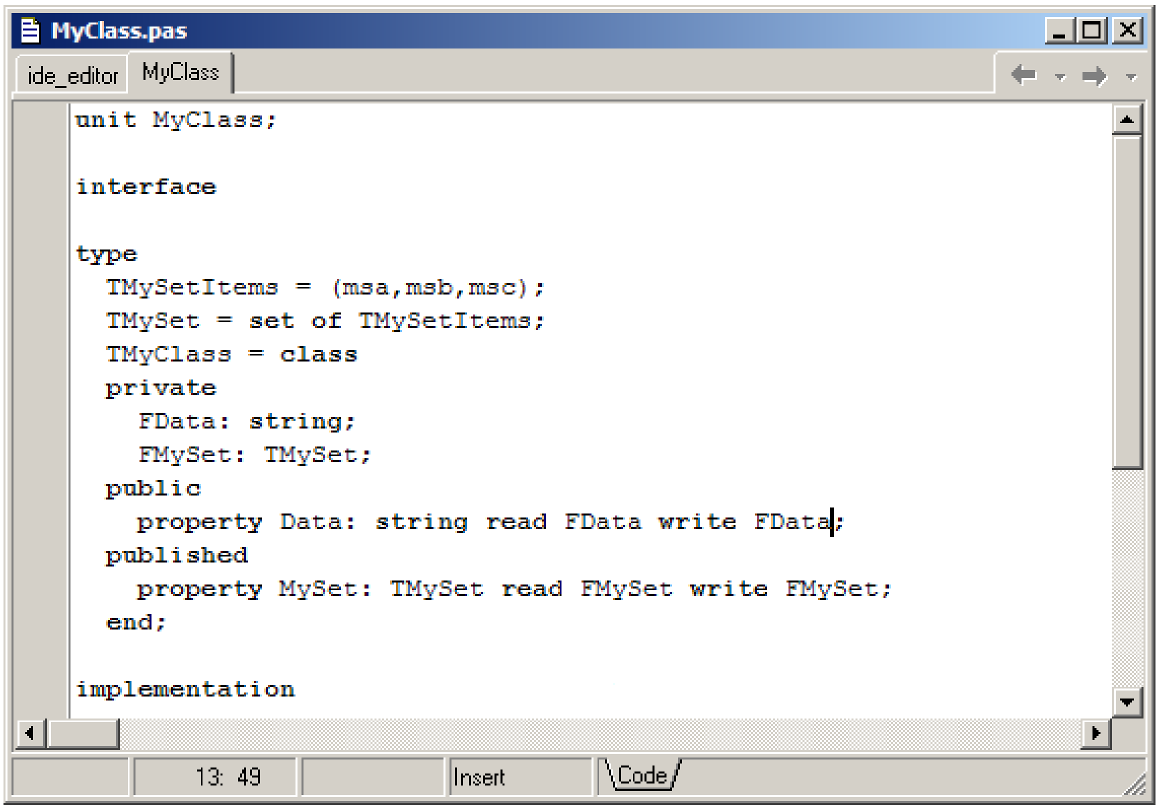
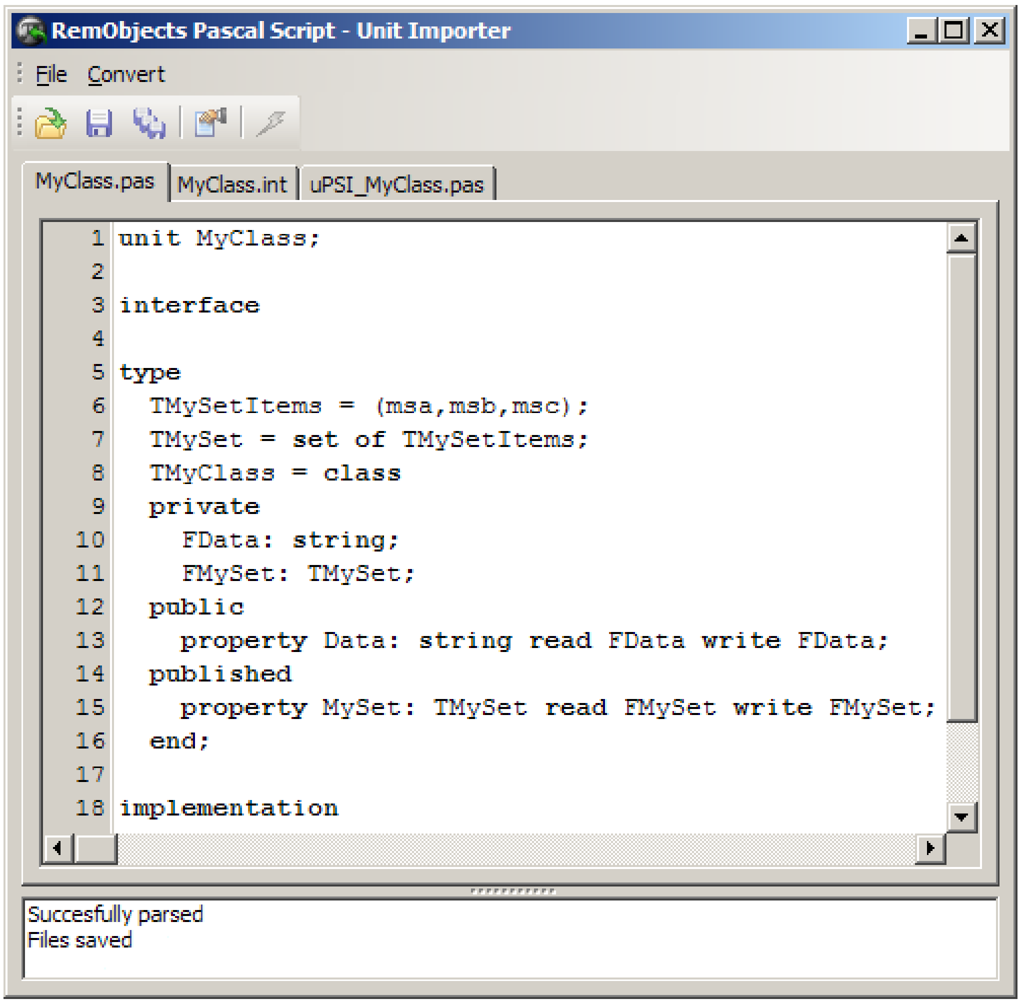
PS02 - Using Classes with RemObjects PascalScript

This article explains how to use existing classes in RemObjects Pascal Script by using a copy of the \Samples\Debug project. The following class has been added to that project:



```
unit MyClass;

interface

type
  TMySetItems = (msa,msb,msc);
  TMySet = set of TMySetItems;
  TMyClass = class
  private
    FData: string;
    FMySet: TMySet;
