Created for http://bugs.freepascal.org/view.php?id=25607

by Maciej Izak (aka. hnb) - hnb.code@gmail.com

All of examples can't be compiled with FPC, but they works with no problem in any Delphi version (tested on D2007, XE2 32bit and 64bit, XE4 32bit and 64bit).

Example 1: Constructors with override

```
type
  TA = class
    constructor Create(A: Integer = 0); overload; virtual;
  end;

  TB = class(TA)
    constructor Create(A: Integer); overload; override;
  end;

  TClassB = class of TB;

constructor TA.Create(A: Integer = 0);
begin
  WriteLn('TA.Create');
end;

constructor TB.Create(A: Integer);
begin
  WriteLn('TB.Create');
end;

var
  B: TB;
  ClassB: TClassB;
begin
  B := TB.Create; // TA.Create (VMT is not used
                  // compiler can determine)

  B.Create; // call TB.Create because of VMT rules

  ClassB := TB;
  ClassB.Create; // call TB.Create because of VMT rules

  ReadLn;
end.
```

Example 2: Constructors without override

```
type
  TA = class
    constructor Create(A: Integer = 0); overload;
  end;

  TB = class(TA)
    constructor Create(A: Integer); overload;
  end;

  TClassB = class of TB;

constructor TA.Create(A: Integer = 0);
begin
  WriteLn('TA.Create');
end;

constructor TB.Create(A: Integer);
begin
  WriteLn('TB.Create');
end;

var
  B: TB;
  ClassB: TClassB;
begin
  B := TB.Create; // TA.Create (VMT is not used
                  // compiler can determine)

  B.Create; // call TA.Create because of VMT rules

  ClassB := TB;
  ClassB.Create; // call TA.Create because of VMT rules

  ReadLn;
end.
```

Example 3: Class methods with override

```
type
  T0 = class
    class procedure Foo;
  end;

  TA = class(T0)
    class procedure Foo(A: Integer = 0); overload; virtual;
  end;

  TB = class(TA)
    class procedure Foo(A: Integer); overload; override;
  end;

  TClassB = class of TB;

class procedure T0.Foo();
begin
  WriteLn('T0.Foo');
end;

class procedure TA.Foo(A: Integer = 0);
begin
  WriteLn('TA.Foo');
end;

class procedure TB.Foo(A: Integer);
begin
  WriteLn('TB.Foo');
end;

var
  B: TB;
  ClassB: TClassB;
begin
  TB.Foo; // call TA.Foo (VMT is not used, compiler can determine)

  B := TB.Create;
  B.Foo; // call TB.Foo because of VMT rules

  ClassB := TB;
  ClassB.Foo; // call TB.Foo because of VMT rules

  ReadLn;
end.
```

Example 4: Class methods without override

```
type
  T0 = class
    class procedure Foo;
  end;

  TA = class(T0)
    class procedure Foo(A: Integer = 0); overload;
  end;

  TB = class(TA)
    class procedure Foo(A: Integer); overload;
  end;

  TClassB = class of TB;

class procedure T0.Foo();
begin
  WriteLn('T0.Foo');
end;

class procedure TA.Foo(A: Integer = 0);
begin
  WriteLn('TA.Foo');
end;

class procedure TB.Foo(A: Integer);
begin
  WriteLn('TB.Foo');
end;

var
  B: TB;
  ClassB: TClassB;
begin
  TB.Foo; // call TA.Foo (VMT is not used, compiler can determine)

  B := TB.Create;
  B.Foo; // call TA.Foo because of VMT rules

  ClassB := TB;
  ClassB.Foo; // call TA.Foo because of VMT rules

  ReadLn;
end.
```

Example 5: Methods with override

```
type
  T0 = class
    procedure Foo;
  end;

  TA = class(T0)
    procedure Foo(A: Integer = 0); overload; virtual;
  end;

  TB = class(TA)
    procedure Foo(A: Integer); overload; override;
  end;

  TClassB = class of TB;

procedure T0.Foo();
begin
  WriteLn('T0.Foo');
end;

procedure TA.Foo(A: Integer = 0);
begin
  WriteLn('TA.Foo');
end;

procedure TB.Foo(A: Integer);
begin
  WriteLn('TB.Foo');
end;

var
  B: TB;
  ClassB: TClassB;
begin
  B := TB.Create;
  B.Foo; // call TB.Foo because of VMT rules

  ReadLn;
end.
```

Example 6: Methods without override

```
type
  T0 = class
    procedure Foo;
  end;

  TA = class(T0)
    procedure Foo(A: Integer = 0); overload;
  end;

  TB = class(TA)
    procedure Foo(A: Integer); overload;
  end;

  TClassB = class of TB;

procedure T0.Foo();
begin
  WriteLn('T0.Foo');
end;

procedure TA.Foo(A: Integer = 0);
begin
  WriteLn('TA.Foo');
end;

procedure TB.Foo(A: Integer);
begin
  WriteLn('TB.Foo');
end;

var
  B: TB;
  ClassB: TClassB;
begin
  B := TB.Create;
  B.Foo; // call TA.Foo because of VMT rules

  ReadLn;
end.
```