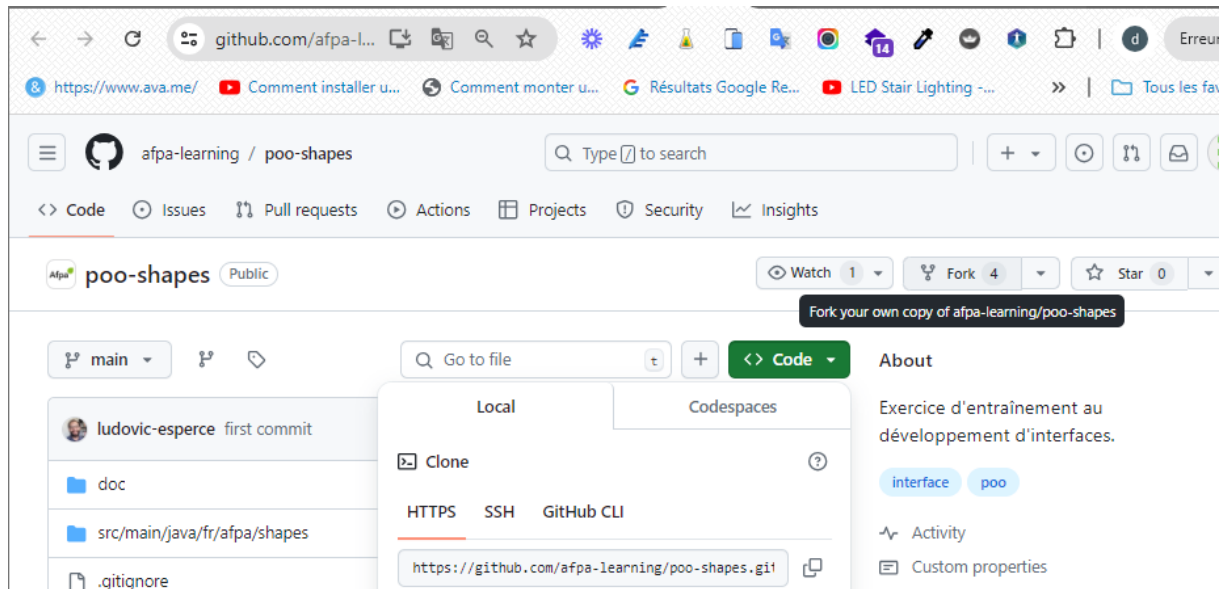
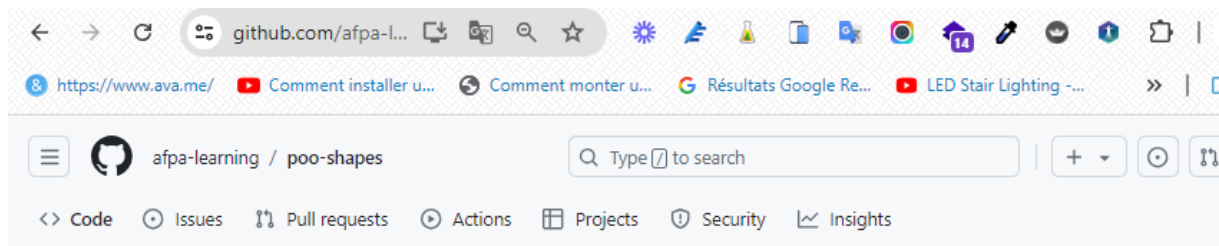


TP4 POO Shapes l'implémentation d'interfaces






Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. [View existing forks.](#)

Required fields are marked with an asterisk (*).

Owner * Repository name *

 zorro82 / zorro82/TP4-POO-Shapes

✓ Your new repository will be created as zorro82-TP4-POO-Shapes-implementation-d-interface.
The repository name can only contain ASCII letters, digits, and the characters `.`, `-`, and `_`.


By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description (optional)

Exercice d'entrainement au développement d'interfaces.

