OOP – Car Class

Functions vs Procedures

In Delphi:

- a **procedure** is a subroutine that performs a specific task and**does not return a value**

  ```
  procedure PrintMessage (Message: string);
  begin
  Showmessage (Message);
  end;
  ```

  In this example, the PrintMessage procedure takes a string parameter called Message and outputs it using a ShowMessage
- while a **function** is a subroutine that **returns a value**after performing a specific task.

  ```
  function AddNumbers (a, b : Integer ) : Integer ;
  begin
  result := a + b;
  end;
  ```

  In this example, the AddNumbers function takes two Integer parameters called *a* and *b*, adds them together, and returns the result using the `result` keyword.

OOP

In programming, objects and classes are two fundamental concepts used in object-oriented programming (OOP).

1. Class

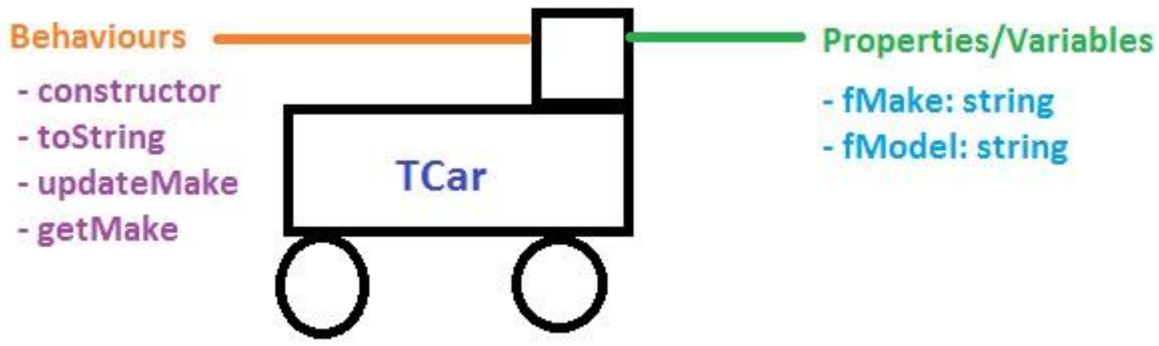
- A class is a blueprint or a template for creating objects.
- It defines a set of properties and behaviours that an object of that class will have.

1.1. Class basic template

```
unit Example_U ;

interface
uses StdCtrls , SysUtils ;

type
  TExample  = class(TObject)
    private
      // variables here, precede with f.
      // ie fName: String
      // also known as properties
    public
      // proecedures, functions, constructure, tostring
      // declarations here
      // also known as behaviors

  end;

implementation
      // implementation goes here

end.
```

1.2. Class Example

For example, a class can be defined to represent a car, with properties such as make and model, and behaviors such as update the car make.



