

Documentation de la classe `String`

Question 1 : D'après la documentation, quelle est la valeur de l'expression `"Goodenough".substring(4).substring(0,2).toUpperCase()` ?

D'après la documentation, on a les valeurs suivantes :

1. `"Goodenough".substring(4)` a pour valeur `"enough"`
2. `"enough".substring(0,2)` a pour valeur `"en"`
3. `"en".toUpperCase()` a pour valeur `"EN"`

L'expression `"Goodenough".substring(4).substring(0,2).toUpperCase()` a donc pour valeur `"EN"`.

Question 2 : La classe `String` contient deux méthodes nommées `substring`. À quelle condition a-t-on le droit de définir deux méthodes ayant le même nom ? Comment appelle-t-on cette notion ?

On a droit d'avoir plusieurs méthodes ayant le même nom si les type des arguments des méthodes sont différents. Plus formellement, il faut que la séquence des types soient différentes. On appelle cette notion la surcharge de méthode.

Interface `StringTransform`

Question 3 : Écrivez le code de l'interface `StringTransform`.

```
public interface StringTransform {  
    public String applyTo(String s);  
}
```

Question 4 : Écrivez le code des classes `UpperCaseStringTransform`, `LowerCaseStringTransform`, `PrefixStringTransform` et `PostfixStringTransform`.

```

public class UpperCaseStringTransform implements StringTransform{
    public String applyTo(String s) {
        return s.toUpperCase();
    }
}

public class LowerCaseStringTransform implements StringTransform {
    public String applyTo(String s) {
        return s.toLowerCase();
    }
}

public class PrefixStringTransform implements StringTransform {
    int n;
    public PrefixStringTransform(int n) {
        this.n = n;
    }
    public String applyTo(String s) {
        return s.substring(0, n);
    }
}

public class PostfixStringTransform implements StringTransform{
    int n;
    public String applyTo(String s) {
        return s.substring(s.length()-n);
    }
    public PostfixStringTransform(int n) {
        this.n = n;
    }
}

```

Question 5 : Écrivez le code de la classe StringTransforms.

```

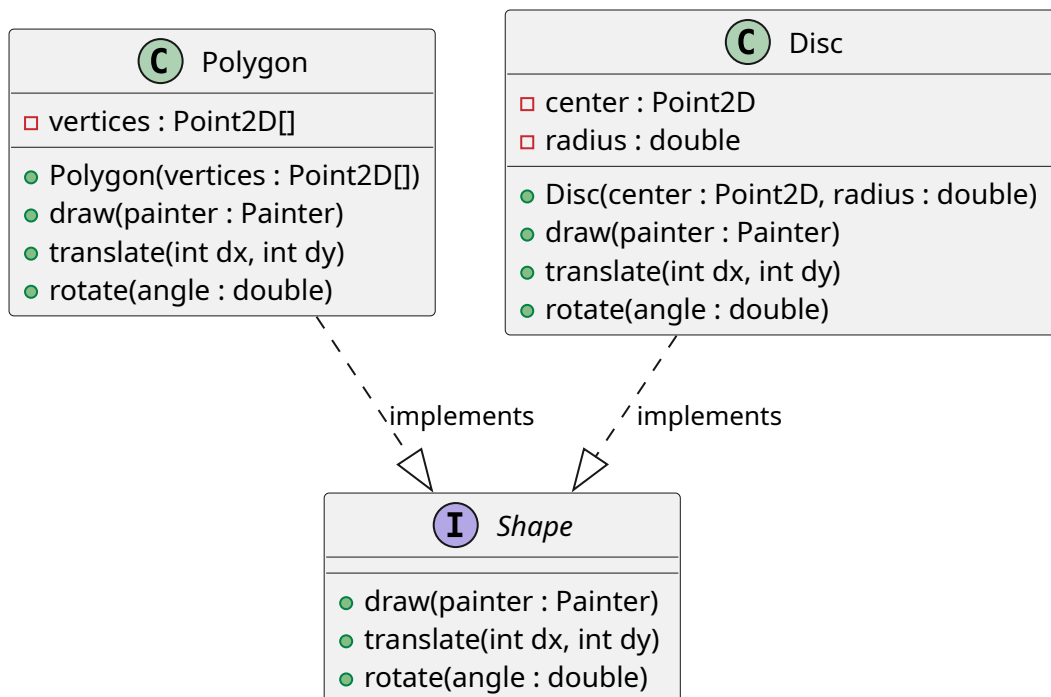
public class StringTransforms{
    public static String[] applyTransformToStrings(String[] strings,
                                                    StringTransform transform){
        String[] transformedStrings = new String[strings.length];
        for(int index = 0; index < transformedStrings.length; index++){
            transformedStrings[index] = transform.applyTo(strings[index]);
        }
        return transformedStrings;
    }
}

