Aix*Marseille université
Initiative d'excellence
Année 2018-19

Domaine Sciences et Technologies
Licence informatique

Programmation 2 : TD 4
Code UE : SIN3U02

Interfaces

1 Interface `StringTransform`

1. Écrivez l'interface `StringTransform` contenant l'unique méthode `String applyTo(String s)`. Les implémentations de cette méthode doivent produire une nouvelle chaîne de caractères à partir de la chaîne `s` puis retourner le résultat de cette transformation.
2. Écrivez les classes suivantes implémentant l'interface `StringTransform` :
   - `UpperCaseStringTransform` convertit les caractères de `s` en majuscules.
   - `LowerCaseStringTransform` convertit les caractères de `s` en minuscules.
   - `PrefixStringTransform` conserve les `n` premiers caractères de `s`. La valeur de `n` est fournie lors de la construction d'une instance de la classe.
   - `PostfixStringTransform` conserve les `n` derniers caractères de `s`. La valeur de `n` est fournie lors de la construction d'une instance de la classe.

   On pourra utiliser les méthodes `toUpperCase` et `toLowerCase` ainsi que les deux méthodes `substring` de la classe `String` :
   - `String substring(int index)` : returns a string that is a substring of this string. The substring begins with the character at the specified `index` and extends to the end of this string.
   - `String substring(int beginIndex, int enIndex)` : returns a string that is a substring of this string. The substring begins at the specified `beginIndex` and extends to the character at index `endIndex - 1`. Thus the length of the substring is `endIndex - beginIndex`.
3. Écrivez la méthode statique `String[] applyTransformToStrings(String[] strings, StringTransform transform)` qui applique la transformation `transform` aux chaînes du tableau `strings` et qui retourne un tableau contenant les chaînes transformées.
4. Écrivez la classe `CompositeStringTransform` qui implémente l'interface `StringTransform` et qui applique successivement sur la chaîne `s` les transformations du tableau `StringTransform[] transforms` passé au constructeur.

2 Interface `Shape`

On souhaite utiliser l'interface suivante pour des formes géométriques :

```
public interface Shape{
    double getPerimeter();
    void draw(Painter painter, Color color);
    Shape translate(int dx, int dy);
    double getArea();
}
```

Écrivez les classes suivantes qui implémentent l'interface `Shape` et que l'on peut construire à partir d'un certain nombre de points.

- `Triangle` : construit à partir de trois points qui correspondront à ses sommets.
- `Rectangle` : construit à partir de deux sommets qui correspondront à deux sommets diamétralement opposés du Rectangle (les cotés étant supposés parallèles aux axes)

On utiliser la classe `Point` ci-dessous. Dessiner une figure géométrique consiste à tracer les traits entre les sommets reliés dans la figure. Les transformations de figure devront renvoyer un nouvelle figure sans modifier l'ancienne.

Pou l'aire S d'un triangle, on pourra utiliser la formule suivante dans laquelle a, b et c sont les longueurs des cotés : $S = \sqrt{p(p-a)(p-b)(p-c)}$ avec $p = \frac{a+b+c}{2}$.

```
public class Point {
    public final int x, y;
    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }
    public double distanceTo(Point p){
        return Math.hypot(this.x - p.x, this.y - p.y);
    }
    void drawLine(Point p, Painter painter, Color color){
        painter.addLine(x, y, p.x, p.y, color);
    }
    public Point translate(int dx, int dy) {
        return new Point(x + dx, y + dy);
    }
}
```

3 Interface `Formula`

Dans cet exercice, vous allez écrire une interface `Formula` qui correspond à une formule mathématique. Une formule est au départ définie récursivement de la manière suivante :

— une variable (classe `Variable`) est une formule,
— l'addition de deux formules (classe `Sum`) est une formule,
— la multiplication de deux formules (classe `Product`) est une formule.

À partir d'une formule, On souhaite pouvoir obtenir sa valeur et une représentation sous forme de chaîne de caractères.

Le code ci-dessous décrit le comportement voulue pour les classes `Variable`, `Sum` et `Product` implémentant l'interface formule :

```
Variable x = new Variable("x", 2.5);
Variable y = new Variable("y", 4);
Formula formula = new Sum(x, new Product(y, new Sum(x, y)));
System.out.println(formula.asString()); // "(x+(y*(x+y))"
System.out.println(formula.asValue()); // "28.5"
x.set(5);
System.out.println(formula.asValue()); // "41.0"
```

1. Décrivez l'interface `Formula`.
2. Écrivez les classes `Variable`, `Sum` et `Product`.
3. Modifiez les classes `Sum` et `Product` de façon à réaliser les opérations sur un ensemble de formules. Ces formules sont passées au constructeur sous la forme d'un tableau. Faites en sorte que le code ci-dessus fonctionne toujours après la modification.
4. Ajoutez les classes suivantes :
   `AbsoluteValue` : valeur absolue $|f|$, `Square` : carré f^2, `SquareRoot` : racine carrée $\sqrt{f}$, `Power` : puissance f^k, `Minimum` : minimum $\min(f_1, f_2, \ldots, f_k)$, `Maximum` : maximum $\max(f_1, f_2, \ldots, f_k)$.