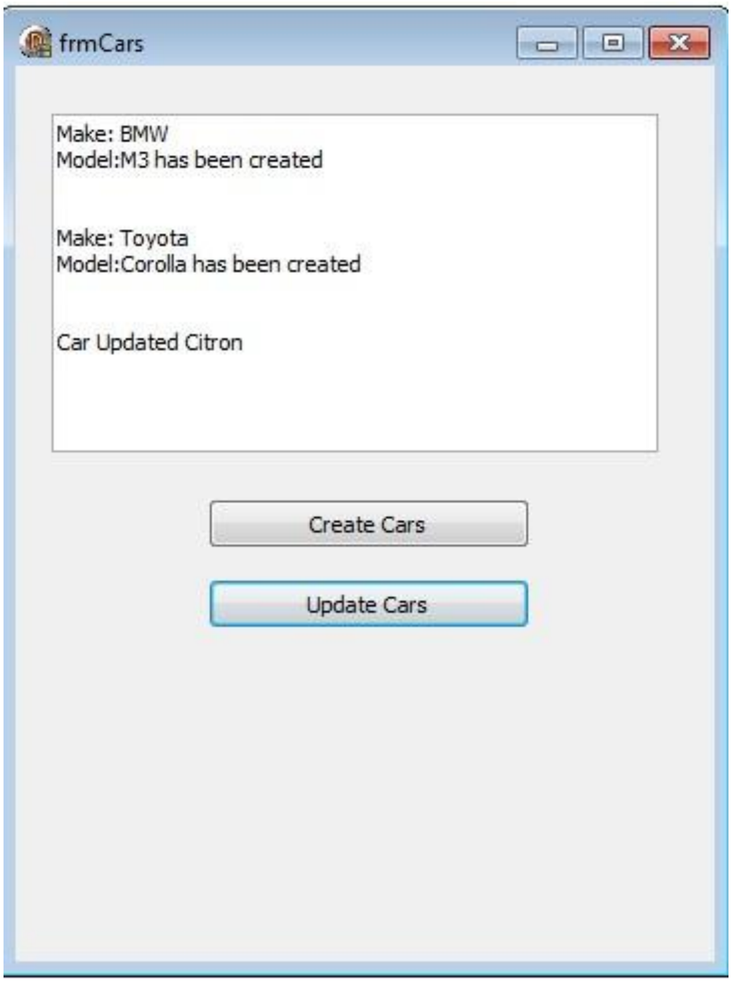
1.2.1. Car Clas Code

```pascal
unit Car_U;

interface

uses StdCtrls, SysUtils;

type
  TCar = class(TObject)
  private
    // properties
    fMake: string;
    fModel: string;
  public
    // behaviours
    constructor Create(sMake,sModel : string);
    procedure updateMake (sMake : string);
    function getMake: string;
    function toString : string;
  end;

implementation

constructor TCar.Create(sMake: string; sModel: string);
begin
  fMake := sMake;
  fModel := sModel;
end;

procedure TCar.updateMake(sMake: string);
begin
  fMake := sMake;
end;

function TCar.getMake : string;
begin
  result := fMake;
end;

function TCar.toString : string;
begin
  result := 'Make: ' + fMake + #13 + 'Model:' + fModel;
end;

end.
```

In the main Unit we create two **instances** of the car class. The first button uses the constructor and toString method.

> Create Cars uses the constructor to create/instantiate the two cars. It then uses the class method **tostring** to output values to the richedit The second button UpdateCars:

> Uses the procedure updateMake to update the cars make. Then the function getMake to return the update Make of the car.


```
Make: BMW
Model:M3 has been created


Make: Toyota
Model:Corolla has been created


Car Updated Citron
```

1.2.2. Main Form Code

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs,Car_U, StdCtrls, ComCtrls;

type
  TfrmCars = class(TForm)
    redOutput: TRichEdit;
    btnCreate: TButton;
    btnUpdate: TButton;
    procedure btnCreateClick(Sender: TObject);
    procedure btnUpdateClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmCars: TfrmCars;
  objBMW : TCar;
  objToyota : TCar;

implementation

{$R *.dfm}

procedure TfrmCars.btnCreateClick(Sender: TObject);
begin
  objBMW := TCar.Create('BMW','M3');
  objToyota := TCar.Create('Toyota','Corolla');
  redoutput.Lines.Add(objBMW.toString  + ' has been created');
  redOutput.Lines.Add(#13);
  redoutput.Lines.Add(objToyota.toString  + ' has been created');
  redOutput.Lines.Add(#13);
end;

procedure TfrmCars.btnUpdateClick(Sender: TObject);
begin
  objBMW.updateMake('Citron');
  redOutput.Lines.Add('Car Updated' + ' ' + objBMW.getMake);
end;

end.
```

2. Object

- An object, on the other hand, is an instance of a class.
- It is a concrete entity that has the properties and behaviours defined by its class.

For example, an object of the car class can represent a specific car, with a particular colour, make, and model, and can be started, stopped, and accelerated.

In summary, classes are used to define the blueprint or template for creating objects, while objects are the concrete entities that have properties and behaviours defined by their class.

3. Special Class Methods

3.1. ToString Method

In Delphi OOP, ToString is a method used to convert an object to a string.

> In simple terms, it is just taking all the variables/properties of the class and outputting it all together.

```
function TPlayer.toString : String;
begin
result := 'Name: ' + fPlayerName + #13 + 'Weight: ' +
FloatToStr(fWeightOfPlayer)+ #13+ 'Current score is: ' + intToStr
(fScore);
```

3.2. Constructor

In Delphi OOP, a constructor is a special method or procedure that is called when an object is created. It is responsible for initializing the object's properties and allocating any necessary resources.

```
constructor TCar.Create(sMake: string; sModel: string);
begin
  fMake := sMake;
  fModel := sModel;
end;
```