TP 4 - Méthodes sans résultat

Ce TP a pour objectif d'aborder la notion de méthode sans résultat. Il est constitué de deux parties.

- La première partie consiste à manipuler des classes `Crayon` et `Feuille` à partir de leur documentation pour comprendre **comment utiliser des méthodes fournies**.
- La seconde partie reprend la classe `Personne` et propose d'ajouter des méthodes simples à cette classe.

Les deux parties sont censées prendre environ 1h chacune.

1 Classe `Crayon` et `Dessin`

L'objectif de cette est d'aborder la notion de méthode à travers quelques classes graphiques simples d'utilisation :

- la classe `Feuille` représente une feuille sur laquelle dessiner ;
- la classe `Crayon` représente un crayon permettant de tracer des traits.

Chacune de ces classes dispose de commentaires javadoc. Le résultat de la javadoc vous est fourni en annexe.

1.1 Création des objets

Afin de pouvoir dessiner correctement, il est nécessaire de disposer

- d'un objet `Feuille` sur lequel dessiner ;
- d'un objet de type `Crayon`.

Question 1

A l'aide de la javadoc fournie, écrire un `main` qui construit une feuille de taille (600x400) et un crayon de couleur noire.

1.2 Utilisation du crayon

La classe `Crayon` permet de dessiner sur une feuille grâce à différentes méthodes décrites dans la javadoc.

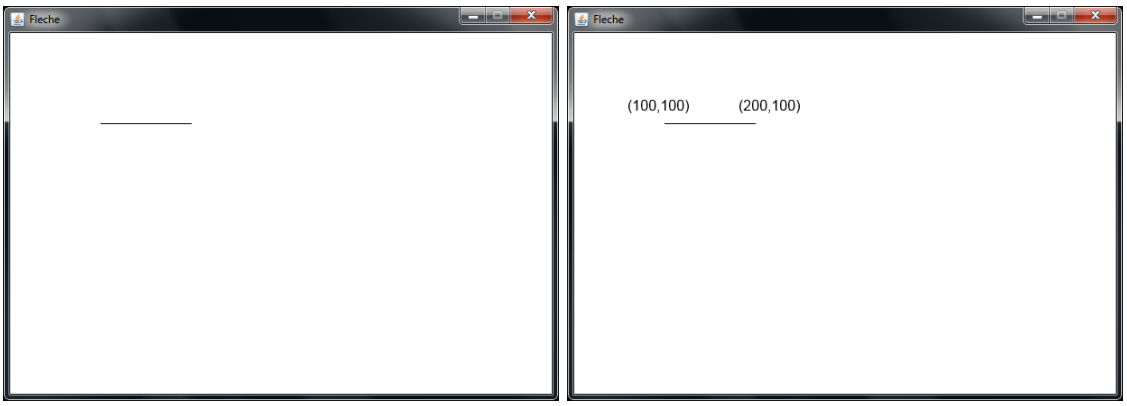

FIGURE 1 – Trait à dessiner

Question 2

A l'aide de la javadoc fournie, utilisez la classe `Crayon` pour dessiner un trait horizontal partant des coordonnées (100,100) vers les coordonnées (200,100).

1.3 Dessin évolué

Dessin de flèche

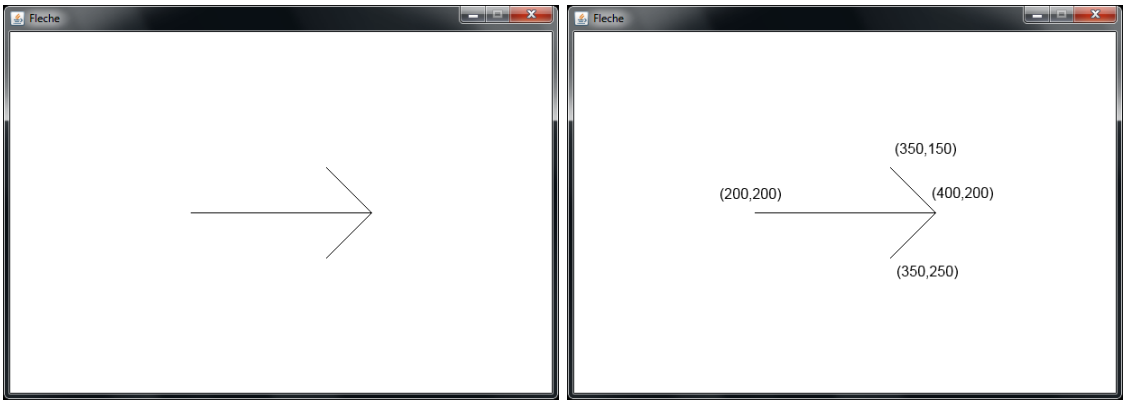

FIGURE 2 – Fleche à dessiner

Question 3

A l'aide de la javadoc fournie, écrivez un `main` dans la classe `DessinFleche` pour dessiner la flèche de la figure 2.

