Travaux Dirigés de Java n°2

— Etienne.Duris@univ-mlv.fr —

▸ Exercice 1. (Héritage)

Définir une classe `Disque` *contenant trois champs de type* `double` *:* `x` *et* `y` *représentent le centre du disque et* `rayon` *le rayon de ce disque.*

Définir un constructeur pour cette classe acceptant trois paramètres pour initialiser chacun des champs ainsi qu'un constructeur surchargé sans paramètre construisant un disque de centre (0,0) et de rayon 1.

Définir, dans la classe `Disque`*, une méthode* `deplace(double x, double y)` *qui déplace le disque et une méthode* `toString()` *retournant une représentation de ce disque sous la forme d'une chaîne de caractères de type* `String` *(par exemple* `Disque - centre: (0,0) - rayon: 1` *pour un disque créé par le constructeur sans paramètre).*

Créer quelques objets de la classe `Disque` *dans la méthode* `main()` *de la classe et tester ces méthodes sur ces objets.*

Définir une classe `Anneau` *comme héritant de la classe* `Disque`*. On considère pour cela qu'un anneau est défini comme un disque avec un rayon supplémentaire, "interne", qui définit le sous-disque intérieur manquant à un disque pour définir un anneau. Ainsi, le seul champ à définir explicitement dans* `Anneau` *est le champ* `rayonInterne` *de type* `double`*.*

Définir un constructeur pour `Anneau` *acceptant quatre paramètres (les deux coordonnées, le rayon et le rayon interne) ainsi qu'un constructeur surchargé sans paramètre qui crée un anneau ayant les caractéristiques d'un disque par défaut et un rayon interne nul.*

Définir une méthode `largeur()` *dans la classe* `Anneau` *qui retourne la largeur de l'anneau.*

Créer, dans une méthode `main()` *de test de l'une des deux classes, un objet de chaque classe et tester les différentes méthodes.*

▸ Exercice 2. (Redéfinition)

Dans la classe `Disque`*, définir une méthode* `surface()` *donnant la surface du disque.*

Que dire de la méthode `surface()` *héritée de* `Disque` *dans* `Anneau` *? Il en va de même pour la méthode* `toString()`*.*

Redéfinir ces deux méthodes dans la classe `Anneau`*.*

Définir un champ `couleur` *dans la classe* `Disque` *comme un entier représentant un code de couleur.*

Redéfinir un champ `couleur` *dans la classe* `Anneau` *comme étant de type* `String`*.*

Que dire des comportements des membres redéfinis. Quelle différence entre un champ redéfini et une méthode redéfinie ?

Comment accéder au champ représentant le code couleur d'un objet de classe `Anneau` ?

Comment accéder à la méthode `surface()` définie dans la classe `Disque` à partir d'un objet de la classe `Anneau` ?

▸ Exercice 3. (Sous-typage des références)

Créer un objet de la classe `Anneau`. Affecter la référence de cet objet à une variable `d` déclarée de type `Disque`. Quelle est la méthode utilisée lors d'un appel à `d.surface()` ? Quel est le type de `d.couleur` ? Pourquoi ?

Comment savoir, lorsqu'on dispose d'une variable déclarée de type `Disque`, si la référence contenue dans cette variable est en réalité une référence à un anneau ?

▸ Exercice 4. (De l'égalité)

Créer deux objets identiques de la classe `Disque`, `d1` et `d2`. Par exemple :

```
Disque d1 = new Disque(1,1,5);
Disque d2 = new Disque(1,1,5);
```

Tester l'égalité primitive (`==`) de ces deux variables. Pourquoi ce résultat ?

Utiliser la méthode `equals()` de la classe `Object` pour comparer les deux variables. Pourquoi ce résultat ?

Comment faire pour avoir une méthode de comparaison de ces deux variables ?

Que faire pour pouvoir comparer deux variables de la classe `Anneau` ?