

Объектно-ориентированное программирование

Объекты: свойства и методы

Основной единицей в объектно-ориентированном программировании является *программный объект*, который объединяет в себе как описывающие его данные (свойства), так и средства обработки этих данных (методы).

Если говорить образно, то объект — это существительное, свойства объекта — прилагательные, методы — глаголы.

Классы объектов

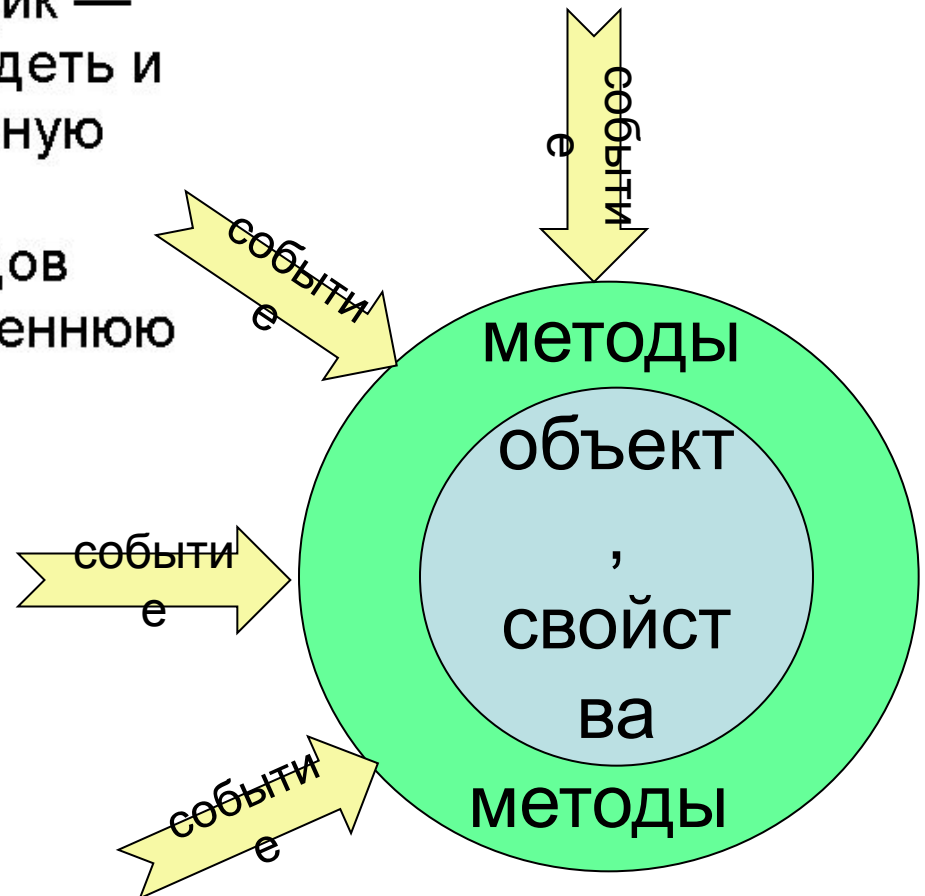
Классы объектов являются шаблонами, определяющими наборы свойств, методов и событий, по которым создаются объекты.

Объект, созданный по шаблону класса, является ***экземпляром класса***.

Различные экземпляры класса обладают одинаковым набором свойств, однако значения свойств у них могут быть различными.

