TP7

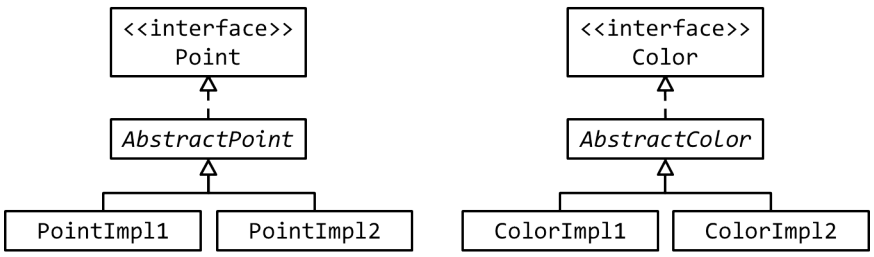
Exercice – factorisation de code

Dans un TP précédent, nous avons écrit deux implémentations d'une pile de formes géométriques.

1. Factorisez le code des classes implémentant les interfaces `Formes` et `PileFormes`.
2. Exécutez le programme de test (sans avoir à le modifier).
3. Ajoutez un attribut représentant le centre de gravité d'une forme géométrique dans le plan.
4. Ajoutez une méthode `translater` qui déplace le centre de gravité de la forme géométrique (et donc la figure elle-même) suivant un vecteur passé en paramètre.
5. Complétez le programme de test.

Exercice – délégation pour simuler l'héritage multiple

Soit le diagramme de classes suivant :



Nous souhaitons créer des points colorés qui soient de type `PointImpl1` ou `PointImpl2`, avec des couleurs définies par `ColorImpl1` ou `ColorImpl2`. Pour cela, nous aimerions définir un type `PointWithColor` qui hérite, par exemple, à la fois de `PointImpl1` et `ColorImpl2`. L'héritage multiple n'étant pas supporté en Java, nous allons le simuler de trois manières différentes, en créant une classe `PointWithColor` qui implémente les deux interfaces `Point` et `Color`.

1. Soit la classe `PointWithColor1` :

```
class PointWithColor1 implements Color, Point {
  private double[] coords; // (abscisse, ordonnee)
  private int[] color; // RGB
  ...
}
```

   Que pouvez-vous dire de cette solution ? Précisez ses inconvénients.

2. Soit la classe `PointWithColor2` :

```
class PointWithColor2 implements Color, Point {
  private PointImpl1 p;
  private ColorImpl2 c;
  ...
}
```

   Que pouvez-vous dire de cette deuxième solution ? Quelles sont ses limites ?

3. Comment transformer la classe `PointWithColor2` pour répondre à notre problème ?

Exercice – histoires des codes secrets (Simon Singh)

La cryptographie est l'ensemble des méthodes utilisées pour cacher le sens d'un message. Elle peut être divisée en deux branches : la transposition et la substitution.

Dans la transposition, les lettres du message sont simplement redistribuées, produisant une anagramme. Un exemple d'un tel cryptosystème est la transposition en dents de scie. Il s'agit d'écrire le message sur deux lignes, une lettre sur la ligne supérieure, une lettre sur la ligne inférieure. Il suffit ensuite d'enchaîner la suite des lettres de la ligne inférieure à celle de la ligne supérieure, pour réaliser le message crypté, comme illustré ci-dessous :

```
J V C S S P R
 A A E T U E
```

Dans cet exemple, le message clair `JAVACESTSUPER` devient le message crypté `JVCSSPRAAETUE`. Un autre exemple de transposition consiste à faire jouer des paires de lettres, afin que la première et la deuxième échangent leur place, de même pour la troisième et la quatrième, et ainsi de suite, comme illustré ci-dessous :

```
JA VA CE ST SU PE R
AJ AV EC TS US EP R
```

Dans cet exemple, le message clair `JAVACESTSUPER` devient le message crypté `AJAVECTSUSEPR`. Dans ces deux exemples, le destinataire du message n'aura qu'à inverser le procédé.

Dans la substitution, chaque lettre du message est remplacée par une lettre différente, mais garde sa place dans le message. Il y a donc un alphabet clair utilisé pour écrire le message original, et un alphabet chiffré pour les lettres substituées à celles d'origine. Un exemple d'un tel cryptosystème (issu du *Kàma-sùtra*) consiste à apparier au hasard les lettres de l'alphabet et à substituer ensuite dans le message original la nouvelle lettre de la paire à celle d'origine, comme illustré ci-dessous :

```
ACEJPRSTUVXYZ
||||||||||||
BDFGHIKLMNOQW
```

Dans cet exemple, le message clair `JAVACESTSUPER` devient le message crypté `GBNBDFKLKMHFI`. Un autre exemple est le chiffre décalé de César. Il s'agit de remplacer chaque lettre du message par une lettre 3 places plus loin dans l'alphabet, comme illustré ci-dessous :

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
DEFGHIJKLMNOPQRSTUVWXYZABC
```

Dans cet exemple, le message clair `JAVACESTSUPER` devient le message crypté `MDYDFHVWVXSHU`. Ce chiffrement peut être généralisé avec n'importe quel nombre compris entre 1 et 25 pour le décalage. Nous pouvons également redisposer au hasard les lettres de l'alphabet pour créer un alphabet chiffré. Dans ces exemples de substitution, l'alphabet chiffré reste le même au cours de tout le chiffrement. On parle de substitution monoalphabétique. Il est également possible d'utiliser un alphabet chiffré qui change au cours du chiffrement. Ce changement s'exécute selon une clé. On parle de substitution polyalphabétique. Un exemple est le chiffre de Vigenère qui utilise 26 alphabets différents (formant le carré de Vigenère), chacun étant un alphabet décalé selon le chiffre de César, et un mot clé établit quel alphabet chiffré sera utilisé pour crypter chaque lettre du message. Par exemple, utilisons le mot clé "BLAISE" pour crypter un message, comme illustré ci-dessous :

```
clé    : BLAISEBLAISEB
clair  : JAVACESTSUPER
crypté : KLVIUITESCHIS
```

Nous souhaitons écrire une application permettant de crypter et décrypter des messages, en utilisant les différents cryptosystèmes présentés ci-dessus. Nous considérons que les messages à crypter sont constitués uniquement de lettres majuscules sans espace.

1. Proposez un diagramme de classes qui modélise le problème.
2. Écrivez en Java une implémentation des classes du diagramme, en évitant les duplications de code.
3. Écrivez un programme de test.
4. Écrivez une méthode de classe qui supprime les espaces d'une chaîne de caractères passée en paramètre et la met en majuscule.
5. Réflexion :
   (a) Quels accesseurs contiennent vos classes ?
   (b) Toutes vos méthodes devraient-elles pouvoir être redéfinies dans une sous-classe ?