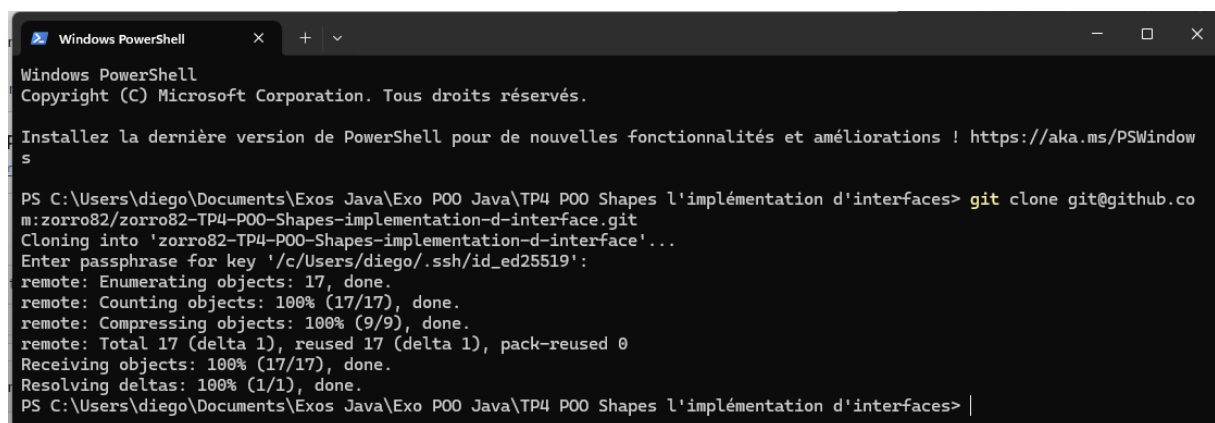
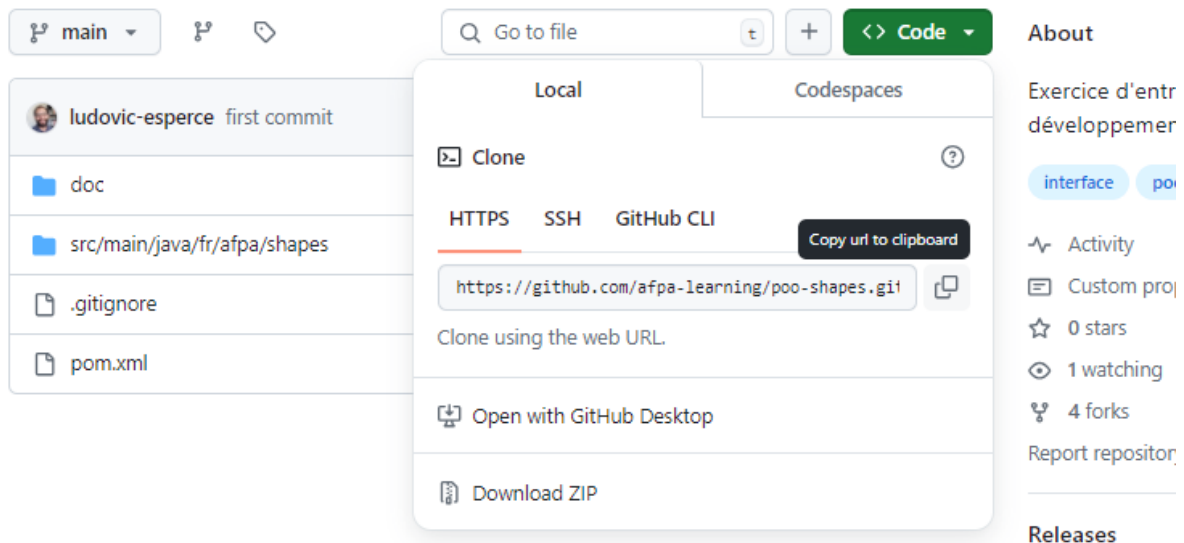
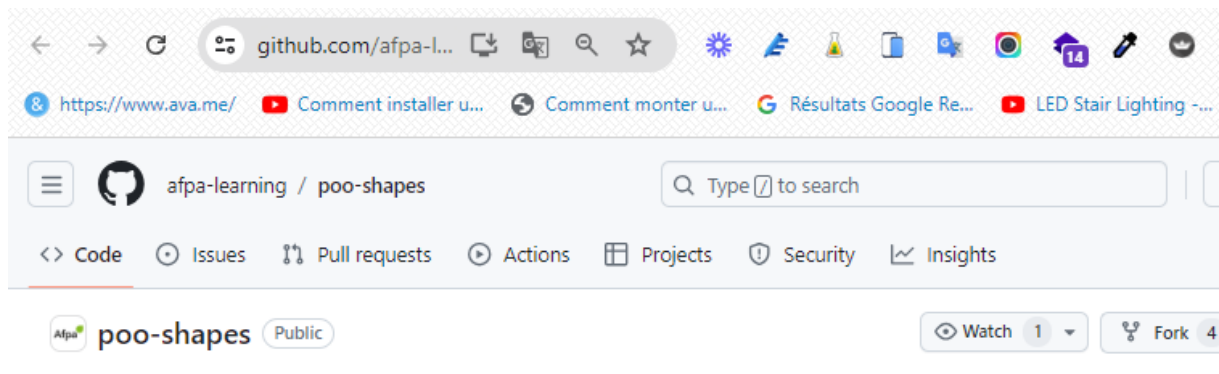
☒ Copy the `main` branch only

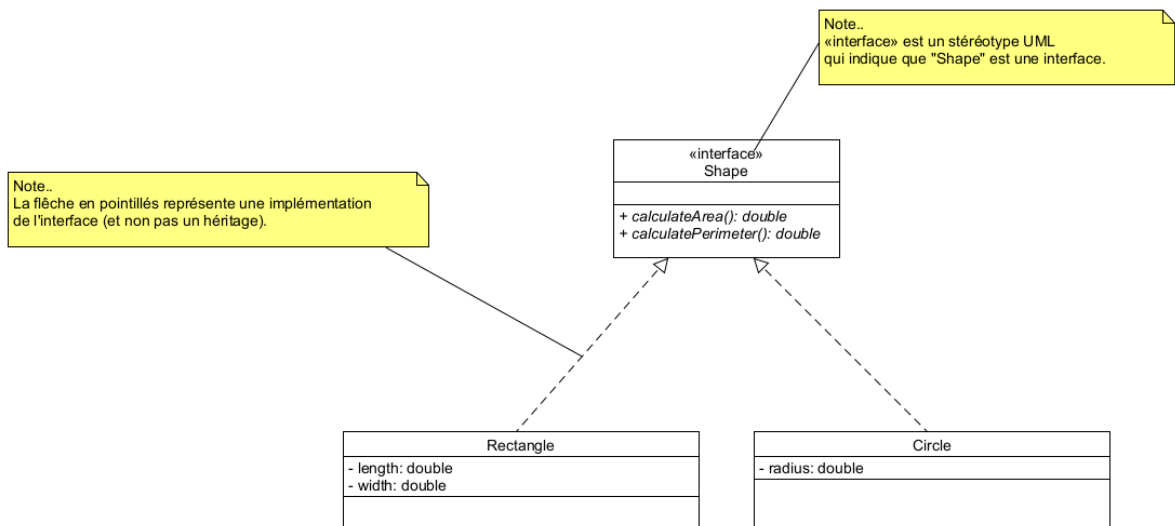
Contribute back to afpa-learning/poo-shapes by adding your own branch. [Learn more.](#)

 You are creating a fork in your personal account.

