PROGRAMMATION ORIENTÉE OBJET

1. Création de classe simple

Exercice : Classe Personne

1. Créez une classe Personne avec les propriétés nom et age, initialisées via le constructeur.
2. Ajoutez une méthode sePresenter qui affiche : "Bonjour, je m'appelle [nom] et j'ai [age] ans.".
3. Créez une instance de Personne et appelez la méthode sePresenter.

Résultat attendu :

```
const alice = new Personne("Alice", 30);
alice.sePresenter(); // Bonjour, je m'appelle Alice et j'ai 30 ans.
```

2. Encapsulation

Exercice : Classe CompteBancaire

1. Créez une classe CompteBancaire avec un champ privé #solde initialisé à 0.
2. Ajoutez un **getter** pour consulter le solde.
3. Ajoutez deux **setters** :
   - deposer(montant) : Permet de déposer un montant positif.
   - retirer(montant) : Permet de retirer un montant si le solde est suffisant.
4. Créez une instance et effectuez plusieurs opérations sur le compte.

Résultat attendu :

```
const compte = new CompteBancaire();
compte.deposer(500);
console.log(compte.solde); // 500
compte.retirer(200);
console.log(compte.solde); // 300
compte.retirer(400); // Solde insuffisant !
console.log(compte.solde); // 300
```

3. Héritage

Exercice : Classe Animal et sous-classes

1. Créez une classe Animal avec une propriété nom et une méthode parler qui affiche : "Cet animal fait un bruit.".
2. Créez une classe Chien qui hérite de Animal.
   - Redéfinissez la méthode parler pour afficher : "Le chien [nom] aboie.".
3. Créez une classe Chat qui hérite également de Animal.
   - Redéfinissez la méthode parler pour afficher : "Le chat [nom] miaule.".

4. Créez des instances de Chien et Chat et appelez leur méthode parler.

Résultat attendu :

```
const rex = new Chien("Rex");
rex.parler(); // Le chien Rex aboie.

const minou = new Chat("Minou");
minou.parler(); // Le chat Minou miaule.
```

4. Polymorphisme

Exercice : Liste d'animaux

1. Reprenez les classes Animal, Chien, et Chat.
2. Créez un tableau animaux contenant plusieurs instances de Chien, Chat, et Animal.
3. Parcourez le tableau avec une boucle et appelez la méthode parler sur chaque élément.

Résultat attendu :

```
const animaux = [new Chien("Rex"), new Chat("Minou"), new Animal("Un animal")];
animaux.forEach(animal => animal.parler());
// Le chien Rex aboie.
// Le chat Minou miaule.
// Cet animal fait un bruit.
```

5. Exercices combinés

Exercice : Classe Employe et Manager

1. Créez une classe Employe avec les propriétés nom, poste, et salaire.
   - Ajoutez une méthode afficherDetails qui affiche : "Nom : [nom], Poste : [poste], Salaire : [salaire]".
2. Créez une classe Manager qui hérite de Employe.
   - Ajoutez une propriété supplémentaire equipe (tableau de noms des employés sous sa supervision).
   - Redéfinissez la méthode afficherDetails pour inclure les noms de l'équipe.
3. Créez une instance de Manager et testez.

Résultat attendu :

```
const manager = new Manager("Alice", "Directrice", 8000, ["Bob", "Charlie"]);
manager.afficherDetails();
// Nom : Alice, Poste : Directrice, Salaire : 8000
// Equipe : Bob, Charlie
```

6. Application pratique

Exercice : Gestion d'une bibliothèque

1. Créez une classe Livre avec les propriétés titre, auteur, et pages.
    - Ajoutez une méthode afficherInfo pour afficher les détails du livre.
2. Créez une classe Bibliotheque avec une propriété privée #livres (tableau).
    - Ajoutez une méthode ajouterLivre pour ajouter un livre à la bibliothèque.
    - Ajoutez une méthode afficherLivres pour afficher les détails de tous les livres.
3. Ajoutez plusieurs livres à la bibliothèque et affichez les détails.

Résultat attendu :

```
const bibliotheque = new Bibliotheque();
bibliotheque.ajouterLivre(new Livre("1984", "George Orwell", 328));
bibliotheque.ajouterLivre(new Livre("Le Petit Prince", "Antoine de Saint-Exupéry", 96));
bibliotheque.afficherLivres();
// Titre : 1984, Auteur : George Orwell, Pages : 328
// Titre : Le Petit Prince, Auteur : Antoine de Saint-Exupéry, Pages : 96
```