  public
    property Data: string read FData write FData;
  published
    property MySet: TMySet read FMySet write FMySet;
  end;

implementation
```

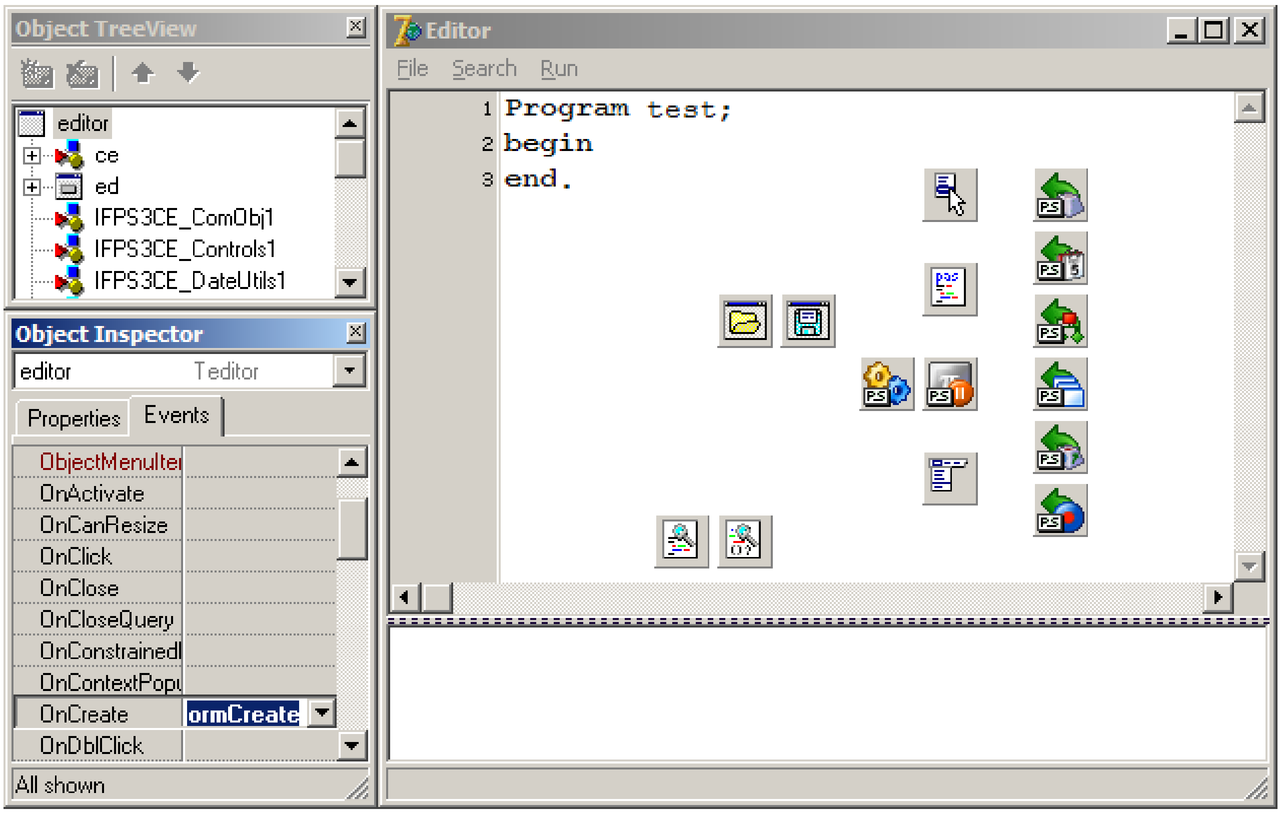
The MyClass file is then opened in the Unit Import tool, which generates units for us:



```
unit MyClass;

interface

type
  TMySetItems = (msa,msb,msc);
  TMySet = set of TMySetItems;
  TMyClass = class
  private
    FData: string;
    FMySet: TMySet;
  public