Create fork

Documents > Exos Java > Exo POO Java > TP4 POO Shapes l'implémentation d'interfaces > zorro82-TP4-POO-Shapes-implementation-d-interface >				
Trier v Afficher v ...				
Nom				
.git		Modifié le	Type	Taille
doc		09/07/2024 16:07	Dossier de fichiers	
src		09/07/2024 16:07	Dossier de fichiers	
.gitignore		09/07/2024 16:07	Fichier source Git l...	1 Ko
pom.xml		09/07/2024 16:07	Fichier source XML	3 Ko





Classe GeomerticShapeMain.java :

```
package fr.afpa.shapes;
/*
    Objectif : développer des classes représentant des formes géométriques
    (Rectangle, Cercle et Triangle)
    Pour chacune de ces classes il faudra implémenter deux méthodes :
        - une qui renverra le périmètre -> double calculatePerimeter()
        - une qui renverra l'aire -> double calculateArea()

    Pour apprendre à implémenter une interface vous pouvez vous référer au
    diaporama présenté en formation
    Vous pouvez également regarder la vidéo suivante :
    https://www.youtube.com/watch?v=OkEwPtRaqY4

    TODO implémentez une classe "Rectangle" comprenant les attributs présentés
    sur le diagramme UML contenu dans le sous-dossier "doc"
    Le diagramme UML est sous format "uxf". Pour pouvoir le lire il vous
    faudra le logiciel UMLET
    UMLET est disponible à l'adresse suivante : https://www.umlet.com/

    TODO implémentez une classe "Circle" comme présentée par le diagramme UML

    TODO créez une interface nommée "Shape" comprenant deux méthodes
    abstraites de calcul :
        - double calculatePerimeter()
        - double calculateArea()

    TODO faites en sorte que la classe "Rectangle" implémente l'interface
    "Shape" et implémentez les deux méthodes
    Rappel de calcul :
        - perimetre_rectangle = 2 * longueur + 2 * largeur
        - aire_rectangle = longueur * largeur

    TODO faites en sorte que la classe "Circle" implémente l'interface "Shape"
    et implémentez les deux méthodes
    Rappel de calcul :
        - périmètre du cercle = 2 *  $\pi$  * rayon
        - aire_cercle =  $\pi$  * rayon^2

    La valeur  $\pi$  peut être retrouvée en Java en utilisant "Math.PI"
    La puissance de 2 peut être effectuée en utilisant la méthode static
    "pow" de la classe "Math"
    -> plus d'informations par ici
    https://codegym.cc/fr/groups/posts/fr.575.math-pi-en-java

*/
class GeometricShapeMain
{
```

```

    public static void main(String[] args)
    {
        // TODO instancier plusieurs objets des classes Rectangle et Circle (2
instances de chaque)

        // TODO ajouter ces objets à une instance de la classe "ArrayList"
        // plus d'information sur la classe "ArrayList" ->
https://info.clg.qc.ca/java/structures/array-list

        // TODO boucler sur les éléments de la liste afin d'afficher le
périmètre et l'aire de chaque objet
    }
}

```

Classe Shape.java :

```

package fr.afpa.shapes;

// TODO ajouter ICI l'interface "Shape"
// pour apprendre à déclarer une interface -> https://gayerie.dev/epsi-b3-java/langage\_java/interface.html

```

Classe Rectangle.java

```

package fr.afpa.shapes;

// TODO implémenter la classe Rectangle
// se référer au diagramme UML contenu dans le sous-dossier "doc" pour
retrouver les informations concernant ses attributs

```

Classe Circle.java :

```

package fr.afpa.shapes;

// TODO implémenter la classe Circle
// se référer au diagramme UML contenu dans le sous-dossier "doc" pour
retrouver les informations concernant ses attributs

```

Objectif : développer des classes représentant des formes géométriques (Rectangle, Cercle et Triangle)

Pour chacune de ces classes il faudra implémenter deux méthodes :

- une qui renverra le périmètre -> double calculatePerimeter()
- une qui renverra l'aire -> double calculateArea()

Pour apprendre à implémenter une interface vous pouvez vous référer au diaporama présenté en formation

Vous pouvez également regarder la vidéo suivante :

<https://www.youtube.com/watch?v=OkEwPtRaqY4>

implémentez une classe "Rectangle" comprenant les attributs présentés sur le diagramme UML contenu dans le sous-dossier "doc"

Le diagramme UML est sous format "uxf". Pour pouvoir le lire il vous faudra le logiciel UMLET

UMLET est disponible à l'adresse suivante : <https://www.umlet.com/>

Nous avons extrait les informations du diagramme UML concernant les classes `Rectangle` et `Circle`. Nous pouvons maintenant compléter et vérifier les fichiers fournis et écrire le code Java pour ces classes.

Interface `Shape`

Tout d'abord, définissons une interface `Shape` avec les méthodes `calculatePerimeter` et `calculateArea` :

```
public interface Shape {  
    double calculatePerimeter();  
    double calculateArea();  
}
```

Classe `Rectangle`

Voici la classe `Rectangle` avec les méthodes `calculatePerimeter` et `calculateArea` :

```
public class Rectangle implements Shape {  
    private double length;  
    private double width;  
  
    public Rectangle(double length, double width) {  
        this.length = length;  
        this.width = width;  
    }  
}
```

Avec ces classes, nous avons implémenté les formes géométriques demandées avec les méthodes pour calculer le périmètre et l'aire.

TODO implémentez une classe "Circle" comme présentée par le diagramme UML

basons la classe `Circle` sur le diagramme UML. D'après les informations extraites, la classe `Circle` doit avoir un attribut `radius`.

Voici la classe `Circle` complète :

Classe `circle`

Voici la classe `Circle` implémentée selon le diagramme UML :

```
public class Circle implements Shape {
    private double radius;

    public Circle(double radius) {
        this.radius = radius;
    }
}
```

Explication

- **Attribut :**
 - `radius` (type `double`) : Rayon du cercle.
- **Constructeur :**
 - `public Circle(double radius)` : Initialise le cercle avec un rayon donné.

Cette classe utilise l'interface `Shape` et implémente les méthodes nécessaires pour calculer le périmètre et l'aire du cercle.

TODO créez une interface nommée "Shape" comprenant deux méthodes abstraites de calcul :

- double calculatePerimeter()
- double calculateArea()

Voici comment créer l'interface Shape avec les méthodes abstraites calculatePerimeter et calculateArea :

```
public interface Shape {  
    double calculatePerimeter();  
    double calculateArea();  
}
```

Explication

- **Interface Shape :**
 - Contient deux méthodes abstraites :
 - double calculatePerimeter() : Méthode pour calculer le périmètre.
 - double calculateArea() : Méthode pour calculer l'aire.

