
Exercice 1 : Cercle

Le but de cet exercice est d'écrire une classe représentant les cercles.

Écrivez un programme **`TestCercle.java`** dans lequel vous définissez une classe **`Cercle`** ayant comme attributs privés le rayon du cercle (de type **`double`**), et la position de son centre (les deux coordonnées en «x» et «y» de type **`double`**).

Déclarez ensuite des méthodes «set» publiques pour cette classe, par exemple :

```
void setCentre(double x, double y)
void setRayon(double)
```

Il n'est *a priori* pas nécessaire de définir de «getters» pour cette classe. Ajoutez ensuite les méthodes (faisant aussi partie de l'interface d'utilisation) :

- **`double surface()`** qui calcule et retourne la surface du cercle (pi fois le carré du rayon);
- **`boolean estInterieur(double x, double y)`** qui teste si le point de coordonnées (x,y) passé en paramètre fait ou non partie du cercle (frontière comprise : disque fermé). La méthode retournera **`true`** si le test est positif, et **`false`** dans le cas contraire.

Dans le programme principal, instanciez trois objets de la classe **`Cercle`**, affectez des valeurs de votre choix à leur attributs et testez vos méthodes **`surface`** et **`estInterieur`**.

Remarque : la constante pi est donnée par **`Math.PI`**.

Exercice 2 : tour de magie

2.1 Organisation de l'exercice

Dans cet exercice, nous vous demandons d'élaborer un programme orienté objets de manière indépendante pour la première fois. Nous avons donc organisé l'exercice en 2 parties :

1. une première (de 2.2 et 2.3) qui décrit le problème à résoudre et qui, idéalement devrait suffire pour élaborer puis écrire le programme de façon autonome, certainement en plusieurs tentatives;
2. une seconde (partie 2.4), pour aider au cas où la première partie vous semble insuffisante, vu que c'est votre première conception autonome d'un programme. Dans un premier temps, essayez de ne pas la lire et revenez-y si nécessaire.

2.2 Buts du programme

On souhaite ici écrire un programme « simulant » le tour de magie élémentaire suivant :

> *Un magicien demande à un spectateur d'écrire sur un papier son âge et la somme qu'il a en poche (moins de 100 francs suisses). L'assistant doit ensuite le lire (sans rien dire), puis effectuer secrètement le calcul suivant : multiplier l'âge par 2, lui ajouter 5, multiplier le résultat par 50, ajouter la somme en poche, et soustraire le nombre de jours que contient une année, puis finalement donner le résultat à haute voix.*
> *En ajoutant mentalement (rapidement !) 115 au chiffre reçu, le magicien trouve tout de suite l'âge et la somme en poche (qui étaient restés secrets).*

Modéliser ce tour de magie, en définissant au moins les classes (simples) **`Magicien`**, **`Assistant`** et **`Spectateur`**. Il pourrait également être utile de disposer d'une classe **`Papier`**.

L'instance de **`Spectateur`** devra demander son âge à l'utilisateur du programme ainsi que la somme d'argent en poche, et s'assurer qu'une valeur correcte est entrée (entre 0 et 99).

Essayer de faire une modélisation la plus exacte possible ; faire notamment usage des droits d'accès là où cela semble pertinent. Pour chaque méthode, effectuer un affichage à l'écran de l'opération en cours et de l'acteur qui la réalise.

Note :

- Il existe de nombreuses variantes possibles. Commencer par un modèle très simple, et le faire évoluer pour se rapprocher de la situation « réelle » décrite.
- **Rappel :** pour pouvoir lire une donnée depuis le clavier, il faut avoir au préalable déclaré une variable de type **`Scanner`** :

```
private final static Scanner clavier = new Scanner(System.in);

...

int i = clavier.nextInt(); // par exemple pour lire un entier
```

avec au préalable l'importation **`import java.util.Scanner`** en début de programme.

Vous pourrez déclarer la variable **`clavier`** (ou équivalent) dans la classe **`Spectateur`** ici (c'est un point sur lequel nous reviendrons plus tard dans le cours).

