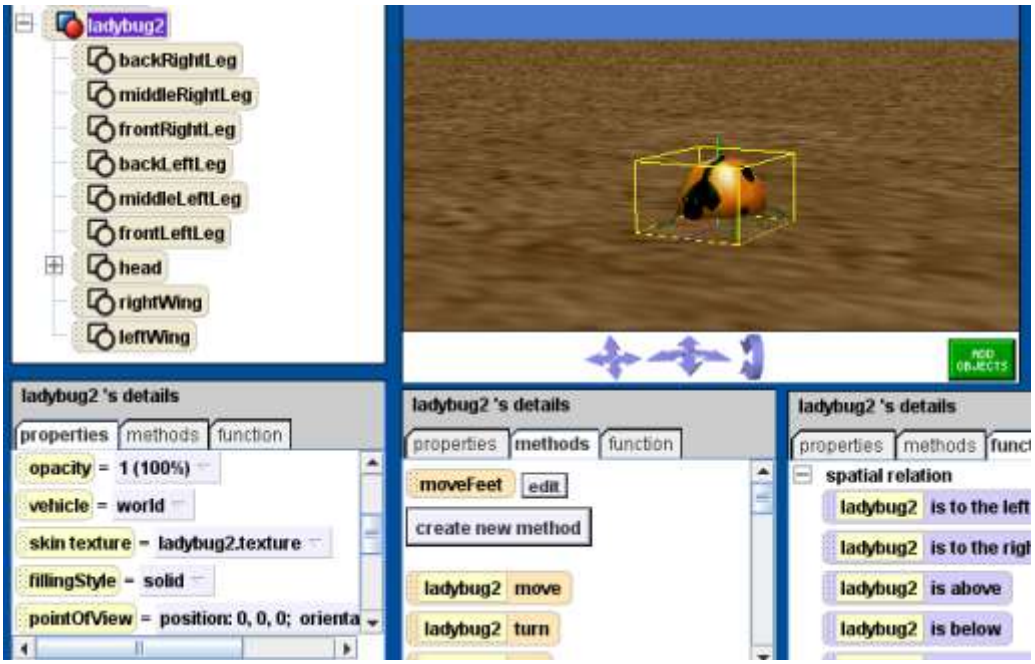
INTRODUCTION TO OOP



Objective

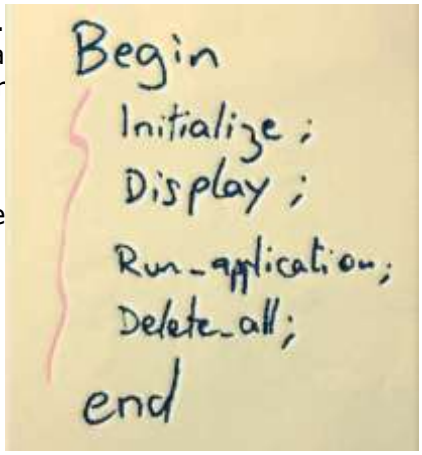
To be able

- to explain with simple words what Object Oriented Programming is
- to use objects in a program

Procedural programming

Definition

In procedural programming, a program contains several steps. These steps can be complex, and in this case, each step calls a routine also known as procedure, method, sub-routine or function.
It's a *downward analysis.*



Most of the time, procedures and functions change data in the program.