    property Data: string read FData write FData;
  published
    property MySet: TMySet read FMySet write FMySet;
  end;

implementation
```

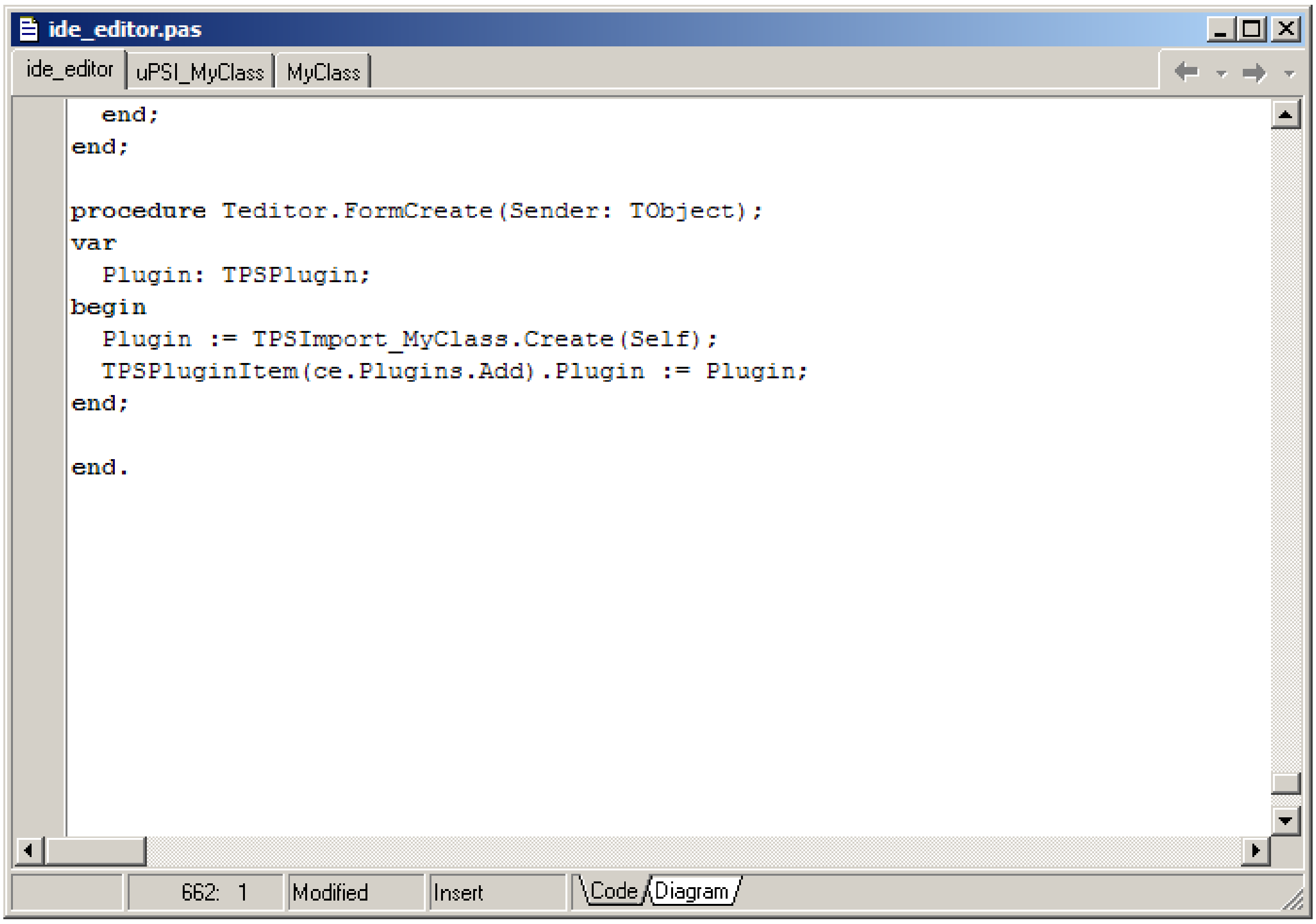
```
Succesfully parsed
Files saved
```

Press the"Go" button and add the generated uPSI_MyClass.pas to your project. Now select the main form and create a handler for its OnCreate:



```
Program test;
begin
end.
```

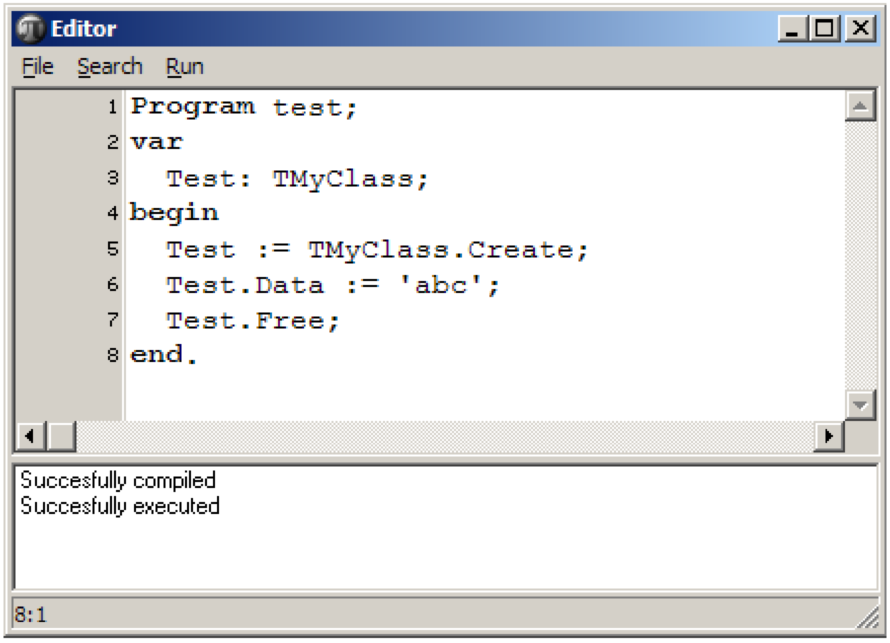
In the Create event, we instantiate a TPSImport_MyClass (i.e. the class generated by the import tool) and add it to the script engine's plugin list:



```
  end;
end;

procedure Teditor.FormCreate(Sender: TObject);
var
  Plugin: TPSPlugin;
begin
  Plugin := TPSImport_MyClass.Create(Self);
  TPSPluginItem(ce.Plugins.Add).Plugin := Plugin;
end;

end.
```

When you are going to use this plugin class from multiple projects, you can also install it in a package so that it gets installed in the Component Palette.