Cette interface sera implémentée par les classes représentant des formes géométriques telles que Rectangle, Circle et Triangle.

De même, vous pouvez implémenter d'autres formes géométriques en utilisant cette interface Shape.

TODO faites en sorte que la classe "Rectangle" implémente l'interface "Shape" et implémentez les deux méthodes

Rappel de calcul :

- $\text{perimetre_rectangle} = 2 * \text{longueur} + 2 * \text{largeur}$
- $\text{aire_rectangle} = \text{longueur} * \text{largeur}$

Pour faire en sorte que la classe Rectangle implémente l'interface Shape et implémenter les méthodes calculatePerimeter et calculateArea, voici le code complet :

Interface Shape

```
public interface Shape {  
    double calculatePerimeter();  
    double calculateArea();  
}
```

Classe Rectangle

```
public class Rectangle implements Shape {  
    private double length;  
    private double width;  
  
    public Rectangle(double length, double width) {  
        this.length = length;  
        this.width = width;  
    }  
  
    @Override  
    public double calculatePerimeter() {  
        return 2 * length + 2 * width;  
    }  
  
    @Override  
    public double calculateArea() {  
        return length * width;  
    }  
}
```

Explication

- **Attributs :**
 - length (type double) : La longueur du rectangle.
 - width (type double) : La largeur du rectangle.
- **Constructeur :**
 - public Rectangle(double length, double width) : Initialise le rectangle avec une longueur et une largeur données.
- **Méthodes :**
 - @Override public double calculatePerimeter() : Calcule et retourne le périmètre du rectangle en utilisant la formule $2 \times \text{longueur} + 2 \times \text{largeur}$

- `@Override public double calculateArea()` : Calcule et retourne l'aire du rectangle en utilisant la formule $\text{longueur} \times \text{largeur}$.

Avec cette implémentation, la classe `Rectangle` respecte l'interface `Shape` et fournit les calculs nécessaires pour le périmètre et l'aire.

TODO faites en sorte que la classe "Circle" implémente l'interface "Shape" et implémentez les deux méthodes

Rappel de calcul :

- périmètre du cercle = $2 * \pi * \text{rayon}$
- aire_cercle = $\pi * \text{rayon}^2$

La valeur π peut être retrouvée en Java en utilisant "Math.PI"

La puissance de 2 peut être effectuée en utilisant la méthode static "pow" de la classe "Math"

-> plus d'informations par ici <https://codegym.cc/fr/groups/posts/fr.575.math-pi-en-java>

Pour que la classe Circle implémente l'interface Shape et implémente les méthodes calculatePerimeter et calculateArea, voici comment procéder :

Interface Shape

```
public interface Shape {  
    double calculatePerimeter();  
    double calculateArea();  
}
```

Classe circle

```
public class Circle implements Shape {  
    private double radius;  
  
    public Circle(double radius) {  
        this.radius = radius;  
    }  
  
    @Override  
    public double calculatePerimeter() {  
        return 2 * Math.PI * radius;  
    }  
  
    @Override  
    public double calculateArea() {  
        return Math.PI * Math.pow(radius, 2);  
    }  
}
```

Explication

- **Attribut :**
 - o radius (type double) : Rayon du cercle.
- **Constructeur :**

- `public Circle(double radius)` : Initialise le cercle avec un rayon donné.

- **Méthodes :**

- `@Override public double calculatePerimeter()` : Calcule et retourne le périmètre du cercle en utilisant la formule $2 \times \pi \times \text{rayon}$.
`\text{rayon}`.
- `@Override public double calculateArea()` : Calcule et retourne l'aire du cercle en utilisant la formule $\pi \times \text{rayon}^2$, avec `Math.PI` pour la valeur de π et `Math.pow(radius, 2)` pour élever le rayon à la puissance de 2.

// TODO instancier plusieurs objets des classes Rectangle et Circle (2 instances de chaque)

Voici une classe `GeometricShapeMain` qui instancie plusieurs objets des classes `Rectangle` et `Circle` :

Code

```
public class GeometricShapeMain {
    public static void main(String[] args) {
        // Instancier des objets Rectangle
        Rectangle rectangle1 = new Rectangle(5, 10);
        Rectangle rectangle2 = new Rectangle(3, 7);

        // Instancier des objets Circle
        Circle circle1 = new Circle(4);
        Circle circle2 = new Circle(6);

        // Afficher les périmètres et les aires des rectangles
        System.out.println("Rectangle 1 - Perimeter: " +
            rectangle1.calculatePerimeter() + ", Area: " + rectangle1.calculateArea());
        System.out.println("Rectangle 2 - Perimeter: " +
            rectangle2.calculatePerimeter() + ", Area: " + rectangle2.calculateArea());

        // Afficher les périmètres et les aires des cercles
        System.out.println("Circle 1 - Perimeter: " +
            circle1.calculatePerimeter() + ", Area: " + circle1.calculateArea());
        System.out.println("Circle 2 - Perimeter: " +
            circle2.calculatePerimeter() + ", Area: " + circle2.calculateArea());
    }
}
```

Explication

1. Instanciation des objets `Rectangle` :

- o `Rectangle rectangle1 = new Rectangle(5, 10);` : Crée un rectangle avec une longueur de 5 et une largeur de 10.
- o `Rectangle rectangle2 = new Rectangle(3, 7);` : Crée un rectangle avec une longueur de 3 et une largeur de 7.

2. Instanciation des objets `Circle` :

- o `Circle circle1 = new Circle(4);` : Crée un cercle avec un rayon de 4.
- o `Circle circle2 = new Circle(6);` : Crée un cercle avec un rayon de 6.

3. Affichage des périmètres et des aires :

- o Utilisation des méthodes `calculatePerimeter` et `calculateArea` pour calculer et afficher les périmètres et les aires des rectangles et des cercles.

Les classes `Shape`, `Rectangle`, et `Circle` sont correctement définies dans le même package ou dans des packages accessibles pour que ce code fonctionne correctement.

Ces classes et le code de la méthode `main` devraient fonctionner ensemble pour créer et afficher les périmètres et les aires des formes géométriques.

