**IN101 – TP4** 2011/2012 v2

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

**Durée : 3 h**

# 1 OBJECTIFS

- Maîtriser l'héritage et la redéfinition
- Maîtriser les méthodes de la classe Object
- Savoir aussi structurer une application avec des paquetages
- **Attention !** Suivre scrupuleusement les instructions données dans ce sujet, et **dans l'ordre**. Lorsque vous voyez le symbole ▼, lisez la phrase suivante avant de traiter la question.

# 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 pas fait antérieurement, fermer tous les projets ouverts dans Bluej ou, si Bluej n'est pas ouvert, le lancer. **Visualiser le sujet dans un navigateur** pour bénéficier des liens **ET en pdf** pour qu'il soit plus agréable à lire.
Créer un répertoire `tp4` 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" (héritage, redéfinition, Object)*

Cet exercice va consister à structurer une application affichant des formes géométriques.

### 2.2.1 Ouvrir le projet

Télécharger le fichier `proj_formes.jar` lié à cet énoncé, et l'enregistrer dans le répertoire `tp4` précédemment créé.

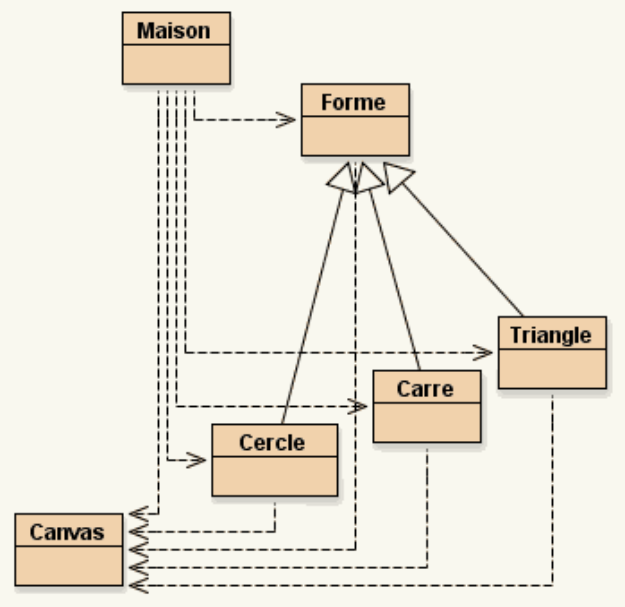
Lancer *BlueJ* et ouvrir, le fichier `.jar` sauvegardé ci-dessus. [ *Open non-BlueJ* ].

### 2.2.2 Découvrir et essayer le projet `proj_formes`

- Ce projet est quasiment identique au projet proj_maison du TP2. Instanciez la classe `Maison` et vérifiez le dessin obtenu par rapport aux instructions de la méthode `dessine()` de cette classe.
- On souhaite modifier les méthodes `vaDroite()`, `vaGauche()`, `vaHaut()`, `vaBas()` de la classe `Cercle` pour qu'elles ne déplacent le cercle que de 10 (au lieu de 20).
  *Est-ce souhaitable d'avoir à effectuer cette modification 4 fois ?* ▼
- Déclarez au début de la classe `Cercle` une constante nommée `PAS` et utilisez-la dans ces 4 méthodes pour que ce genre de modifications soit plus aisé à l'avenir.
- On désire maintenant effectuer la même amélioration dans les classes `Carre` et `Triangle`.
  *Est-ce souhaitable d'avoir à effectuer cette modification 3 fois ?* ▼
  Plus généralement, parcourez le code source des classes `Cercle`, `Carre`, et `Triangle`.
  *Que remarquez-vous ? (le paragraphe suivant apporte une solution)*

### 2.2.3 Restructurer ce projet

Comme vous l'avez compris, beaucoup de points communs existent entre ces 3 classes, et il faut évidemment utiliser l'héritage pour éviter autant de duplication de code. A la fin de l'exercice, on devrait se retrouver dans cette situation :




**Aide-mémoire**

Les constructeurs sans paramètre doivent utiliser les valeurs suivantes.