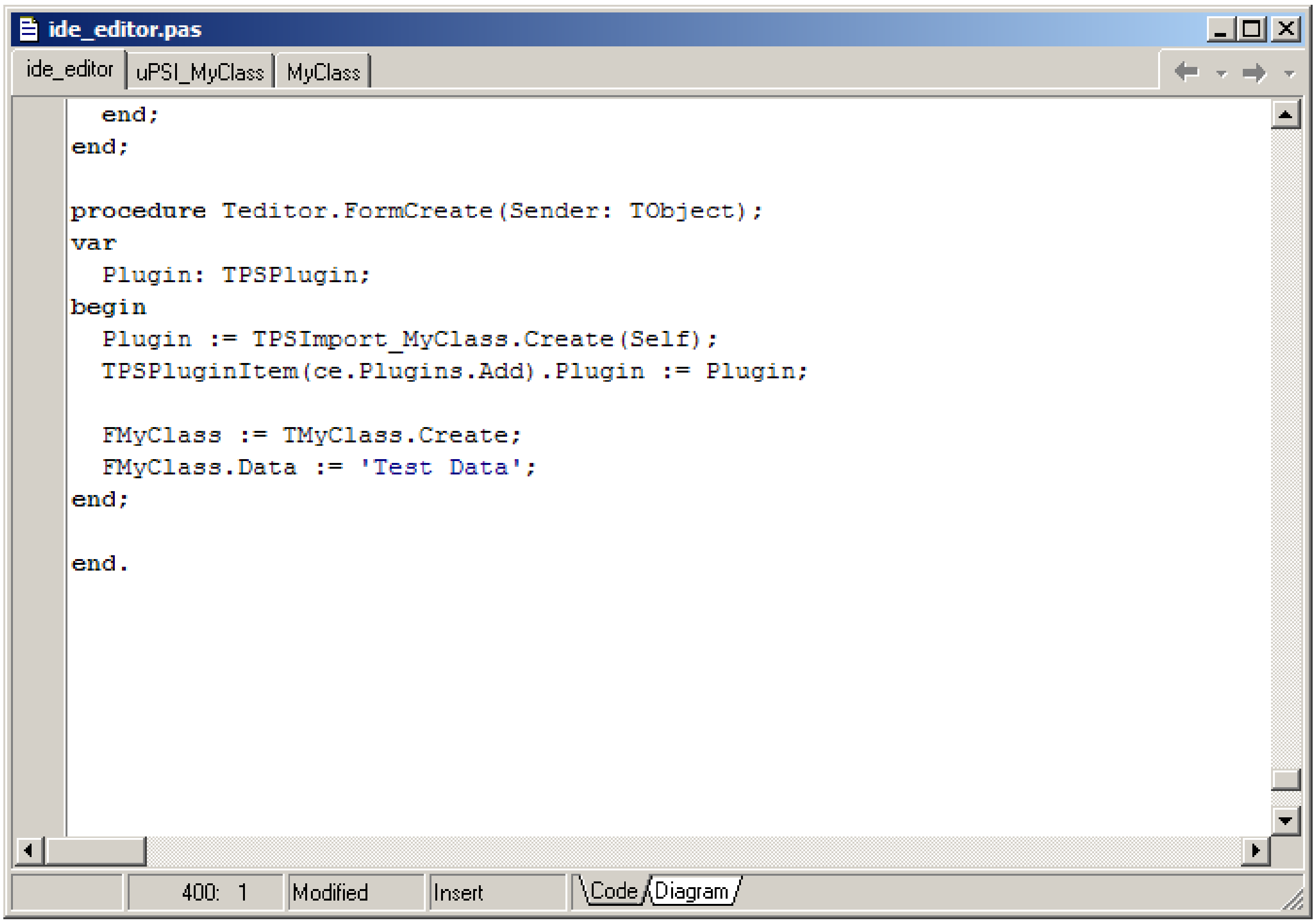
Now it's possible to use the class from the script engine:



```
Program test;
var
  Test: TMyClass;
begin
  Test := TMyClass.Create;
  Test.Data := 'abc';
  Test.Free;
end.