PROGRAM = ALGORITHMS + DATA

Inconvenience

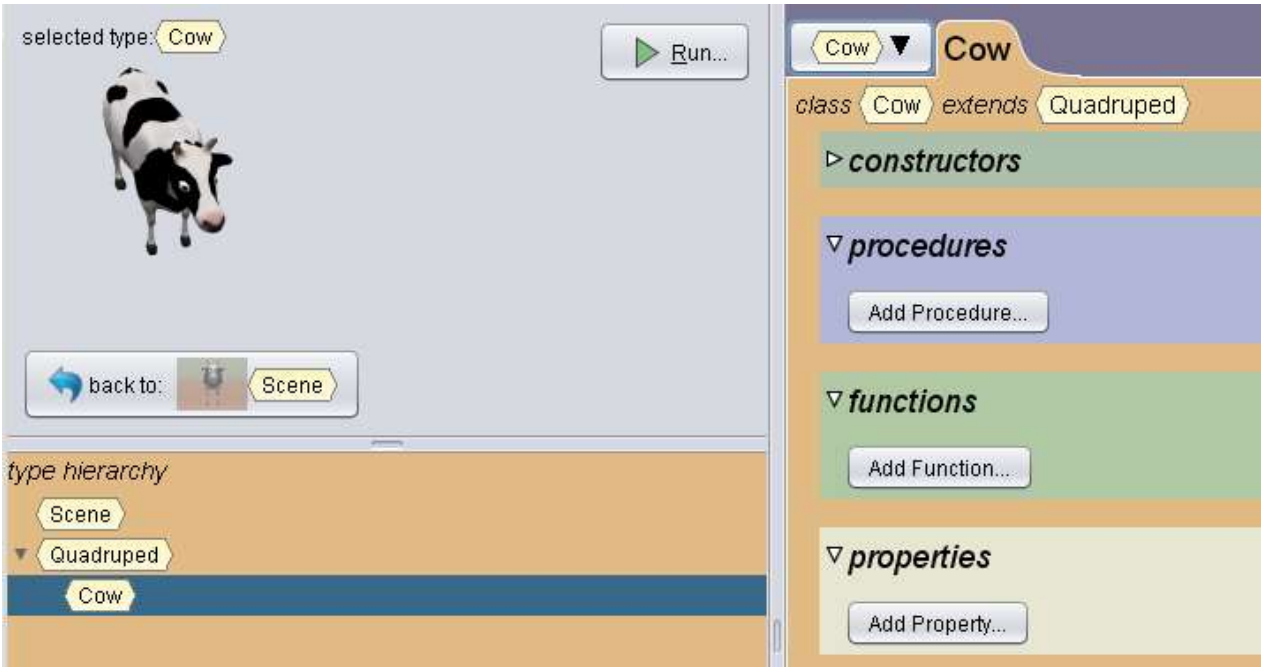
- If you want to change the behavior of a software you need to change the entire code.
- A new development in a software means that we need to write the entire new code
- Software development could be very long and too expensive

Object oriented Programming (OOP)

Definition

We saw that procedural programming isn't a good method if we need to develop complex softwares.
The OOP is a programming approach where the different software elements are described as an object.

Each element in the real world could be represent as an object.



For example a Cow
- has different actions : to walk, to moo, to lie, ...
- has several specifications : color, size, name, weight...

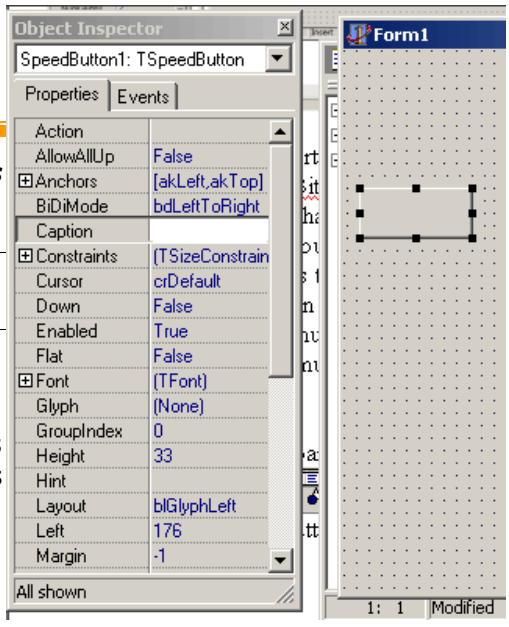
To represent this animal in object approach Pour représenter cet animal dans l'approche objet :

- ***Procedures***, ***Methods*** and ***Functions*** discribe it behavior.
- ***Properties*** specifies the specifications.

Object

An object is a package code which contains ***procedures*** and ***properties***.