Résultat :

Classe `Shape` :

```
package fr.afpa.shapes;

// Ajouter ICI l'interface "Shape"
// pour apprendre à déclarer une interface -> https://gayerie.dev/epsi-b3-
java/langage_java/interface.html
public interface Shape {
    public abstract double calculatePerimeter();
    public abstract double calculateArea();
}
```

Classe `Rectangle` :

```
package fr.afpa.shapes;

// Implémenter la classe Rectangle
// se référer au diagramme UML contenu dans le sous-dossier "doc" pour
retrouver les informations concernant ses attributs
public class Rectangle implements Shape {
    // Attributs :
    private double length;
    private double width;

    //Constructeurs :
    public Rectangle (double length, double width) {
        this.length = length;
        this.width = width;
    }

    // Getter
    public double getLength() {
        return length;
    }

    public double getWidth() {
        return width;
    }
}
```

```

// Setter
public void setLength(double length) {
    this.length = length;
}

public void setWidth(int width) {
    this.width = width;
}

@Override
public double calculatePerimeter() {
    // Modifier le code pour calculer le périmètre d'un rectangle
    // Rappel de calcul : perimetre_rectangle = 2 * longueur + 2 * largeur
    return 2 * (length + width);
}

@Override
public double calculateArea() {
    // Modifier le code pour calculer la surface d'un rectangle
    // Rappel de calcul : aire_rectangle = longueur * largeur
    return length * width;
}
}

```

Classe Circle :

```

package fr.afpa.shapes;

//import fr.afpa.shapes.Shape;

public class Circle implements Shape {
    // Attributs :
    private double radius;

    //Constructeurs :
    public Circle(double radius) {
        this.radius = radius;
    }

    // Getter
    public double getRadius() {
        return radius;
    }

    // Setter
    public void setRadius(double radius) {
        this.radius = radius;
    }
}

```



```

@Override
public double calculatePerimeter() {
    // Modifier le code pour calculer le périmètre d'un cercle
    // Rappel de calcul : périmètre du cercle = 2 *  $\pi$  * rayon
    return 2 * Math.PI * radius;
}

@Override
public double calculateArea() {
    // Modifier le code pour calculer la surface d'un cercle
    // Rappel de calcul : aire_cercle =  $\pi$  * rayon^2
    return Math.PI * radius * radius;
}
}

```

Classe GeometricShapeMain.java :

```

package fr.afpa.shapes;
/*
class GeometricShapeMain
{
    public static void main(String[] args)
    {
        // Instancier des objets de classe Rectangle
        Rectangle rectangle1 = new Rectangle(5, 10);
        Rectangle rectangle2 = new Rectangle(3, 7);

        // Instancier des objets de classe Circle
        Circle circle1 = new Circle(4);
        Circle circle2 = new Circle(6);

        // Afficher les périmètres et les aires des rectangles
        System.out.println("Rectangle 1 - Length: " + rectangle1.getLength() +
            ", Area: " + rectangle1.getWidth());
        System.out.println("Rectangle 1 - Perimeter: " +
            rectangle1.calculatePerimeter() + ", Area: " + rectangle1.calculateArea());
        System.out.println("Rectangle 2 - Length: " + rectangle2.getLength() +
            ", Area: " + rectangle2.getWidth());
        System.out.println("Rectangle 2 - Perimeter: " +
            rectangle2.calculatePerimeter() + ", Area: " + rectangle2.calculateArea());

        // Afficher les périmètres et les aires des cercles
        System.out.println("Circle 1 - Radius: " + circle1.getRadius());
    }
}

```

```
        System.out.println("Circle 1 - Perimeter: " +  
circle1.calculatePerimeter() + ", Area: " + circle1.calculateArea());  
        System.out.println("Circle 2 - Radius: " + circle2.getRadius());  
        System.out.println("Circle 2 - Perimeter: " +  
circle2.calculatePerimeter() + ", Area: " + circle2.calculateArea());  
    }  
}
```



```
PROBLÈMES 7 SORTIE CONSOLE DE DÉBOGAGE TERMINAL PORTS + v  
-implementation-d-interface\target\classes' 'fr.afpa.shapes.GeometricShapeMain'  
  
Rectangle 1 - Lenght: 5.0, Area: 10.0  
Rectangle 1 - Perimeter: 30.0, Area: 50.0  
Rectangle 2 - Lenght: 3.0, Area: 7.0  
Rectangle 2 - Perimeter: 20.0, Area: 21.0  
Circle 1 - Radius: 4.0  
Circle 1 - Perimeter: 25.132741228718345, Area: 50.26548245743669  
Circle 2 - Radius: 6.0  
Circle 2 - Perimeter: 37.69911184307752, Area: 113.09733552923255  
PS C:\Users\diego\Documents\Exos Java\Exo P00 Java\TP4 P00 Shapes l'implémentation d'interfaces\TP4-P00-Shapes-implémentation-d-interface\TP4-P00-Shapes-implémentation-d-interface>
```

```
// TODO ajouter ces objets à une instance de la classe "ArrayList"
// plus d'information sur la classe "ArrayList" -> https://info.clg.qc.ca/java/structures/array-list
```

Pour ajouter ces objets à une instance de la classe `ArrayList`, nous devons utiliser l'interface `List` de Java et ajouter les objets créés à cette liste. Voici comment faire :

Code Complet

Interface Shape

```
public interface Shape {
    double calculatePerimeter();
    double calculateArea();
}
```

Classe Rectangle

```
public class Rectangle implements Shape {
    private double length;
    private double width;

    public Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }

    @Override
    public double calculatePerimeter() {
        return 2 * length + 2 * width;
    }

    @Override
    public double calculateArea() {
        return length * width;
    }
}
```

Classe Circle

```
public class Circle implements Shape {
    private double radius;

    public Circle(double radius) {
        this.radius = radius;
    }

    @Override
    public double calculatePerimeter() {
        return 2 * Math.PI * radius;
    }

    @Override
    public double calculateArea() {
        return Math.PI * radius * radius;
    }
}
```