```

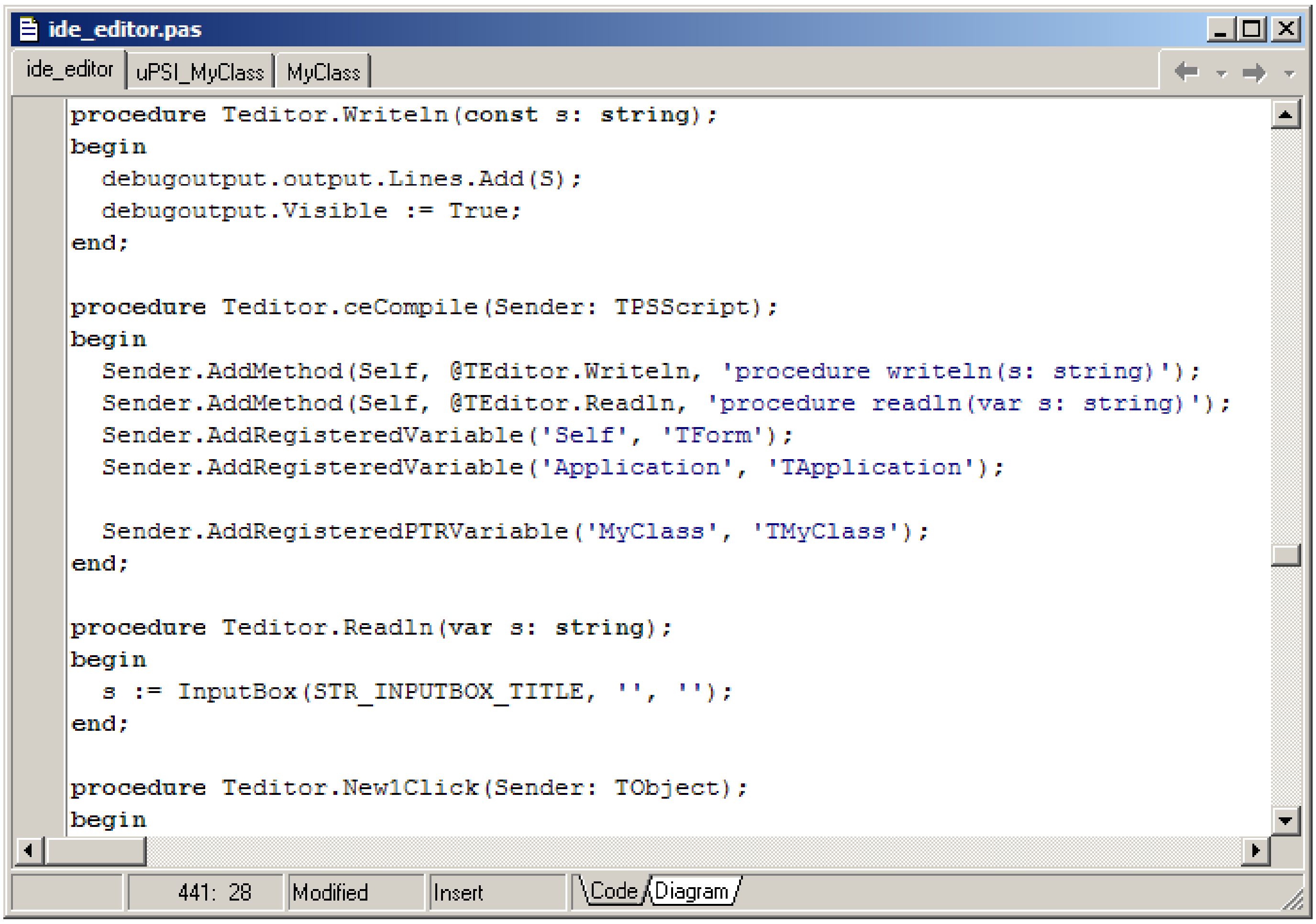
```
Succesfully compiled
Succesfully executed
```

Using an existing TMyClass in the script engine
It's also possible to use existing TMyClass variables from the script engine. To do this, you add a FMyClass: TMyClass to your form class and create it:



