Exercice 1 : Définir une Classe Abstraite et Implémenter des Sous-classes

1. **Définissez une classe abstraite `Shape` avec les méthodes suivantes :**
   - Une méthode abstraite `double getArea();`
   - Une méthode abstraite `double getPerimeter();`
   - Une méthode concrète `void display();` qui affiche "This is a shape".

---

2. **Créez une classe `Circle` qui étend la classe abstraite `Shape`. Dans cette classe :**
   - Ajoutez un champ `radius`.
   - Implémentez les méthodes `getArea()` et `getPerimeter()`. Utilisez la formule (Area = PI ×radius^2) et (Perimeter = 2 ×PI ×radius).
   - Surchargez la méthode `display()` pour afficher "This is a circle".

---

3. **Créez une classe `Rectangle` qui étend la classe abstraite `Shape`. Dans cette classe :**
   - Ajoutez les champs `width` et `height`.
   - Implémentez les méthodes `getArea()` et `getPerimeter()`. Utilisez la formule (Area = width ×height) et (Perimeter = 2 ×(width + height)).
   - Surchargez la méthode `display()` pour afficher "This is a rectangle".

---

4. **Créez une classe principale `Main` avec une méthode `main` pour tester vos classes `Circle` et `Rectangle`.**

Exercice 2 : Classe Abstraite avec des Méthodes Abstraites et Concrètes

1. **Définissez une classe abstraite `Employee` avec les méthodes suivantes :**
   - Une méthode abstraite `double calculateSalary();`

- Une méthode concrète `void displayDetails();` qui affiche les détails de l'employé.
- Ajoutez des champs `name` et `id`.

---

2. **Créez une classe `FullTimeEmployee` qui étend la classe abstraite `Employee`. Dans cette classe :**
   - Ajoutez un champ `salary`.
   - Implémentez la méthode `calculateSalary()` pour retourner le salaire.
   - Surchargez la méthode `displayDetails()` pour afficher les détails de l'employé à temps plein.

---

3. **Créez une classe `PartTimeEmployee` qui étend la classe abstraite `Employee`. Dans cette classe :**
   - Ajoutez des champs `hourlyRate` et `hoursWorked`.
   - Implémentez la méthode `calculateSalary()` pour retourner le produit de `hourlyRate` et `hoursWorked`.
   - Surchargez la méthode `displayDetails()` pour afficher les détails de l'employé à temps partiel.

---

4. **Créez une classe principale `Main` avec une méthode `main` pour tester vos classes `FullTimeEmployee` et `PartTimeEmployee`.**

Exercice 3 : Héritage Multiple via Interfaces et Classes Abstraites

1. **Définissez une interface `Drivable` avec les méthodes suivantes :**
   - `void drive();`
   - `void stop();`

---

2. **Définissez une classe abstraite `Vehicle` avec les champs et méthodes suivants :**
   - Un champ `make`.
   - Une méthode abstraite `void start();`
   - Une méthode concrète `void displayMake();` qui affiche la marque du véhicule.

---

3. **Créez une classe `Car` qui étend la classe abstraite `Vehicle` et implémente l'interface `Drivable`. Dans cette classe :**
   - Implémentez les méthodes `start()`, `drive()`, et `stop()` avec des affichages appropriés.

---

4. **Créez une classe principale `Main` avec une méthode `main` pour tester votre classe `Car`.**