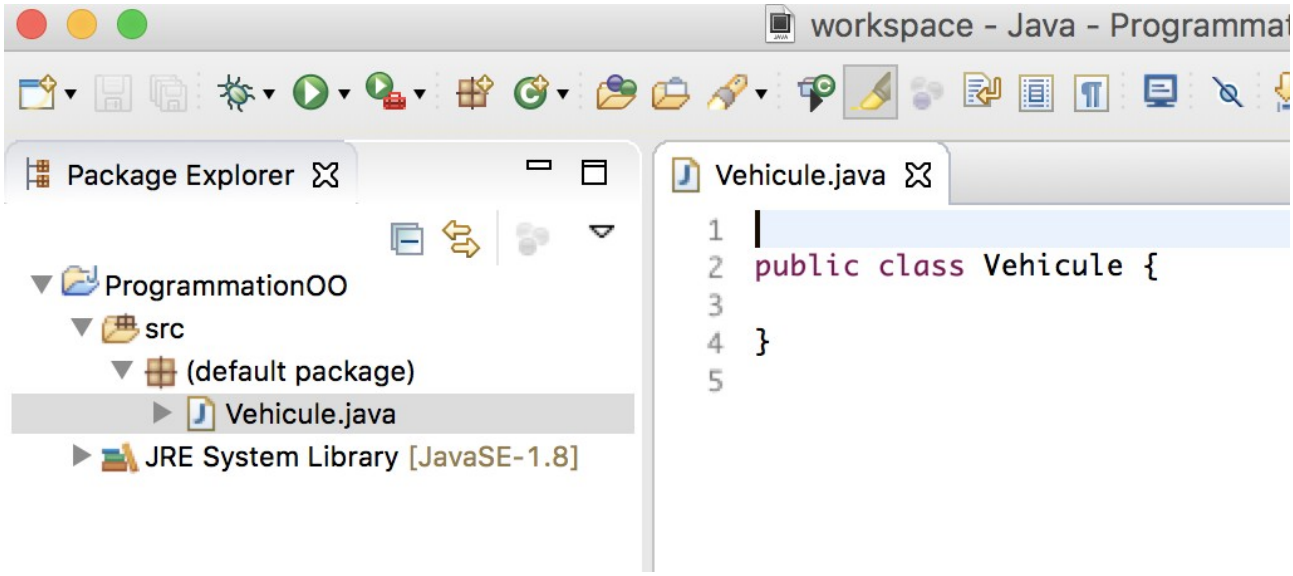
Pas à pas Démonstration – Classes, objets et encapsulation

Pour illustrer les principes de Programmation orientée objet,
Créez dans Eclipse un nouveau projet Java dans Eclipse

On va y définir un certain nombre de classes pour illustrer tous les principes de programmation orientée objet.

On va commencer par la définition d'une classe Vehicule, dans laquelle on va définir quelques attributs et des méthodes.
J'ai donc ici une structure de classe extrêmement basique la classe Vehicule définie dans le fichier Vehicule.java
01'00



Pour respecter les principes d'encapsulation.
les attributs par défaut vont être définis comme étant privés et les méthodes par défaut seront publiques.
C'est des choix sur lesquels on peut revenir plus tard si nécessaire.
Commencez par définir un attribut de type entier qui sera la vitesse du véhicule.
Définir un deuxième attribut, vitesse maximum, également de type entier.

```
public class Vehicule {
	private int vitesse;
	private int vitesseMaximum;
}
```

Ajoutez des méthodes , accélérer() et freiner() qui ne renvoient aucune valeur et ne prendront aucun paramètre.
Etant public , ces méthodes seront accessibles d'un point de vue des objets externes qui vont créer des objets Vehicule et qui vont les manipuler.

```
public class Vehicule {
	private int vitesse;
	private int vitesseMaximum;

	public void accelerer(){ }
	public void freiner{ }
}
```

Implémentation d"accélérer() : cela consiste à incrémenter la vitesse de 1 dans le corps de accelerer().
L'encapsulation va permettre de veiller à certaines règles concernant les modification des attributs comme par exemple veiller à ne pas pouvoir dépasser la vitesse maximale. On va coder un test en java me disant "si la vitesse est inférieure à la vitesseMaximum , alors je pourrais envisager d'augmenter cette vitesse .

```
public class Vehicule {
	private int vitesse;
	private int vitesseMaximum;

	public void accelerer(){
		if(vitesse < vitesseMaximum){
			vitesse++;
		}
	}

	public void freiner(){ }
}
```

Pour la méthode freiner(), je vais éviter d'obtenir une vitesse négative.

```
public class Vehicule {


	private int vitesse;

	private int vitesseMaximum;


	public void accelerer(){

		if(vitesse < vitesseMaximum){

			vitesse++;

		}

	}


	public void freiner(){

		if(vitesse>0)vitesse--;

	}

}
```

C'est l'encapsulation qui permet de garantir des valeur cohérentes.

