TP n° 2

M02 : Dessin dans le terminal

Exercice 1 [Un rectangle]

Écrivez une classe Rectangle qui construit et affiche un rectangle dessiné dans le terminal avec des étoiles. Le rectangle peut être soit vide, soit plein (rempli avec des étoiles).

1. Vous fournirez 3 constructeurs :
   - En passant la hauteur, la largeur et si le rectangle est rempli ou non ;
   - En ne passant que la hauteur et si le rectangle est rempli ou non : dans ce cas, la largeur sera égale à la hauteur ;
   - En ne passant aucun argument : dans ce cas, des valeurs par défaut seront utilisées.
2. Écrivez une méthode publique `afficher()` qui affiche le rectangle sur le terminal.
3. Écrivez une méthode publique `vider()` et `remplir()` qui change le remplissage du rectangle comme son nom l'indique.
4. Écrivez une méthode publique `toString()` qui affiche une chaîne de caractères donnant les informations caractérisant le rectangle, par exemple *Rectangle de hauteur 30, de largeur 10, vide.*
5. Écrivez deux accesseurs `getLargeur()` et `geHauteur()` qui, comme leur nom l'indiquent, retournent les valeurs de la largeur et de la hauteur du rectangle.

Exercice 2 [Un cercle]

Écrivez une classe Cercle qui construit et affiche un cercle dessiné dans le terminal avec des étoiles. Le cercle peut être soit vide, soit plein (rempli avec des étoiles).

1. De façon similaire à la classe Rectangle de l'exercice précédent, vous fournirez 3 constructeurs :
   - En passant le diamètre et si le cercle est rempli ou non ;
   - En ne passant que si le cercle est rempli ou non : dans ce cas, la largeur sera égale à une valeur par défaut ;
   - En ne passant aucun argument : dans ce cas, des valeurs par défaut seront utilisées.
2. Écrivez une méthode publique `afficher()` qui affiche le cercle sur le terminal. Attention, un peu de géométrie est nécessaire pour calculer la longueur des lignes à afficher.
3. Écrivez une méthode publique `vider()` et `remplir()` qui change le remplissage du cercle comme son nom l'indique.
4. Écrivez une méthode publique `toString()` qui affiche une chaîne de caractères donnant les informations caractérisant le cercle, par exemple *Cercle de diamètre 20, plein.*
5. Écrivez un accesseur `getDiametre()` qui renvoie le diamètre du cercle.

Exercice 3 [Une classe d'utilitaires d'affichage]

Le but de cet exercice est d'écrire une classe fournissant des utilitaires d'affichage à l'écran (dans un terminal). Les manipulations de curser et de l'état du terminal affiché se font via des commandes utilisant des caractères spéciaux envoyés à la sortie standard (`System.out`). Ils commencent tous par le caractère d'échappement `ESC` (`\033[`).

1. Créez une classe Ecran. Il ne fait pas sens d'instancier cette classe : quelle particularité doit-elle donc avoir pour ne pas être instanciable ?