Classe GeometricShapeMain

```
import java.util.ArrayList;
import java.util.List;

public class GeometricShapeMain {
    public static void main(String[] args) {
        // Instancier des objets Rectangle
        Rectangle rectangle1 = new Rectangle(5, 10);
        Rectangle rectangle2 = new Rectangle(3, 7);

        // Instancier des objets Circle
        Circle circle1 = new Circle(4);
        Circle circle2 = new Circle(6);

        // Créer une liste pour stocker les formes géométriques
        List<Shape> shapes = new ArrayList<>();

        // Ajouter les objets à la liste
        shapes.add(rectangle1);
        shapes.add(rectangle2);
        shapes.add(circle1);
        shapes.add(circle2);

        // Afficher les périmètres et les aires des formes dans la liste
        for (Shape shape : shapes) {
            System.out.println("Shape - Perimeter: " +
                shape.calculatePerimeter() + ", Area: " + shape.calculateArea());
        }
    }
}
```

Explication

1. Importation des Classes Nécessaires :

- o import java.util.ArrayList;
- o import java.util.List;

2. Instanciation des Objets Rectangle et Circle :

- o Crée deux objets Rectangle et deux objets Circle comme dans l'exemple précédent.

3. Création d'une Liste pour Stocker les Formes Géométriques :

- o List<Shape> shapes = new ArrayList<>(); : Crée une liste de formes géométriques.

4. Ajout des Objets à la Liste :

- o Utilise la méthode add pour ajouter chaque objet Rectangle et Circle à la liste.

5. Affichage des Périmètres et des Aires des Formes dans la Liste :

- o Parcourt la liste avec une boucle for et affiche le périmètre et l'aire de chaque forme en utilisant les méthodes calculatePerimeter et calculateArea.

Ce code ajoute les objets créés à une instance de ArrayList et affiche leurs périmètres et aires. Assurez-vous que toutes les classes (Shape, Rectangle, Circle, et GeometricShapeMain) sont dans le même package ou dans des packages accessibles pour que ce code fonctionne correctement.

// TODO boucler sur les éléments de la liste afin d'afficher le périmètre et l'aire de chaque objet

Pour boucler sur les éléments de la liste et afficher le périmètre et l'aire de chaque objet, nous pouvons utiliser une boucle `for-each` pour itérer sur chaque élément de la liste et appeler les méthodes `calculatePerimeter` et `calculateArea`. Voici le code complet mis à jour :

Interface Shape

```
public interface Shape {  
    double calculatePerimeter();  
    double calculateArea();  
}
```

Classe Rectangle

```
public class Rectangle implements Shape {  
    private double length;  
    private double width;  
  
    public Rectangle(double length, double width) {  
        this.length = length;  
        this.width = width;  
    }  
  
    @Override  
    public double calculatePerimeter() {  
        return 2 * length + 2 * width;  
    }  
  
    @Override  
    public double calculateArea() {  
        return length * width;  
    }  
}
```

Classe circle

```
public class Circle implements Shape {  
    private double radius;  
  
    public Circle(double radius) {  
        this.radius = radius;  
    }  
  
    @Override  
    public double calculatePerimeter() {  
        return 2 * Math.PI * radius;  
    }  
  
    @Override  
    public double calculateArea() {  
        return Math.PI * radius * radius;  
    }  
}
```

```
}
```

Classe `GeometricShapeMain`

```
import java.util.ArrayList;
import java.util.List;

public class GeometricShapeMain {
    public static void main(String[] args) {
        // Instancier des objets Rectangle
        Rectangle rectangle1 = new Rectangle(5, 10);
        Rectangle rectangle2 = new Rectangle(3, 7);

        // Instancier des objets Circle
        Circle circle1 = new Circle(4);
        Circle circle2 = new Circle(6);

        // Créer une liste pour stocker les formes géométriques
        List<Shape> shapes = new ArrayList<>();

        // Ajouter les objets à la liste
        shapes.add(rectangle1);
        shapes.add(rectangle2);
        shapes.add(circle1);
        shapes.add(circle2);

        // Boucler sur les éléments de la liste et afficher le périmètre et
        // l'aire de chaque objet
        for (Shape shape : shapes) {
            System.out.println("Shape - Perimeter: " +
                shape.calculatePerimeter() + ", Area: " + shape.calculateArea());
        }
    }
}
```

Explication

- 1. Importation des Classes Nécessaires :**
 - o `import java.util.ArrayList;`
 - o `import java.util.List;`
- 2. Instanciation des Objets `Rectangle` et `Circle` :**
 - o Crée deux objets `Rectangle` et deux objets `Circle` comme dans l'exemple précédent.
- 3. Création d'une Liste pour Stocker les Formes Géométriques :**
 - o `List<Shape> shapes = new ArrayList<>();` : Crée une liste de formes géométriques.
- 4. Ajout des Objets à la Liste :**
 - o Utilise la méthode `add` pour ajouter chaque objet `Rectangle` et `Circle` à la liste.
- 5. Boucler sur les Éléments de la Liste et Afficher le Périmètre et l'Aire de Chaque Objet :**
 - o Utilise une boucle `for-each` pour parcourir chaque élément de la liste et appelle les méthodes `calculatePerimeter` et `calculateArea` pour afficher les valeurs.

Ce code itère sur les éléments de la liste `shapes` et affiche le périmètre et l'aire de chaque forme géométrique, fournissant une sortie claire et lisible pour chaque objet dans la console.

Autre méthode :

Pour implémenter les classes `Rectangle`, `Circle` et `Triangle` ainsi que l'interface `Shape` en suivant les consignes, nous allons procéder par étapes :

1. Créer l'interface `Shape`.
2. Implémenter la classe `Rectangle` pour qu'elle implémente l'interface `Shape`.
3. Implémenter la classe `Circle` pour qu'elle implémente l'interface `Shape`.
4. Adapter le programme principal pour tester les différentes formes géométriques.