```

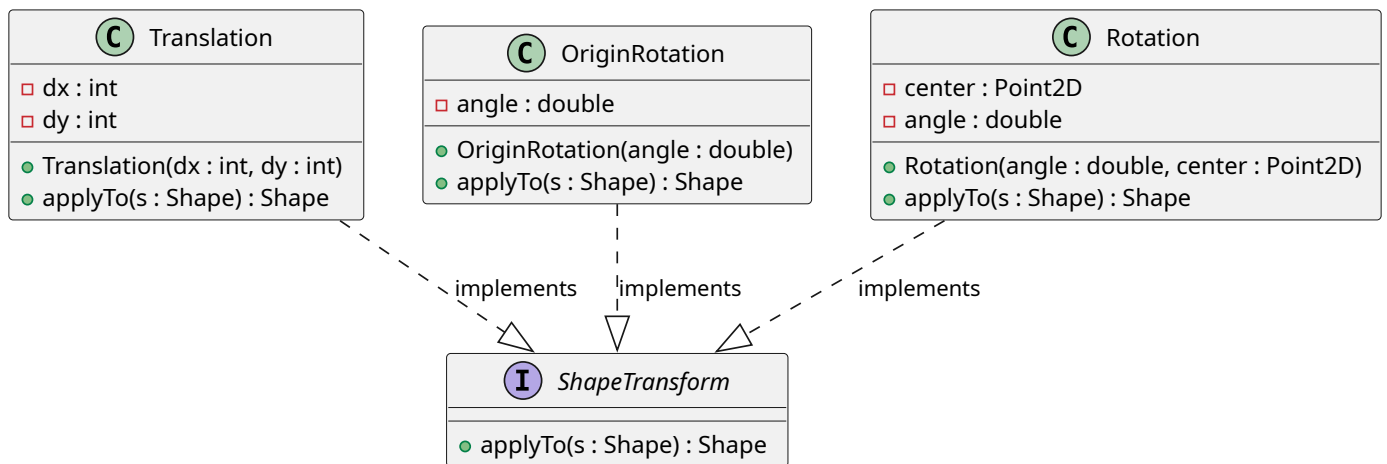
Question 6 : Écrivez le code de la classe `CompositeStringTransform`.

Formes géométriques

Question 7 : Dessinez le diagramme des classes `Polygon` et `Disc` en ajoutant une interface (regroupant leurs méthodes communes) qu'elles devront implémenter.



Question 8 : Dessinez le diagramme des classes `Translation`, `OriginRotation` et `Rotation` en ajoutant une interface (regroupant leurs méthodes communes) qu'elles devront implémenter.



Question 9 : Donner le code de la classe `Rotation` en supposant que les classes `Point2D`, `Translation` et `OriginRotation` sont déjà codées.

```

public class Rotation{
    private final Point2D center;
    private final double angle;

    public Rotation(double angle, Point2D center){
        this.angle = angle;
        this.center = center;
    }

    public void applyTo(Shape s){
        new Translation(-center.getX(),-center.getY()).applyTo(s);
        new Rotation(angle).applyTo(s);
        new Translation(center.getX(),center.getY()).applyTo(s);
    }
}

```