Инкапсуляция

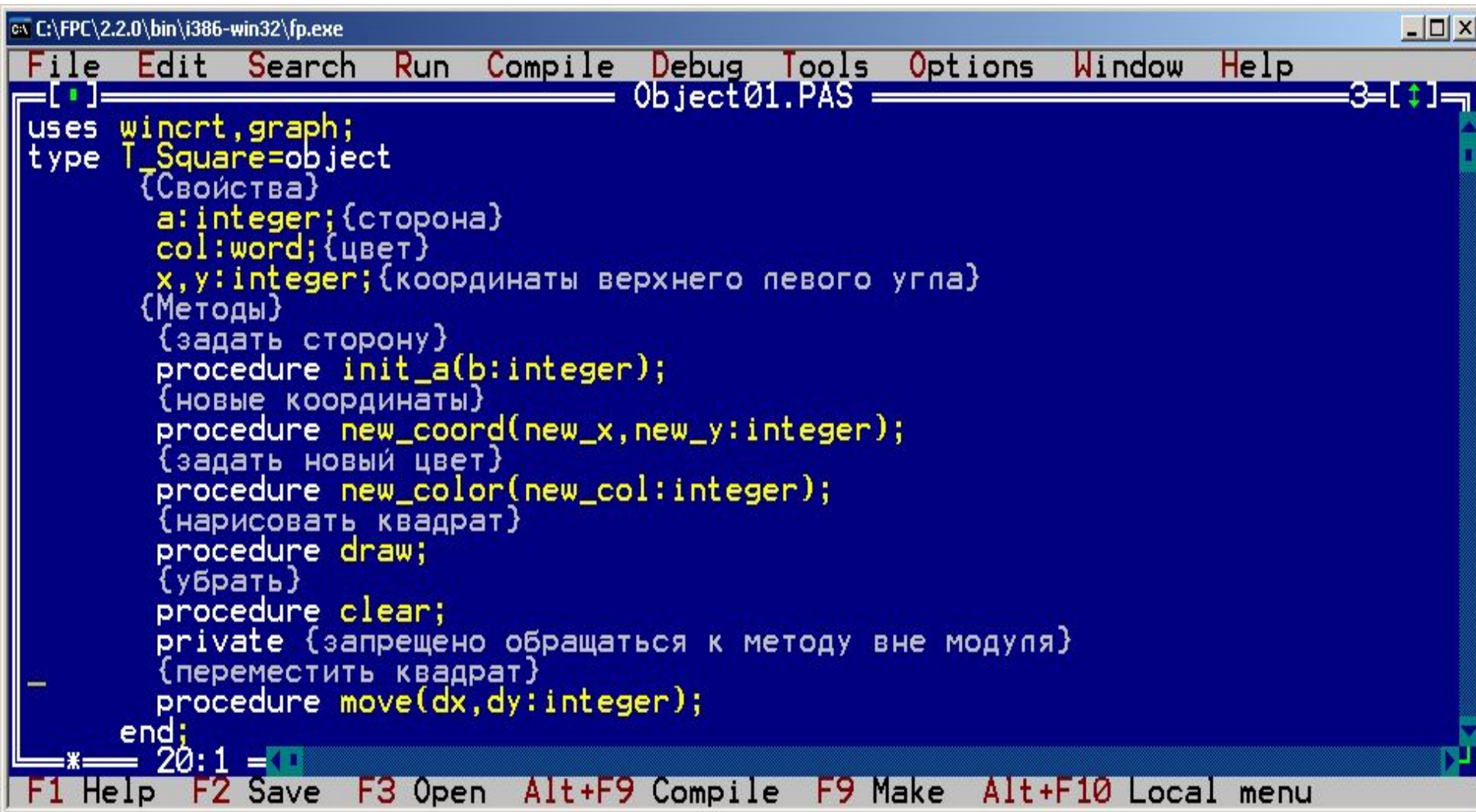
Инкапсуляция — это принцип, согласно которому любой класс должен рассматриваться как чёрный ящик — пользователь класса должен видеть и использовать только интерфейсную часть класса (т. е. список декларируемых свойств и методов класса) и не вникать в его внутреннюю реализацию.