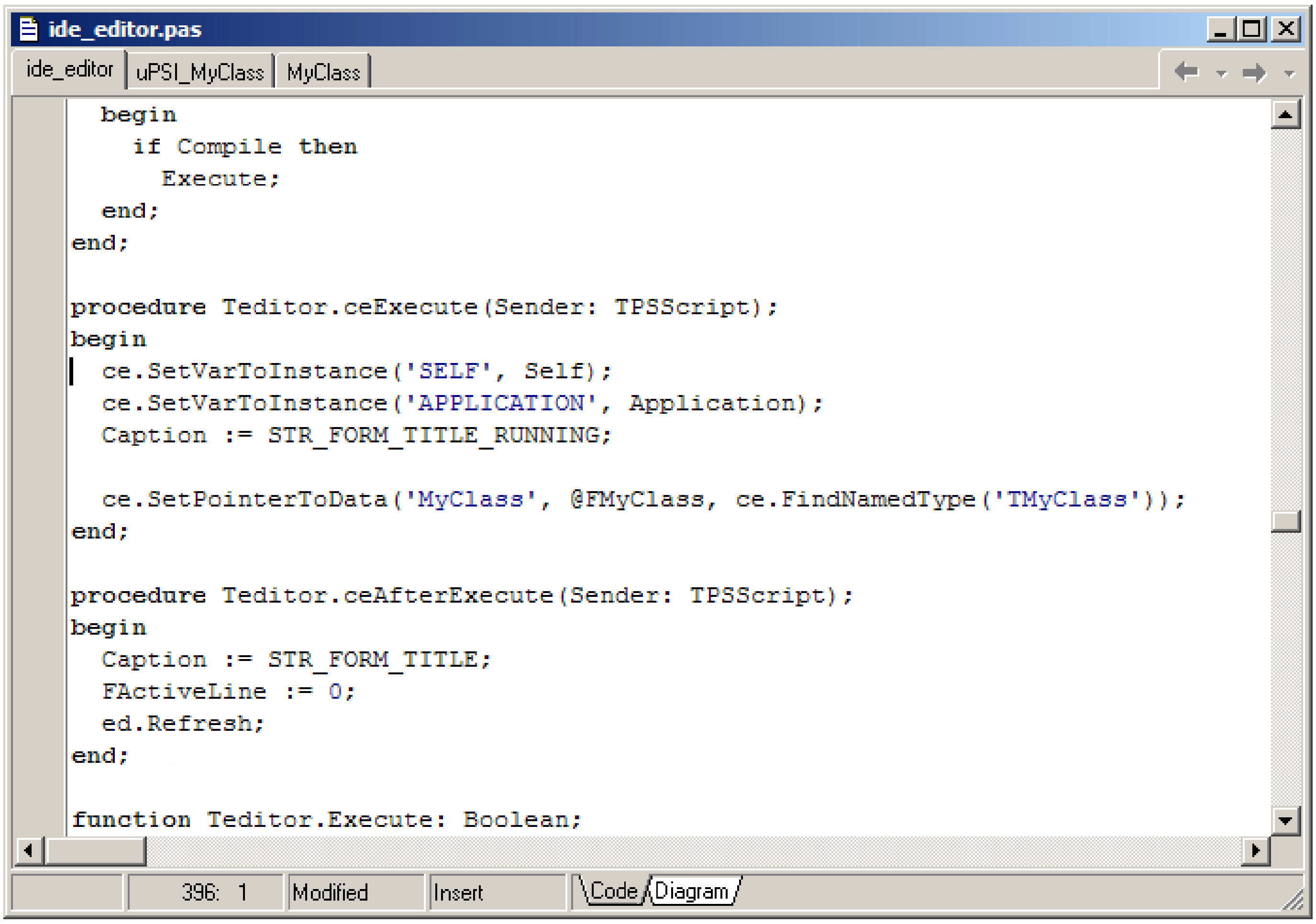
```
  end;
end;

procedure Teditor.FormCreate(Sender: TObject);
var
  Plugin: TPSPlugin;
begin
  Plugin := TPSImport_MyClass.Create(Self);
  TPSPluginItem(ce.Plugins.Add).Plugin := Plugin;

  FMyClass := TMyClass.Create;
  FMyClass.Data := 'Test Data';
end;

end.
