JavaScript Orienté Objet

1. Encapsulation : La propriété privée #carburant de Vehicule empêche l'accès direct et est modifiée seulement via la méthode rechargerCarburant .

2. Héritage : Les classes Voiture et Moto héritent de Vehicule, partageant ses propriétés et méthodes.

3. Polymorphisme : La méthode demarrer est redéfinie dans chaque sous-classe, montrant le polymorphisme.

Création d'une Classe de Base Vehicule

Nous commençons par définir une classe de base Vehicule avec des propriétés publiques pour le type et la vitesse, et une propriété privée pour le niveau de carburant.

```
class Vehicule {
 #carburant;  // Propriété privée pour encapsuler le niveau de carburant

 constructor(type, vitesseMax) {
  this.type = type;
  this.vitesseMax = vitesseMax;
  this.#carburant = 100;  // Carburant initial
 }

 // Méthode pour afficher les informations du véhicule
 afficherInfo() {
  console.log(`Type: ${this.type}, Vitesse Max: ${this.vitesseMax} km/h, Carburant: ${this.#carburant}%`);
 }

 // Méthode pour recharger le carburant
 rechargerCarburant(quantite) {
```

```
    this.#carburant = Math.min(100, this.#carburant + quantite);
    console.log(` Carburant rechargé à ${this.#carburant}%`);
  }

  // Méthode pour démarrer le véhicule
  demarrer() {
    if (this.#carburant > 0) {
      console.log(` ${this.type} démarre.`);
    } else {
      console.log(` ${this.type} ne peut pas démarrer car le carburant est épuisé.`);
    }
  }
}

const vehicule = new Vehicule('Véhicule générique', 120);
vehicule.afficherInfo();
vehicule.demarrer();
```

Héritage – Création de Sous-Classes

Créons des sous-classes Voiture et Moto qui héritent de Vehicule. Chaque sous-classe pourra avoir un comportement spécifique pour la méthode demarrer.

```
class Voiture extends Vehicule {
  demarrer() {
    console.log(` La voiture ${this.type} démarre en douceur.`);
  }
}
```

```
class Moto extends Vehicule {
  demarrer() {
    console.log(` La moto ${this.type} démarre avec puissance. `);
  }
}

const voiture = new Voiture('Voiture', 180);
const moto = new Moto('Moto', 200);
voiture.demarrer();  // Affiche: "La voiture Voiture démarre en douceur."
moto.demarrer();     // Affiche: "La moto Moto démarre avec puissance."
```

Polymorphisme

– Utilisation d'une Méthode avec des Comportements Différents

Chaque sous-classe redéfinit la méthode demarrer, ce qui montre le polymorphisme, car cette méthode fonctionne différemment en fonction du type de véhicule.

Utilisation Complète avec Encapsulation, Héritage et Polymorphisme

Voici le code complet :

```
class Vehicule {
  #carburant; // Propriété privée

  constructor(type, vitesseMax) {
    this.type = type;
    this.vitesseMax = vitesseMax;
    this.#carburant = 100;
  }
```

```
  afficherInfo() {
    console.log(` Type: ${this.type}, Vitesse Max: ${this.vitesseMax} km/h, Carburant: ${this.#carburant}% `);
  }

  rechargerCarburant(quantite) {
    this.#carburant = Math.min(100, this.#carburant + quantite);
    console.log(` Carburant rechargé à ${this.#carburant}% `);
  }

  demarrer() {
    if (this.#carburant > 0) {
      console.log(` ${this.type} démarre. `);
    } else {
      console.log(` ${this.type} ne peut pas démarrer car le carburant est épuisé. `);
    }
  }
}

class Voiture extends Vehicule {
  demarrer() {
    console.log(` La voiture ${this.type} démarre en douceur. `);
  }
}

class Moto extends Vehicule {
  demarrer() {
    console.log(` La moto ${this.type} démarre avec puissance. `);
```

```
  }
}

// Utilisation
const vehiculeGenerique = new Vehicule('Véhicule générique', 120);
const voiture = new Voiture('Voiture de sport', 200);
const moto = new Moto('Moto de course', 220);

const vehicules = [vehiculeGenerique, voiture, moto];

// Affichage des informations et démarrage pour chaque véhicule
vehicules.forEach(vehicule => {
  vehicule.afficherInfo();
  vehicule.demarrer();
});
```