

Classes / Objets

- Télécharger l'archive `tp1.zip`.
- Extraire les fichiers de l'archive.
- Explorer le répertoire `tp1`.
- Lancer `BlueJ` et ouvrir le projet `tp1`.

Exercice 1

Ouvrir la classe `Lampe`

1. quels sont les attributs de la classe ?
2. Quels sont les constructeurs de la classe ?
3. Quels sont les méthodes de la classe ?
4. Quelle est l'interface de la classe ?
5. Créer une instance `lampe1` de la classe. Quel est l'état de l'instance ?
6. Créer une deuxième instance `lampe2` de la classe. Quel est l'état de l'instance ?
7. Appeler la méthode `allumer()` de l'instance `lampe1`. Relever l'état de l'instance `lampe1` puis l'état de l'instance `lampe2`.

Exercice 2

Ouvrir la classe `Piece`

1. quels sont les attributs de la classe ?
2. Quels sont les constructeurs de la classe ?
3. Quels sont les méthodes de la classe ?
4. Quelle est l'interface de la classe ?
5. Les classes `Lampe` et `Piece` sont-elles liées ?
6. Créer une instance `piece1` de la classe. Quel est l'état de l'instance ? Que se passera-t-il si on appelle la méthode `allumer()` ?
7. Ajouter un constructeur à la classe qui crée un objet `Lampe` et enregistre son identité dans l'objet `Piece` en cours de création.

Exercice 3

L'objectif de l'exercice est de définir une classe `Porte` modélisant la porte d'une maison. La seule propriété d'une porte est sa couleur (`String`). L'interface de la classe ne contient que deux méthodes :

- `String couleur()` qui retourne la couleur de la porte.
- `void peindre(String couleur)` qui change la couleur de la porte.

Enfin, à la création d'un objet `Porte`, on peut choisir la couleur de la porte. Il doit aussi être possible de créer un objet `Porte` avec l'instruction `new Porte()` (dans ce cas, la porte est de couleur blanche).

1. Définir la classe `Porte`.
2. Est-il possible d'écrire les instructions suivantes dans un programme ?

```
Porte p = new Porte();  
p.peindre(null);
```

Exercice 4

L'objectif est d'écrire une classe `Peintre` modélisant un peintre. Un peintre n'a aucune propriété. L'interface d'un peintre ne contient qu'une méthode de signature `void peindre(Porte porte, String couleur)` dont le comportement est le suivant :

- Le peintre peint la porte dans la couleur donnée en argument.

— Si aucune couleur n'est donnée (`couleur = null`), le peintre ne fait rien et notifie cela par un court message affiché dans une console (le choix du message est libre).

1. Définir la classe `Peintre`
2. Les classes `Porte` et `Peintre` sont-elles liées ?
3. Est-il possible que l'attribut `couleur` d'un objet `Porte` puisse être égale à `null` si sa couleur est modifiée par des objets `Peintre`.

Exercice 5

1. Quelle est l'interface de la classe `Moteur1` ?
2. Quelle est l'interface de la classe `Moteur2` ?
3. Appeler la méthode `main` de la classe de la classe `Programme1`
4. Appeler la méthode `main` de la classe de la classe `Programme2`
5. Laquelle des deux classes `Programme1` et `Programme2` doit-on modifier si on change le nom de l'attribut `puissance` des classes `Moteur1` et `Moteur2` ? Pourquoi n'a-t-on pas à modifier l'autre ?

Exercice 6

Pn veut créer un simple compteur ayant une valeur initiale nulle. Il s'agit de créer une classe `Compteur`. L'interface de la classe propose trois méthodes :

- `reinitialiser` qui remet à zéro le compteur.
- `incrémenter` qui augmente de un le compteur.
- `decrémenter` qui diminue de un le compteur. Un compteur ne peut pas être négatif. Si le compteur est à 0, la méthode ne fait rien.

1. Définir la classe `Compteur`
2. Ecrire une classe `DemoCompteur` qui contient un programme qui
 - (a) créera un compteur et affichera sa valeur,
 - (b) l'incrémentera 10 fois puis affichera sa valeur,
 - (c) le décrémentera 20 fois puis affichera sa valeur.L'affichage de ce programme doit donner quelque chose comme : 0 10 0

Exercice 7

L'objectif est de définir une classe `Point` modélisant un point du plan donné par ses coordonnées dans le plan, c'est-à-dire son abscisse x et son ordonnée y .

1. Définir un constructeur public initialisant les coordonnées d'un point :

```
Point p = new Point(0d,0d);
```
2. Définir les méthodes publiques suivantes :
 - (a) deux méthodes `getX` et `getY` qui retournent l'abscisse et l'ordonnée du point.
 - (b) une méthode `etat` qui retourne l'état de l'instance dans une chaîne de caractères.
 - (c) une méthode `deplacer` qui permet de déplacer le point : `p1.deplacer(1d,2d)`
3. Définir une méthode `comparer()` qui permet de comparer les coordonnées de deux points en retournant un booléen (`true` si les coordonnées des deux points sont identiques et `false` dans le cas contraire).
4. Écrire une classe `DemoPoint` qui
 - (a) créera deux points de même coordonnées,
 - (b) affichera l'état des deux points,
 - (c) comparera les coordonnées des deux points en affichant un message indiquant si les points ont les mêmes coordonnées ou non,
 - (d) déplacera un des deux points,
 - (e) affichera l'état des deux points,
 - (f) et comparera de nouveau les coordonnées des deux points en affichant un message indiquant si les points ont les mêmes coordonnées ou non.