Une fois cette classe créee on va pouvoir envisager de créer une classe principale, pour utiliser Vehicule.

Dans le répertoire de code source , je crée une nouvelle classe ,nommée Principale,contenant un main.
Je vais y instancier la classe Vehicule , créer des objets de type Vehicule.

Je déclare une variable pour référencer un objet de type Vehicule et ensuite je l'associe à une nouvelle instance de Vehicule. Je n'ai aucun moyen de fournir de valeur particulière d'initialisation à ma classe au moment de la création de mon objet car j'utilise le constructeur par défaut de la classe Vehicule (le constructeur implicite)

Mais je vais pouvoir invoquer les méthodes accélerer et freiner de mon véhicule.

```
public class Principale {

	public static void main(String[] args) {

		Vehicule veh=new Vehicule();

		veh.accelerer();

		veh.freiner();

	}

}
```

Quand je crée mon véhicule je veux pouvoir transmettre des valeurs de vitesse et de valeurMaximum.
Pour cela dans la classe véhicule je définisse un constructeur qui transmette deux valeurs entières, l'une pour la vitesse , l'autre pour la vitesseMaximum.

Je vais définir un constructeur, c'est à dire une méthode publique qui n'a pas de type de retour, qui porte le nom de la classe.
Je prévois deux paramètre entiers nommés v et vm . L'objectif sera ici de transmettre la valeur de v dans l'attribut vitesse, et de transmettre la valeur de vm dans l'attribut vitesseMaximum.

Donc en écrivant :

```
public Vehicule(int v, int vm){

		vitesse = v;

		vitesseMaximum = vm;

	}
```

Je ne pourrais faire ces initialisation d'attribut que si la vitesse < vitesseMaximum.

```
public class Vehicule {
	private int vitesse;
	private int vitesseMaximum;

	public Vehicule(int v, int vm){
		if(vitesse<vitesseMaximum){
			vitesse= v;
			vitesseMaximum = vm;
		}
	}

	public void accelerer(){
		if(vitesse < vitesseMaximum){
			vitesse++;
		}
	}

	public void freiner(){
		if(vitesse>0)vitesse--;
	}
}
```

=> attention l'opération que l'on vient de réaliser génère une erreur dans la classe principale.

Car le constructeur ( ) n'est pas défini.

En effet à partir du moment où on défini un constructeur explicitement pour une classe , le constructeur par défaut est masqué par ce constructeur.

```
public class Principale {


	public static void main(String[] args) {
		Vehicule veh=new Vehicule(50, 200);
		veh.accelerer();
		veh.freiner();
		//veh.vitesse;
	}
}
```

J'apporte la correction necessaire en fournissant au constructeur utilisé les paramètres entiers prévus par la classe Vehicule.
Mon compilateur instanciait un Vehicule sans paramètres jusqu'à présent mais le fait de coder un constructeur à paramètres dans Vehicule m'empêche désormais d'oublier les paramètres prévu par ce Constructeur explicite.
Cela s'explique par le fait que le constructeur par défaut (sans paramètre) existe implicitement tant qu'on ne rajoute pas d'autre constructeur explicitement. Si on a besoin de construire des Vehicuels sans paramètre il faut "re-coder" un constructeur sans paramètre.

```
public class Vehicule {
	private int vitesse;
	private int vitesseMaximum;
	//Constructeur
	public Vehicule () { }
	public Vehicule(int v, int vm){
		if(vitesse<vitesseMaximum){
			vitesse= v;
			vitesseMaximum = vm;
		}
```

```
	}
	public void accelerer(){
		if(vitesse < vitesseMaximum){
			vitesse++;
		}
	}
	public void freiner(){
		if(vitesse>0)vitesse--;
	}
}
```

Fin