Étape 1 : Créer l'interface `Shape`

```
// Shape.java

public interface Shape {
    double calculatePerimeter();
    double calculateArea();
}
```

Étape 2 : Implémenter la classe `Rectangle`

```
// Rectangle.java

public class Rectangle implements Shape {
    private double length;
    private double width;

    public Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }

    @Override
    public double calculatePerimeter() {
        return 2 * (length + width);
    }

    @Override
    public double calculateArea() {
        return length * width;
    }

    // Getters and Setters if needed
    public double getLength() {
        return length;
    }

    public void setLength(double length) {
        this.length = length;
    }

    public double getWidth() {
        return width;
    }

    public void setWidth(double width) {
        this.width = width;
    }
}
```

```
    }  
}
```

Étape 3 : Implémenter la classe `Circle`

```
// Circle.java  
  
public class Circle implements Shape {  
    private double radius;  
  
    public Circle(double radius) {  
        this.radius = radius;  
    }  
  
    @Override  
    public double calculatePerimeter() {  
        return 2 * Math.PI * radius;  
    }  
  
    @Override  
    public double calculateArea() {  
        return Math.PI * Math.pow(radius, 2);  
    }  
  
    // Getters and Setters if needed  
    public double getRadius() {  
        return radius;  
    }  
  
    public void setRadius(double radius) {  
        this.radius = radius;  
    }  
}
```

Étape 4 : Adapter le programme principal pour tester les différentes formes géométriques

```
// GeometricShapeMain.java  
  
public class GeometricShapeMain {  
    public static void main(String[] args) {  
        Shape rect = new Rectangle(5, 10);  
        System.out.println("Rectangle Area: " + rect.calculateArea());  
        System.out.println("Rectangle Perimeter: " +  
rect.calculatePerimeter());  
  
        Shape circle = new Circle(7);  
        System.out.println("Circle Area: " + circle.calculateArea());  
        System.out.println("Circle Perimeter: " +  
circle.calculatePerimeter());  
  
        Shape triangle = new Triangle(3, 4, 5);  
        System.out.println("Triangle Area: " + triangle.calculateArea());  
        System.out.println("Triangle Perimeter: " +  
triangle.calculatePerimeter());  
    }  
}
```

Avec ces implémentations, nous avons les classes `Rectangle`, `Circle` et `Triangle` qui implémentent l'interface `Shape`. Le programme principal crée des instances de chaque forme et affiche leur aire et périmètre.

Voulez-vous que je modifie ou ajoute d'autres éléments spécifiques dans ces fichiers ?

Resultat final :

Classe GeometricShapeMain.java :

```
package fr.afpa.shapes;

import java.util.ArrayList;
import java.util.List;

/*
    Objectif : développer des classes représentant des formes géométriques
    (Rectangle, Cercle et Triangle)
    Pour chacune de ces classes il faudra implémenter deux méthodes :
    - une qui renverra le périmètre -> double calculatePerimeter()
    - une qui renverra l'aire -> double calculateArea()

    Pour apprendre à implémenter une interface vous pouvez vous référer au
    diaporama présenté en formation
    Vous pouvez également regarder la vidéo suivante :
    https://www.youtube.com/watch?v=OkEwPtRaqY4

    TODO implémentez une classe "Rectangle" comprenant les attributs présentés
    sur le diagramme UML contenu dans le sous-dossier "doc"
    Le diagramme UML est sous format "uxf". Pour pouvoir le lire il vous
    faudra le logiciel UMLET
    UMLET est disponible à l'adresse suivante : https://www.umlet.com/

    TODO implémentez une classe "Circle" comme présentée par le diagramme UML

    TODO créez une interface nommée "Shape" comprenant deux méthodes
    abstraites de calcul :
    - double calculatePerimeter()
    - double calculateArea()

    TODO faites en sorte que la classe "Rectangle" implémente l'interface
    "Shape" et implémentez les deux méthodes
    Rappel de calcul :
    - perimetre_rectangle = 2 * longueur + 2 * largeur
    - aire_rectangle = longueur * largeur

    TODO faites en sorte que la classe "Circle" implémente l'interface "Shape"
    et implémentez les deux méthodes
    Rappel de calcul :
    - périmètre du cercle = 2 *  $\pi$  * rayon
    - aire_cercle =  $\pi$  * rayon^2

    La valeur  $\pi$  peut être retrouvée en Java en utilisant "Math.PI"
    La puissance de 2 peut être effectuée en utilisant la méthode static
    "pow" de la classe "Math"

```

```

    -> plus d'informations par ici
https://codegym.cc/fr/groups/posts/fr.575.math-pi-en-java

*/
public class GeometricShapeMain
{
    public static void main(String[] args)
    {
        // Instancier des objets de classe Rectangle
        Rectangle rectangle1 = new Rectangle(5, 10);
        Rectangle rectangle2 = new Rectangle(3, 7);

        // Instancier des objets de classe Circle
        Circle circle1 = new Circle(4);
        Circle circle2 = new Circle(6);

        System.out.println("\n----- Afficher les périmètres et les aires des
rectangles\n" + "-----");
        // Afficher les périmètres et les aires des rectangles
        System.out.println("Rectangle 1 - Length: " + rectangle1.getLength() +
", Area: " + rectangle1.getWidth());
        System.out.println("Rectangle 1 - Perimeter: " +
rectangle1.calculatePerimeter() + ", Area: " + rectangle1.calculateArea());
        System.out.println("Rectangle 2 - Length: " + rectangle2.getLength() +
", Area: " + rectangle2.getWidth());
        System.out.println("Rectangle 2 - Perimeter: " +
rectangle2.calculatePerimeter() + ", Area: " + rectangle2.calculateArea());

        System.out.println("\n----- Afficher les périmètres et les aires des
cercles\n" + "-----");
        // Afficher les périmètres et les aires des cercles
        System.out.println("Circle 1 - Radius: " + circle1.getRadius());
        System.out.println("Circle 1 - Perimeter: " +
circle1.calculatePerimeter() + ", Area: " + circle1.calculateArea());
        System.out.println("Circle 2 - Radius: " + circle2.getRadius());
        System.out.println("Circle 2 - Perimeter: " +
circle2.calculatePerimeter() + ", Area: " + circle2.calculateArea());

        System.out.println("\n----- Ajouter ces objets à une instance de la
classe \ArrayList\n" + "-----");
        // Ajouter ces objets à une instance de la classe "ArrayList"
        // plus d'information sur la classe "ArrayList" ->
https://info.clg.qc.ca/java/structures/array-list
        // Créer une liste pour stocker les formes géométriques
        List<Shape> shapes = new ArrayList<>();

        // Ajouter les objets à la liste
        // shapes.add(rectangle1.getLength(), rectangle1.getWidth());