2.3 Exemple de déroulement

```
Spectateur] (j'entre en scène)
Quel âge ai-je ? 35
Combien d'argent ai-je en poche (<100) ? 110
Combien d'argent ai-je en poche (<100) ? 12
[Spectateur] (j'ai un montant qui convient)
[Magicien] un petit tour de magie...
[Spectateur] (j'écris le papier)
[Assistant] (je lis le papier)
[Assistant] (je calcule mentalement)
[Assistant] J'annonce : 3397 !
[Magicien]  - hum... je vois que vous êtes agé de 35 ans
    et que vous avez 12 francs suisses  en poche !
```

2.4 Indications plus détaillées

Vous trouverez sur la page suivante quelques indications en vrac qui peuvent vous être utiles. Ne les lisez pas si vous voulez être complètement indépendant (but premier de l'exercice)...

2.4 Indications plus détaillées

- Les « objets » du programme ont déjà été suggérés : **`Magicien`**, **`Assistant`**, **`Spectateur`** et éventuellement **`Papier`** ; réfléchissez alors aux attributs : « qui *a* quoi ? » et surtout aux méthodes : « qui *fait* quoi ? »
- Essayez de définir une méthode pour chaque action élémentaire effectuée : premières actions du spectateur (demander l'âge et la somme d'argent), écrire sur le papier, montrer le papier ; pour l'assistant, lire le papier (qu'il doit donc recevoir), faire le calcul ; etc.
- Procédez par étapes, petit à petit.

Exercice 3 : géométrie

Ecrivez un programme **`Geometrie`** qui permet à l'utilisateur d'entrer les coordonnées (x, y) des sommets d'un triangle. Le programme affiche ensuite le périmètre du triangle ainsi qu'un message indiquant s'il s'agit d'un triangle isocèle. Votre programme doit être orienté objets.
Indications:

- Un triangle est isocèle si au moins deux côtés ont la même longueur.
- La formule pour calculer la distance entre deux points (x_1, y_1) et (x_2, y_2) est: racine carrée de $(x_1 - x_2)^2 + (y_1 - y_2)^2$.
- Java met à disposition la méthode **`Math.sqrt()`** pour calculer la racine carrée. Cette méthode prend un nombre non-négatif en paramètre.

  Exemple:

  ```
  double var = Math.sqrt(9.0);  // la valeur 3.0 sera affectée à var
  ```

Exemple d'affichage du programme pour un triangle isocèle:

```
Construction d'un nouveau point
   Veuillez entrer x : 0
   Veuillez entrer y : 0
Construction d'un nouveau point
   Veuillez entrer x : 2.5
   Veuillez entrer y : 2.5
Construction d'un nouveau point
   Veuillez entrer x : 0
   Veuillez entrer y : 5
Périmètre : 12.071067811865476
Le triangle est isocèle
```

Dans cet exercice, vous élaborerez un programme orienté objets de manière indépendante pour la première fois.

Vous trouverez sur la page suivante quelques indications en vrac qui peuvent vous être utiles. Ne les lisez pas si vous voulez être complètement indépendant (but premier de l'exercice)...

Indications

- Réfléchissez aux objets que vous aimeriez utiliser dans le programme. Vous pourriez par exemple représenter le triangle par une classe **`Triangle`** et ses points par une classe **`Point`**. Une troisième classe **`Geometrie`** pourrait héberger la méthode **`main`**.
- Réfléchissez aux variables (attributs) et méthodes d'instance qui seraient utiles pour les classes **`Triangle`** et **`Point`**.
- Un objet de type **`Point`** a typiquement les coordonnées **`x`** et **`y`**. Un objet de type **`Triangle`** a trois sommets qui peuvent être représentés par des objets de type **`Point`**.
- Les coordonnées des points peuvent par exemple être entrées dans le programme principal à l'aide de la méthode **`scanner.nextDouble()`**.
- Le périmètre d'un triangle peut être calculé comme la somme des distances entre les trois sommets.