OBJECT = PROPERTIES + PROCEDURES



Example :

A button in a Window is an object with his properties (Caption, Down, Enabled, glyph, etc...) and his procedures (clicked(), visble(), setstate(), setposition())

OOP vocabulary

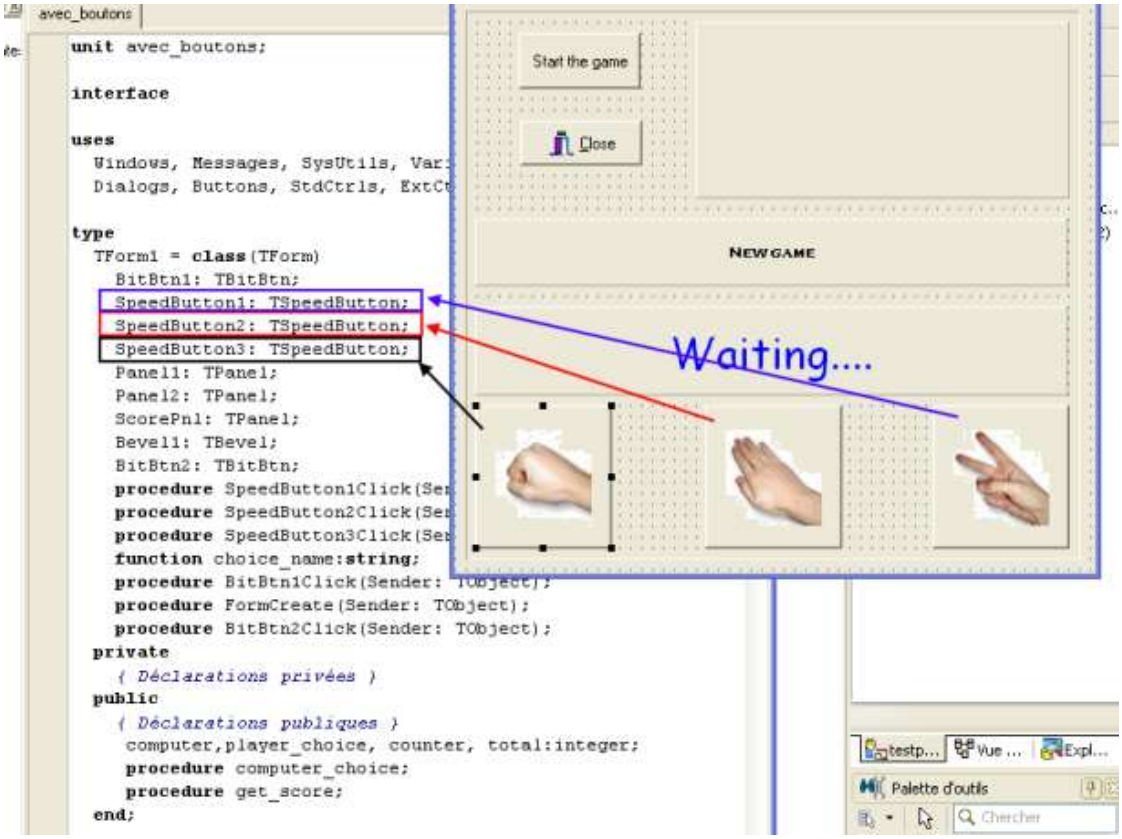
Class

A class is an extensible program-code-template for creating objects. The class is the "mould" of an object. When an object is placed in a program, it is called an ***instance*** of the class.

Objects of the same class will have the same properties but with different values.

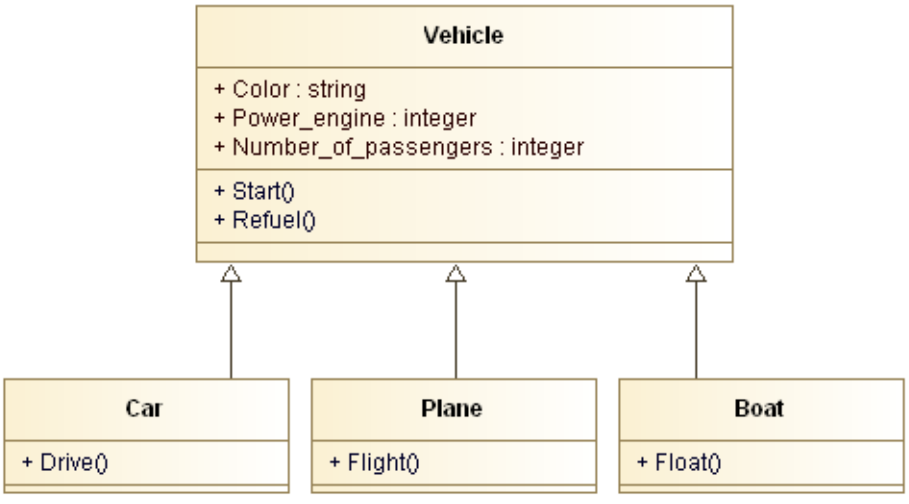
Example :

