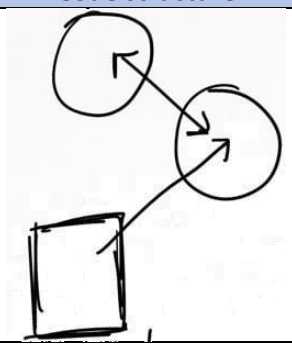
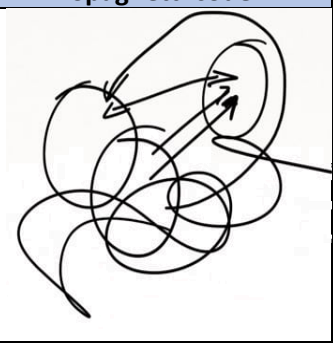
1. Les raisons qui amènent à structurer le code

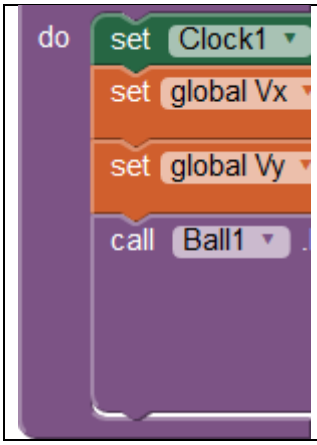
Structurer le code est important pour :

- Gagner en lisibilité
- Faciliter le travail en équipe
- Eviter les erreurs lorsque le programme est long et complexe

Code structuré	« Spaghetti code »
	

2. Les séquences d'instructions

Une Séquence (do) est une suite d'instructions (affectation de Variable, appel de Fonction, …) exécutées séquentiellement, c'est à dire l'une après l'autre.




```
digitalWrite(LED_BUILTIN, HIGH);
delay(1000);
digitalWrite(LED_BUILTIN, LOW);
delay(1000);
```

Cette organisation du code est suffisante lorsque le programme est court.

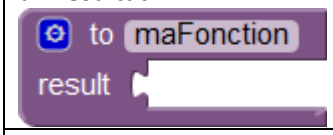
Lorsque les choses deviennent plus compliquées, il faut pouvoir regrouper le code pour en faciliter la lecture, la recherche des erreurs, les modifications et faire apparaitre clairement la structure du code principal.

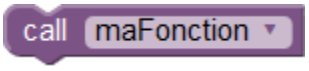
3. Les fonctions

Une fonction permet de regrouper sous un seul nom une Séquence de plusieurs instructions.

Une fonction doit d'abord être déclarée (to). Ensuite elle est appelée (call).

Son appel peut provoquer l'exécution d'une Séquence (do) ou bien renvoyer un résultat (result).

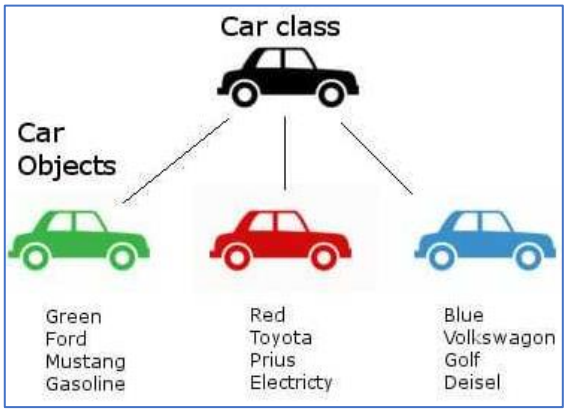
Déclaration :
 
 	
Ou, lorsque la fonction retourne un résultat :
 
 	
	Déclaration :
	`void DashedLine()`
	`{`
	`  Serial.println("----------------");`
	`}`
	Function is created here

Appel :
 
 	
	Appel :
	`DashedLine();` ← Function is called here
	`Serial.println("| Program Menu |");`
	`DashedLine();` ← Function is called again
	Résultat :
	`----------------`
	`| Program Menu |`
	`----------------`

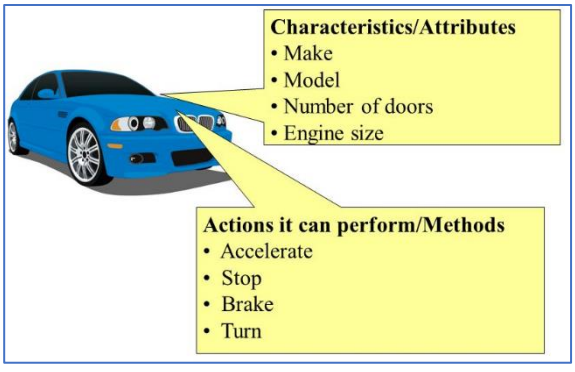
4. La programmation orientée objet (OOP : Object Oriented Programming)

Un objet est un élément numérique complexe, tel un bouton, une balle, un servomoteur, une image, …

Un objet est une instance de classe. Une classe est, en quelques sortes, le moule de fabrication de l'objet.



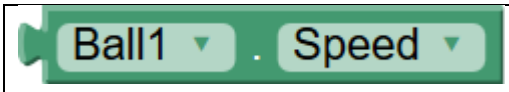
Un objet possède des Propriétés (=caractéristiques = attributs) et des Méthodes.



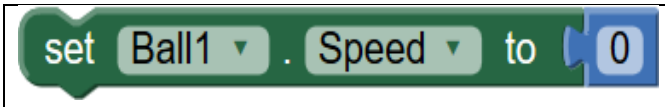
Propriété

Une propriété est une Variable qui caractérise un Objet (taille, couleur, vitesse, visibilité, …)

On peut lire une propriété de l'objet :

	`monServo.read()` avec « monServo » instance de la classe Servo

On peut écrire la valeur voulue pour la propriété :

	`monServo.attach(pin)` avec « monServo » instance de la classe Servo

Méthode

C'est une Fonction propre à un Objet :

	`monServo.write(angle)` ***Parameters*** *servo*: a variable of type Servo *angle*: the value to write to the servo, from 0 to 180

Contrairement à une Fonction (définie par l'utilisateur), une Méthode elle est déjà définie par le constructeur de la classe.

Les 4 principes de la programmation orientée objet sont :

- l'encapsulation,
- l'abstraction,
- l'héritage
- le polymorphisme

Encapsulation	Abstraction	Inheritance	Polymorphism
When an object only exposes the selected information.	Hides complex details to reduce complexity.	Entities can inherit attributes from other entities.	Entities can have more than one form.

5. Variables locales et variables globales

Une variable est un conteneur d'information (nombre, texte, …) doté d'un nom permettant d'accéder à cette information.

Elle peut être :

- une Propriété d'un Objet
- une variable locale (qui n'existe qu'à l'intérieur d'une Fonction)
- une variable globale (qui existe dans toute l'application)

Les variables doivent être crées, en précisant leur type, et initialisées. Elles peuvent ensuite être utilisées dans le code.