`Cercle` : 60, 60, yellow, 35
`Carre` : 60, 60, blue, 35
`Triangle` : 50, 50, red, 20, 40

Créez une nouvelle classe `Forme` ; remplacez tout ce qui s'y trouve par un copier/coller de l'intégralité de la classe `Cercle` ; **avant de compiler**, remplacez tous les « `Cercle` » par des « `Forme` ».

### 2.2.4 La classe Forme

Outre les **attributs communs** *(quels sont-ils ?)*, elle comprendra **uniquement** :

- un constructeur pour les initialiser (int, int, String) *(la forme sera invisible au début)*
- un constructeur sans paramètres (0, 0, "black")
- les méthodes communes (donc non **spéc**ifiques)
- Compilez `Forme` et voir le point suivant qui résout le problème pour trouver ……`Spec`. ▼ (**aide :** Spec veut dire spécifique)
- 2 méthodes ……`Spec` « bidon » (puisqu'on ne sait pas quelles instructions écrire tant qu'on ne connaît pas précisément la forme). Elles ne sont pas censées être appelées, mais seront redéfinies au 2.2.5. Il vaut mieux y mettre un message du style `System.err.println( "Erreur du programmeur !" );` pour être prévenu si ça se produisait quand-même … *Cette façon de faire peu séduisante sera améliorée dans un prochain TP.*
- **Tester** : FormeTest / bouton droit / Test All . *Tout est vert ?*

### 2.2.5 Les 3 sous-classes

Elles comprendront uniquement :

- les attributs spécifiques, le constructeur avec paramètres (à modifier), et le constructeur sans paramètre (ne pas changer **son comportement** !)
- les 2 méthodes ……`Spec` qu'on peut désormais écrire ou plutôt **redéfinir**. *(comme elles étaient avant)* Soit ça compile mais on a « Erreur du programmeur » *(la redéfinition est-elle bien signalée ?)* Soit ça ne compile pas car on ne peut redéfinir une méthode privée *(corrigez, mais pas `public`)* Si ça ne compile toujours pas (droits d'accès trop faibles), corrigez, mais pas `public` .

### 2.2.6 Tester les modifications

**Tester** « la bonne programmation » en cliquant sur « Run tests ». *Tout est vert ?*
**Tester** le bon fonctionnement de cette « application ». *Tout fonctionne comme avant ?*

### 2.2.7 La comparaison de formes

- Redéfinissez uniquement dans `Forme` la méthode `equals` en traitant tous les cas vus au TD4. On considérera que la visibilité n'intervient pas dans le résultat de cette méthode. N'oubliez pas de signaler la redéfinition. Compilez.
- Ajoutez à la fin de la méthode `dessine()` de `Maison` un affichage du résultat de la comparaison entre `aSoleil` et `aSoleil2` *(ces 2 cercles ont des caractéristiques différentes …)*
- Supprimez le changement de couleur de `aSoleil2` et déplacez-le horizontalement de 180 au lieu de 160. *La comparaison est-elle maintenant vraie ?*
- Modifiez la taille de `aSoleil2` d'un facteur 2.5 au lieu de 1.5. Testez.
  Résultat de la comparaison ? *Vous trouvez ça souhaitable ?* ▼
  Corrigez ce défaut en redéfinissant `equals` dans les sous-classes. Testez.
  *Que se passerait-il si on testait `aDiametre` en premier dans `equals()` de `Cercle` ?*
- **Attention !** Une mauvaise méthode `equals` peut créer des problèmes d'affichage dans `Canvas`.