In this program there are three buttons. TSpeedButton is the class and SpeedButton1, SpeedButton2, SpeedButton3 are instances of this class.
For each instance, the *glyph* propertie (the button picture) is different.



Inheritance

The inheritance is a mechanism by which we define a class of object as being a particular case of a more general class.



Thanks the inheritance, it is possible to reuse the code and to develop complex softwares in a short time.

Exemple
Vehicle is an objet with 3 properties : ***COLOR, POWER_ENGINE, NUMBER_OF_PASSENGERS*** and 2 methods ***START()*** and ***REFUEL().***
The objects CAR, PLANE and BOAT inherit the properties and the Methods of VEHICLE.
They have also their own methods

Encapsulation

L'encapsulation est un mécanisme consistant à rassembler les données et les méthodes au sein d'une structure en cachant l'implémentation de l'objet, c'est-à-dire en empêchant l'accès aux données et aux codes informatiques par un autre moyen que les services proposés.

L'encapsulation permet donc de garantir l'intégrité des données contenues dans l'objet.

Pour ce faire, lors de l'écriture des classes on dispose de trois niveaux de lisibilité :

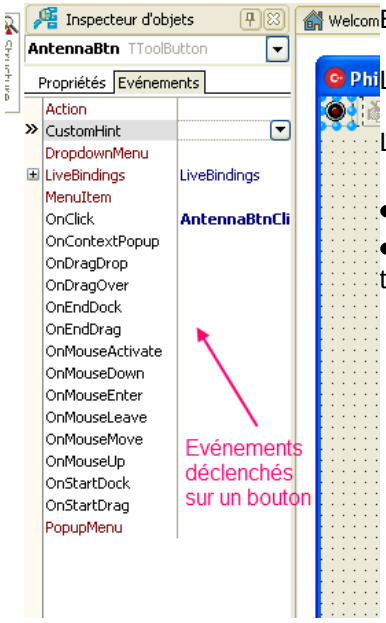
- **publique:** les fonctions de toutes les classes peuvent accéder aux données ou aux méthodes d'une classe définie avec le niveau de visibilité public.
- **privée:** l'accès aux données est limité aux méthodes de la classe elle-même. Il s'agit du niveau de protection des données le plus élevé
- **protégée:** l'accès aux données est réservé aux fonctions des classes héritières, c'est-à-dire par les fonctions membres de la classe ainsi que des classes dérivées

QT Creator :

mainwindow.h | <Selectionner un symbole>

```
1  #ifndef MAINWINDOW_H
2  #define MAINWINDOW_H
3
4  #include <QMainWindow>
5
6  namespace Ui {
7  class MainWindow;
8  }                          MainWindow hérite de QMainWindow
9
10 class MainWindow : public QMainWindow
11 {
12     Q_OBJECT
13
14 public:                                          Déclaration
15     explicit MainWindow(QWidget *parent = 0);    publique
16     ~MainWindow();
17
18 public slots:
19    void on_SetBtn_clicked();
20    void on_ResetBtn_clicked();
21
22 private:                                         Déclaration
23     Ui::MainWindow *ui;                          privée
24
25 };
26
27 #endif // MAINWINDOW_H
28
```

Messages



En programmation objet, les objets s'animent grâce aux méthodes.

Les méthodes sont activées grâce à des messages qui leur sont envoyés.

Les messages peuvent provenir

- d'un élément extérieur : clavier, souris, etc....
- de mécanismes internes au programme orienté objet : temporisation, signal provenant d'un autre objet.....

Vocabulary

Downward analysis	Analyse descendante
inheritance	Héritage
Object	objet
to mould	Mouler