attachés, tous les projets de ce tp (exportés sous forme de fichiers `.jar`). Se déloger.