M1 MIAGE

Programmation Java Avancée : TP1

BRAIRE Célia

Exercice 1 :

2) Raccourci pour écrire *System.out.println().*

3) Propose plusieurs choix d’écriture automatique dont la méthode *toString().*

4) Propose plusieurs méthodes communes dont le programmeur pourrait avoir besoin pour son code.

5) Propose les fonctions *getFoo*() et *setFoo*(). Si on écrit juste *set*, il nous propose *setFoo*().

6) Selon l’endroit ou on fait la manipulation (ici le nom de la classe ou *foo*) on peut modifier le nom de la classe, l’objet, la fonction, etc. dans tout le programme.

Exercice 2 :

1) Le code fonctionne car les attributs de *Point* sont atteignables au sein même de la classe *Point*.

2) *x* et *y* étant des attributs privées de *Point*, on peut y accéder seulement à l’aide des méthodes *getX*() et *getY*().

3) Pour respecter l’encapsulation des données.

4) Un accesseur est une fonction permettant d’accéder à des attributs d’une classe. Exemple : *getX*().

5) ?

6) Le paramètre *x* de la fonction rentre en conflit avec l’attribut *x* de notre classe. Pour régler le problème on va nommer notre paramètre autrement, *x1*.

7) On crée une variable globale (*ptr*) que l’on incrémentera à chaque création de *Point*.

8) Le compilateur fait en fonction des arguments qu’on lui donne.

Exercice 3 :

1) Le code affiche un booléen. *p3!=p1* car ils n’ont pas les mêmes adresses mémoires.

3) *IndexOf* renvoi l’indice de l’élément que l’on cherche et utilise la fonction *equals*. Il faut faire une fonction *equals* adapté a *Point*.

Exercice 4 :

2) Utiliser une *arrayList* permet de contourner les erreurs de taille et d’éviter les exceptions de ce type lors de la compilation.

5) *null* n’est pas un Point

Exercice 5 :

5) La fonction *translate* ne fonctionne pas comme on le voudrait. Elle translate le point de tous les cercles liés à ce point au lieu de seulement lui-même. En rendant la classe *Point* non mutable le problème serait réglé.

6) Même problème, la classe *Circle* doit être non mutable.

Exercice 6 :

1) On utilise l’héritage. La classe *Ring* hérite de la classe *Circle* car un anneau est juste un cercle auquel on a rajouté un rayon.

4) Cela appellera la méthode *toString*() de la classe *Circle* vu que c’est la classe mère.