### 2.2.8 Le comptage de formes et le Garbage Collector

- Ajoutez dans `Forme` de quoi compter le nombre de formes (en + et en - ! *solution page suivante*), tout en affichant un message utilisant `toString()` de `Object` (`Destruction de Cercle@...`) (Ce compteur `aNbFormes` devra rester privé ==> accesseur) **Ne rien tester pour l'instant.**
- La redéfinition de `toString()` dans les 3 sous-classes *(sans duplication de code !)* permettra de montrer les valeurs des attributs … (`Destruction de Cercle@...:10,10,red,30`)
- Ajoutez **au début** du constructeur de `Maison` un affichage du nombre de formes, précédé du message `p1:`. Essayez ; *est-ce bien `p1:0` ?*
- Ajoutez **à la fin** du constructeur de `Maison` un affichage du nombre de formes, précédé du message `p2:`. Essayez ; *est-ce bien `p2:5` ?*
- Créez une deuxième `Maison`. *Est-ce toujours `p1:0` et `p2:5` ?* ▼ Mais heureusement que **non !**
- Après l'affichage du message `p2:`, mettez à `null aMur`, `aToit`, et `aSoleil2` pour que le Garbage Collector puisse détruire ces objets, puis réaffichez le nombre de formes, précédé du message `p3:`. *Est-ce bien `p3:2` ?* ▼
- **Non ?** Mais c'est normal. Le GC ne se déclenche que lorsque la JVM a besoin de mémoire ; donc, pour cet exercice, il faut donc l'appeler explicitement par `System.gc()`. Ajoutez un dernier affichage tout à la fin, précédé du message `p4:`. *Est-ce bien `p4:2` ?* ▼
- **Non ?** Mais c'est normal s'il y a encore des références sur ces 3 objets ; et c'est le cas ! On peut s'en douter en regardant le dessin où aucun objet n'a disparu. On peut s'en convaincre en regardant le code de `Canvas.draw()` qui mémorise effectivement une référence vers l'objet à dessiner.
  Il faut donc faire précéder *(pourquoi pas **suivre ?**)* les affectations à `null` par des `efface()` qui auront le double effet de supprimer toute référence aux objets et de les effacer sur le dessin ! Retestez. Créez une deuxième `Maison`. *Est-ce bien `p4:2` ? puis `p4:4` ?* ▼
- **Non ?** Mais c'est normal. Le GC a seulement noté les objets qui doivent être « finalisés » c'est-à-dire détruits, mais il n'a pas eu le temps de le faire effectivement. Ajoutez l'instruction `pause(100);` pour lui laisser 100ms avant l'instruction suivante. *Est-ce bien `p4:2` puis `p4:4` ?*
- Oui ? Mais on perd 100ms à chaque fois, alors que le GC n'en a besoin que d'une ou deux … Et si vous faites une pause de 2ms tout en lançant un autre programme en parallèle, on ne pourra garantir que le GC aura le temps d'effectuer toutes les finalisations possibles.
  On va donc le lui demander explicitement en remplaçant la pause par `System.runFinalization()`. Retestez. *Ouf !*

### 2.2.9 Les paquetages (relire éventuellement ce résumé de cours)

- Séparez « l'application » du reste des classes « utilitaires ».
- Pour cela, créez un nouveau projet paquetages [menu *Projet*, choix *Nouveau projet*...].
- Ouvrez une fenêtre Terminal sous Linux, allez dans le répertoire de ce tp, et listez le contenu du répertoire `paquetages`.
- Créez un nouveau paquetage `application` [menu *Edition*, choix *Nouveau paquetage*...].
- Double-cliquez sur le nouveau paquetage et ajoutez la classe `Maison` obtenue au 2.2.8 [menu *Edition*, choix *Copier une classe*...].
- Observez la ligne qui a été ajoutée au début de la classe `Maison` et dans la fenêtre Terminal, listez le répertoire `paquetages`, ainsi que le sous-répertoire `application`.
- Recommencez l'opération en créant un nouveau paquetage `graphique` pour les 5 autres classes. Compilez.
- Observez la ligne qui a été ajoutée au début de chaque classe et dans la fenêtre Terminal, listez le répertoire `paquetages`, ainsi que le sous-répertoire `graphique`.
- Retournez dans le paquetage `application`, et compilez. *Que se passe-t-il ?* ▼
  Ajoutez ce qu'il faut pour corriger ce défaut (uniquement les classes indispensables). Compilez. *Un problème d'accès ?* ▼
- Le mode `protected` nous a permis d'appeler `efface()` car nous étions dans le même paquetage, ce qui n'est plus le cas. Passez `efface()` en `public`, recompilez et testez.

## *2.3 Terminer la séance*

Si pas fait antérieurement, sauvegarder les projets ouverts, puis fermer BlueJ [ menu Projet, choix Quitter ]. 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.

Ce sujet a été élaboré par Denis Bureau.

Solution du 2.2.8 :

- incrémenter le compteur dans le constructeur
- décrémenter le compteur dans `finalize()`