Changement de Couleur

Java fournit la classe `Color`. Cette classe est complexe et possède de nombreuses méthodes décrites dans l'API :

`https://docs.oracle.com/javase/8/docs/api/java/awt/Color.html`

Pour utiliser la classe il est nécessaire d'importer le package `java.awt`.

```
1 import java.awt.*;
```

Parmi d'autres choses, elle possède un constructeur qui prend 3 entiers `R`, `G` et `B` en paramètre et qui construit la couleur RGB correspondante

- avec `R` désignant la composante rouge ;
- avec `G` désignant la composante vert ;
- avec `B` désignant la composante bleu.

Question 4

Créer une variable de type `Color` correspondant à la couleur rose avec pour composants (255,0,255).

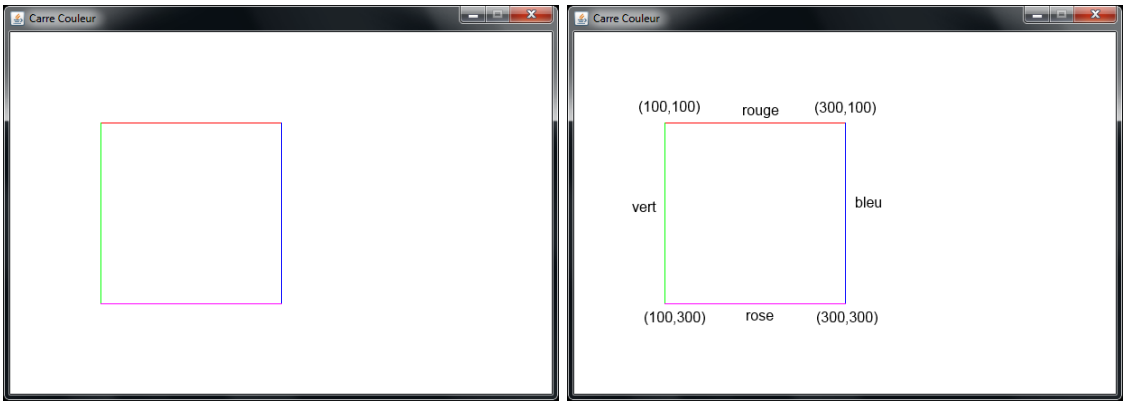

FIGURE 3 – Image multicolore à dessiner

Question 5

En utilisant plusieurs crayons, dessinez la figure 3 dans le `main` de la classe `DessinCouleur`.

Question 6

Si vous le souhaitez, vous pouvez continuer à utiliser ces classes en dehors du TP pour vous familiariser avec l'utilisation de méthodes.

1.4 Test de la classe `Crayon` (Optionnel)

Comme illustré dans le TP précédent, on souhaite recourir à un outil de tests unitaires afin de vérifier le bon fonctionnement de notre programme au cours de son développement. L'outil utilisé ici est la bibliothèque `libtest` introduite au TP 2.

Dans ce contexte, copier l'ensemble du répertoire `libtest` du TP2 dans votre répertoire de travail (là où se trouvent vos classes `Feuille` et `Crayon`).

Ajoutez y un fichier `TestCrayon.java` contenant le squelette de code java suivant :

```
1  import static libtest.Lanceur.lancer;
2  import static libtest.OutilTest.*;
3
4  /**
5   * classe chargee de tester les constructeurs de Crayon
6   */
7  public class TestCrayon {
8
9    /**
10    * test du constructeur vide
11    */
12   public void test1_constructeurVide()
13   { // A FAIRE }
14
15   /**
16    * test du constructeur avec deux parametres
17    */
18   public void test2_constructeurParam()
19   { // A FAIRE }
20
21   /**
22    * lancement des tests
23    */
24   public static void main(String args[])
25   {
26     lancer(new TestCrayon(),args);
27   }
28 }
```

On voit dans ce fichier qu'il y a deux méthodes de test, destinées respectivement à vérifier le bon fonctionnement du constructeur de crayon sans paramètre (constructeur vide) et du constructeur de crayon avec paramètre.

Sachant que la classe `Crayon` contient entre autre les attributs suivants :

```
1  import java.awt.Color;
2
3  public class Crayon {
4          int abscisse;
5          int ordonnee;
6          Color couleur;  // Color est un type prédéfini du langage java
```

et les deux constructeurs suivants :

— un constructeur vide servant à construire un crayon positionné au "centre" (coordonnées $(0, 0)$) de la feuille et de couleur noire (couleur prédéfinie `Color.BLACK`) ;
— un constructeur avec paramètre servant à construire un crayon positionné au "centre" de la feuille avec une couleur de son choix (soit choisie parmi les couleurs prédéfinies de la classe `Color`[1], soit construite par exemple à partir d'un triplet d'entiers au format RGB),

Question 7
Réalisez les tests 1 et 2 décrits ci-dessous.