```

In the script engine's OnCompile Event you need to let the engine know that there is a new variable. There are two ways to do this: AddRegisteredVariable and AddRegisteredPtrVariable. The difference is that AddRegisteredPtrVariable will make changes directly to the variable, so if you free the FMyClass and re-create it, the script engine will use the new variable. AddRegisteredPTRVariable has two parameters. A name and a type. Add this code to the script engine's OnCompile event:



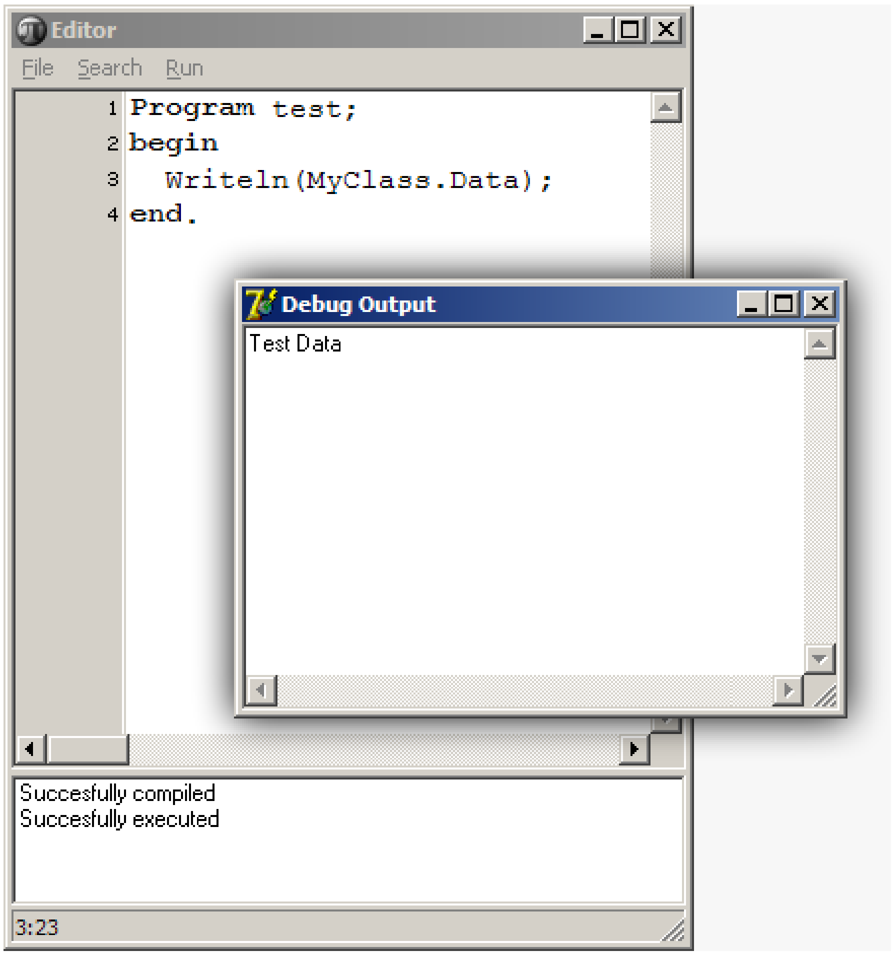
```
procedure Teditor.Writeln(const s: string);
begin
  debugoutput.output.Lines.Add(S);
  debugoutput.Visible := True;
end;

procedure Teditor.ceCompile(Sender: TPSScript);
begin
  Sender.AddMethod(Self, @TEditor.Writeln, 'procedure writeln(s: string)');
  Sender.AddMethod(Self, @TEditor.Readln, 'procedure readln(var s: string)');
  Sender.AddRegisteredVariable('Self', 'TForm');
  Sender.AddRegisteredVariable('Application', 'TApplication');

  Sender.AddRegisteredPTRVariable('MyClass', 'TMyClass');
end;

procedure Teditor.Readln(var s: string);
begin
  s := InputBox(STR_INPUTBOX_TITLE, '', '');
end;

procedure Teditor.New1Click(Sender: TObject);
begin
```

Now the compiler knows about the variable, but the runtime part of the script engine doesn't know where the variable is yet. To let the runtime know, you can use the OnExecute event. The SetPointerToData method can set the value of a script engine variable added with AddRegisteredPTRVariable. The first parameter is the variable name, the second is the address of the variable that contains the instance and the last parameter contains the required type info for the variable (generally you use ce.FindNamedType('nameoftype') here).



```
  begin
    if Compile then
      Execute;
  end;
end;

procedure Teditor.ceExecute(Sender: TPSScript);
begin
  ce.SetVarToInstance('SELF', Self);
  ce.SetVarToInstance('APPLICATION', Application);
  Caption := STR_FORM_TITLE_RUNNING;

  ce.SetPointerToData('MyClass', @FMyClass, ce.FindNamedType('TMyClass'));
end;

procedure Teditor.ceAfterExecute(Sender: TPSScript);
begin
  Caption := STR_FORM_TITLE;
  FActiveLine := 0;
  ed.Refresh;
end;

function Teditor.Execute: Boolean;
```

The final result is:



```
Program test;
begin
  Writeln(MyClass.Data);
end.
```

```
Test Data
```

```
Succesfully compiled
Succesfully executed
```