```

```

        shapes.add(rectangle1);
        //shapes.add(rectangle2.getLength(), rectangle2.getWidth());
        shapes.add(rectangle2);
        // shapes.add(circle1.getRadius());
        shapes.add(circle1);
        // shapes.add(circle2.getRadius());
        shapes.add(circle2);

        System.out.println("\n----- Boucler sur les éléments de la liste afin
d'afficher le périmètre et l'aire de chaque objet\n -----");
        // Boucler sur les éléments de la liste afin d'afficher le périmètre
et l'aire de chaque objet
        // Afficher les périmètres et les aires des formes dans la liste
        for (Shape shape : shapes) {
            System.out.println("Shape : " + shape.toString() + " - Perimeter: " +
shape.calculatePerimeter() + ", Area: " + shape.calculateArea());
        }
    }
}

```

Classe Shape.java :

```

package fr.afpa.shapes;

// Ajouter ICI l'interface "Shape"
// pour apprendre à déclarer une interface -> https://gayerie.dev/epsi-b3-java/langage\_java/interface.html
public interface Shape {
    public abstract double calculatePerimeter();
    public abstract double calculateArea();
}

```

Classe Circle.java :

```

package fr.afpa.shapes;

//import fr.afpa.shapes.Shape;

public class Circle implements Shape {
    // Attributs :
    private double radius;

    //Constructeurs :
    public Circle(double radius) {
        this.radius = radius;
    }

    // Getter
    public double getRadius() {
        return radius;
    }
}

```

```

    }

    // Setter
    public void setRadius(double radius) {
        this.radius = radius;
    }

    @Override
    public double calculatePerimeter() {
        // Modifier le code pour calculer le périmètre d'un cercle
        // Rappel de calcul : périmètre du cercle = 2 *  $\pi$  * rayon
        return 2 * Math.PI * radius;
    }

    @Override
    public double calculateArea() {
        // Modifier le code pour calculer la surface d'un cercle
        // Rappel de calcul : aire_cercle =  $\pi$  * rayon^2
        return Math.PI * radius * radius;
    }

    @Override
    public String toString() {
        return "Circle [radius=" + radius + "]";
    }
}

```

Classe Rectangle.java :

```

package fr.afpa.shapes;

// Implémenter la classe Rectangle
// se référer au diagramme UML contenu dans le sous-dossier "doc" pour
retrouver les informations concernant ses attributs
public class Rectangle implements Shape {
    // Attributs :
    private double length;
    private double width;

    //Constructeurs :
    public Rectangle (double length, double width) {
        this.length = length;
        this.width = width;
    }

    // Getter

```

```
public double getLength() {
    return length;
}

public double getWidth() {
    return width;
}

// Setter
public void setLength(double length) {
    this.length = length;
}

public void setWidth(int width) {
    this.width = width;
}

@Override
public double calculatePerimeter() {
    // Modifier le code pour calculer le périmètre d'un rectangle
    // Rappel de calcul : perimetre_rectangle = 2 * longueur + 2 * largeur
    return 2 * (length + width);
}

@Override
public double calculateArea() {
    // Modifier le code pour calculer la surface d'un rectangle
    // Rappel de calcul : aire_rectangle = longueur * largeur
    return length * width;
}

@Override
public String toString() {
    return "Rectangle [length=" + length + ", width=" + width + "]";
}

}
```



```
PROBLÈMES (5)  SORTIE  CONSOLE DE DÉBOGAGE  TERMINAL  PORTS  + v
PS C:\Users\diego\Documents\Exos Java\Exo P00 Java\TP4 P00 Shapes l'implémentation d'interfaces\TP4-P00-Shapes-im
plementation-d-interface\TP4-P00-Shapes-implementation-d-interface> c;; cd 'c:\Users\diego\Documents\Exos Java\Exo P00 Java\TP4 P00 Shapes l'implémentation d'interfaces\TP4-P00-Shapes-implementation-d-interface\TP4-P00-Shapes-implementation-d-interface'; & 'C:\Users\diego\vscode\extensions\redhat.java-1.32.0-win32-x64\jre\17.0.11-win32-x86_64\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\diego\Documents\Exos Java\Exo P00 Java\TP4 P00 Shapes l'implémentation d'interfaces\TP4-P00-Shapes-implementation-d-interface\TP4-P00-Shapes-implementation-d-interface\target\classes' 'fr.afpa.shapes.GeometricShapeMain'

----- Afficher les périmètres et les aires des rectangles" -----
Rectangle 1 - Lenght: 5.0, Area: 10.0
Rectangle 1 - Perimeter: 30.0, Area: 50.0
Rectangle 2 - Perimeter: 20.0, Area: 21.0

----- Afficher les périmètres et les aires des cercles" -----
Circle 1 - Radius: 4.0
Circle 1 - Perimeter: 25.132741228718345, Area: 50.26548245743669
Circle 2 - Radius: 6.0
Circle 2 - Perimeter: 37.69911184307752, Area: 113.09733552923255

----- Ajouter ces objets à une instance de la classe "ArrayList" -----

----- Boucler sur les éléments de la liste afin d'afficher le périmètre et l'aire de chaque objet" -----
Shape : Rectangle [length=5.0, width=10.0] - Perimeter: 30.0, Area: 50.0
Shape : Rectangle [length=3.0, width=7.0] - Perimeter: 20.0, Area: 21.0
Shape : Circle [radius=4.0] - Perimeter: 25.132741228718345, Area: 50.26548245743669
Shape : Circle [radius=6.0] - Perimeter: 37.69911184307752, Area: 113.09733552923255
PS C:\Users\diego\Documents\Exos Java\Exo P00 Java\TP4 P00 Shapes l'implémentation d'interfaces\TP4-P00-Shapes-im
plementation-d-interface\TP4-P00-Shapes-implementation-d-interface>
```