Méthode testée	cas	numéro du test
constructeur vide	cas normal	test 1
constructeur avec paramètre	cas normal	test 2

2 Classe `Personne` et méthodes sans résultat

2.1 Classe `Personne` fournie

On vous fournit sur arche la classe `Personne` déjà vue en cours (avec les classes `MaDate` et `Adresse`).

```
/**
 * classe Personne
 */
public class Personne {
    // attributs identite
    String nom;
    String prenom;

    // attribut objet adresse
    Adresse adresse;

    //attribut date de naissance
    MaDate dateN;

    /**
     * constructeur par defaut
     */
    public Personne() {
      // une identite anonyme
      this.nom        = "Anonyme";
      this.prenom     = "Anonyme";

      // une adresse par defaut
      this.adresse    = new Adresse();
```

1. Voir `https://docs.oracle.com/javase/8/docs/api/java/awt/Color.html` pour la liste des couleurs prédéfinies.

```
25
26        // une date de naissance par defaut
27        this.dateN  = new MaDate();
28      }
29
30      /**
31       * constructeur avec des parametres
32       *
33       * @param nom nom de la personne
34       * @param prenom prenom de la personne
35       * @param adr adresse de la personne
36       * @param naiss date de naissance de la personne
37       */
38      public Personne(String nom, String prenom, Adresse adr, MaDate
             naiss) {
39        // initialise l'identite
40        this.nom        = nom;
41        this.prenom     = prenom;
42
43        // ATTENTION : eviter les effets de bord et gerer adr null
44        // construit une adresse par copie
45        this.adresse    = new Adresse(adr);
46
47        // ATTENTION : eviter les effets de bord et gerer naiss null
48        // une date de naissance par copie
49        this.dateN  = new MaDate(naiss);
50      }
51    }
```

2.2 Méthode `afficher`

On souhaite pouvoir afficher la valeur d'un objet personne sous le format suivant :

```
------
<Nom> <Prenom>
- né(e) le <jour>/<mois>/<annee>
- habite <Rue> à <Ville> (<codePostal>)
------
```

où `<XXX>` désigne la valeur de l'attribut `XXX`.

Question 8

Écrire la méthode `afficher` de la classe `Personne` pour afficher les attributs de l'objet personne selon le format souhaité.

2.3 Méthodes Setter

On appelle Setter des méthodes qui ont pour objectif de modifier un attribut. Elles ont généralement un nom de la forme `SetAttribut` où `Attribut` est remplacé par le

nom de l'attribut en question. Ces méthodes permettent d'ajouter des vérifications avant de modifier directement la valeur d'un attribut.

Question 9

Réfléchir au profil de la méthode `setNom` de la classe `Personne` qui a pour objectif de modifier le nom de la personne sur laquelle cette méthode s'applique. Écrire ensuite cette méthode.

Question 10

Réfléchir au profil de la méthode `setDate` de la classe `Personne` qui a pour objectif de modifier la date de naissance de la personne. Écrire ensuite cette méthode. Vous ferez attention à bien vérifier que le paramètre est valide. Vous éviterez aussi tout effet de bord.

Question 11

Écrire un programme `ProgChangerDate` qui
- créé une personne `p` par défaut,
- affiche la personne créée,
- créé une date `d`,
- puis modifie la date de naissance de la personne `p` pour lui donner la date `d` en utilisant la méthode `setDate`.

Vous vérifierez ensuite que votre code n'engendre pas d'effet de bord : la modification de la date `d` ne doit pas changer la date de naissance de la personne `p` créée.

Question 12

Réfléchir au profil de la méthode `changerDateJour` de la classe `Personne` qui a pour objectif de modifier le jour de la date de naissance de la personne. Écrire ensuite cette méthode. Vous ferez attention à bien vérifier que la valeur passée en paramètre est bien valide.

Question 13

Tester votre classe `Personne` avec la classe `TestChangerDate` fournie.

2.4 Méthodes `seMarier`

On souhaite changer le nom d'une personne lorsqu'elle se marie à une autre personne.

Question 14

Réfléchir au profil de la méthode `seMarier` qui consiste à modifier le nom de la personne qui se marie pour lui affecter le nom de la personne avec laquelle elle se marie.

Question 15

Écrire la méthode `seMarier`.

Question 16

Tester la méthode `seMarier` en utilisant la classe de test `TestSeMarier` fournie sur arche. Les différents tests doivent réussir.

2.5 Méthode changer `Adresse`

On souhaite permettre à une personne de modifier son adresse à une adresse existante donnée. Bien que cette méthode puisse s'appeler `setAdresse`, on préfère l'appeler `demenager`.

Question 17

Ècrire la méthode `demenager`.

Question 18

Tester votre méthode `demenager` avec la classe de test fournie `TestDemenager`. Tous les tests sont juste et doivent passer. Corriger votre code sinon.