2. Il n'existe pas "directement" en Java de méthode `sleep()` permettant de suspendre l'exécution d'un programme un certain temps. On utilise la méthode `sleep()` de la classe `Thread` (Vous regarderez son utilisation dans la documentation Java). Écrivez une méthode `sleep()` de votre classe Ecran qui fait appel à la méthode `sleep()` de la classe `Thread`, prenant en paramètre le nombre de secondes à attendre. Attention : la méthode `sleep()` de la classe `Thread` soulève une exceptions. Vous l'attrapperz et afficherez la pile d'exécution avec la méthode `printStackTrace()`, et vous ne la propagerez pas.
3. Les commandes envoyées à la sortie standard commencent toutes par le caractère spécial `ESC`. Définissez un attribut de classe constant `ESC` que vous utiliserez par la suite.
4. On efface l'écran en envoyant le caractère ESC puis "2J" à la sortie standard, avec la méthode `System.out.print()`. La sortie standard étant bufferisée, on s'assure ensuite que ces caractères ont été transmis avec la méthode `System.out.flush()`.
5. Testez votre classe Ecran avec votre classe Rectangle et votre classe Cercle. Ecrivez une classe de test qui définit la méthode `main()` telle que le programme affiche un rectangle plein, attende 1 seconde, puis affiche ce rectangle vide, attende 1 seconde, affiche un cercle vide, attende 1 seconde, puis affiche ce cercle plein.
6. Écrivez une méthode de la classe Ecran qui monte le curser de N lignes. Pour cela, on envoie à la sortie standard : `ESC<N>A`, en remplaçant `<N>` par le nombre de lignes.
7. Écrivez une méthode de la classe Ecran qui descend le curser de N lignes. Pour cela, on envoie à la sortie standard : `ESC<N>B`, en remplaçant `<N>` par le nombre de lignes.
8. Écrivez une méthode de la classe Ecran qui décale le curser de N colonnes vers la droite. Pour cela, on envoie à la sortie standard : `ESC<N>C`, en remplaçant `<N>` par le nombre de colonnes.
9. Écrivez une méthode de la classe Ecran qui décale le curser de N colonnes vers la gauche. Pour cela, on envoie à la sortie standard : `ESC<N>D`, en remplaçant `<N>` par le nombre de colonnes.
10. Écrivez une méthode de la classe Ecran qui place le curser à une position donnée du terminal. Pour cela, on envoie à la sortie standard : `ESC<X>;<Y>D`, en remplaçant `<X>` par l'abscisse en nombre de colonnes et <Y> par l'ordonnée en nombre de lignes.
11. Écrivez une méthode de la classe Ecran qui sauve la position actuelle du curser. Pour cela, on envoie à la sortie standard : `ESCs`.
12. Écrivez une méthode de la classe Ecran qui met le curser à la dernière position sauvée. Pour cela, on envoie à la sortie standard : `ESCu`.
13. Écrivez des méthodes de la classe Ecran qui modifient la couleur d'affichage. Quelques exemples de valeurs utilisées pour les couleurs :
    - bleu : `ESC0;34m`
    - rouge : `ESC0;31m`
    - blanc : `ESC1;37m`
    - pas de couleur (couleur par défaut) : `ESC0m`

    Vous remarquerez qu'il s'agit des mêmes couleurs qu'utilisé par `bash` pour régler les couleurs d'un affichage, par exemple celui du prompt `PS1`.

Exercice 4 [Le drapeau français]

1. Modifiez votre classe Rectangle afin de permettre un affichage décalé sur le terminal : par exemple, si vous commencez à afficher un rectangle à une position donnée, la ligne suivante doit être affichée juste en-dessous, en tenant compte de la position du curser au début de l'affichage.
2. Affichez un drapeau français sur votre écran, constitué d'un rectangle bleu, d'un rectangle blanc et d'un rectangle rouge.

Exercice 5 [Cercles triés]

1. Modifiez votre classe Cercle pour qu'elle implémente l'interface `Comparable` : on dit qu'un cercle est "supérieur" à un autre lorsque son diamètre est supérieur. En cas d'égalité du diamètre, un cercle plein est supérieur à un cercle vide.
2. Écrivez un programme qui :
   - Demande à l'utilisateur de saisir les informations sur un nombre de cercles de son choix (diamètre, plein ou vide) ;
   - Crée un objet de la classe Cercle pour chacun des cercles saisis par l'utilisateur ;
   - Place les cercles dans une collection de votre choix ;
   - À la fin de la saisie, les cercles sont triés et affichés par ordre croissant à l'écran.

Exercice 6 [Mickey]

1. Modifiez votre classe Cercle afin de permettre un affichage décalé sur le terminal, de la même façon que pour la classe Rectangle dans l'exercice précédent.
2. Dessinez le visage de Mickey.