Описание объектов

```
type <Имя объекта> = Object
    <свойство>: <тип>;
    ...
    <свойство>: <тип>;
    procedure <метод>;
    ...
    procedure <метод>;
end;
```

Описание объектов

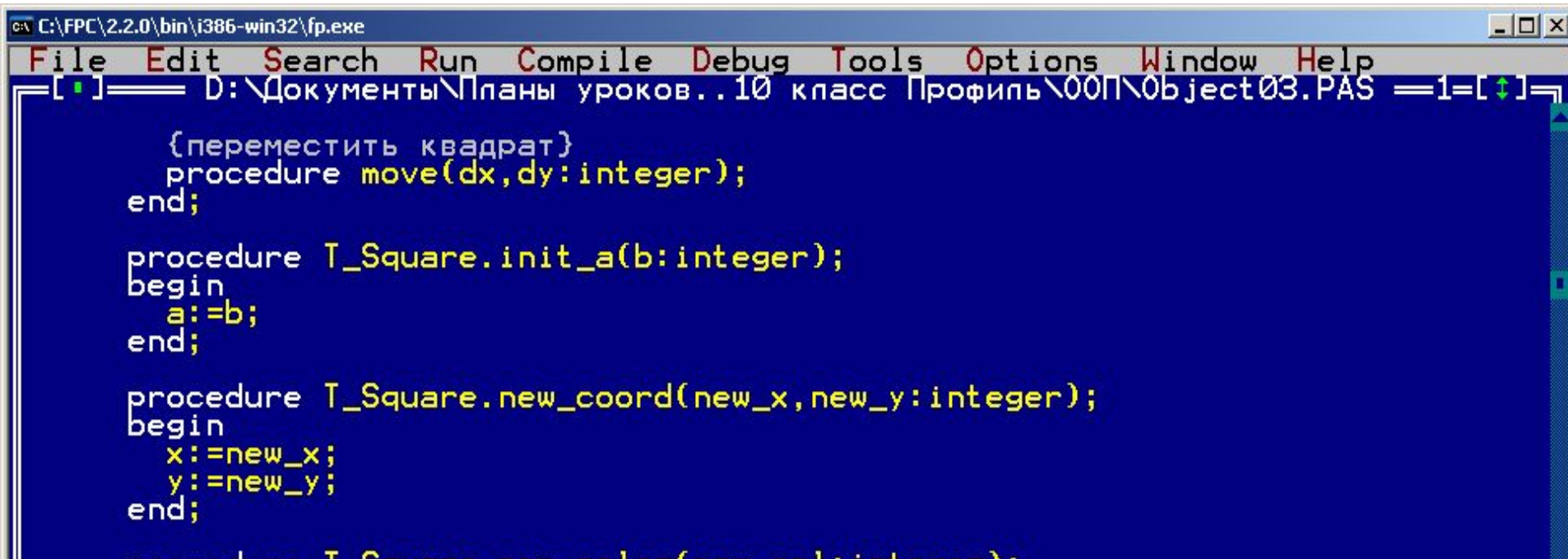


```
C:\FPC\2.2.0\bin\i386-win32\fp.exe
File Edit Search Run Compile Debug Tools Options Window Help
Object01.PAS
uses wincrt, graph;
type T_Square=object
  {Свойства}
  a:integer; {сторона}
  col:word; {цвет}
  x,y:integer; {координаты верхнего левого угла}
  {Методы}
  {задать сторону}
  procedure init_a(b:integer);
  {новые координаты}
  procedure new_coord(new_x,new_y:integer);
  {задать новый цвет}
  procedure new_color(new_col:integer);
  {нарисовать квадрат}
  procedure draw;
  {убрать}
  procedure clear;
  private {запрещено обращаться к методу вне модуля}
  {переместить квадрат}
  procedure move(dx,dy:integer);
end;
* 20:1 =
```

F1 Help F2 Save F3 Open Alt+F9 Compile F9 Make Alt+F10 Local menu

Описание объектов

После декларирования свойств и методов объекта, все процедуры, связанные с методами должны быть описаны (используется точечная нотация).

A screenshot of the Free Pascal Compiler (FPC) IDE. The window title is "C:\FPC\2.2.0\bin\i386-win32\fp.exe". The menu bar includes File, Edit, Search, Run, Compile, Debug, Tools, Options, Window, and Help. The file path is "D:\Документы\Планы уроков..10 класс Профиль\00П\Object03.PAS". The code in the editor is as follows:

```
{переместить квадрат}
procedure move(dx,dy:integer);
end;

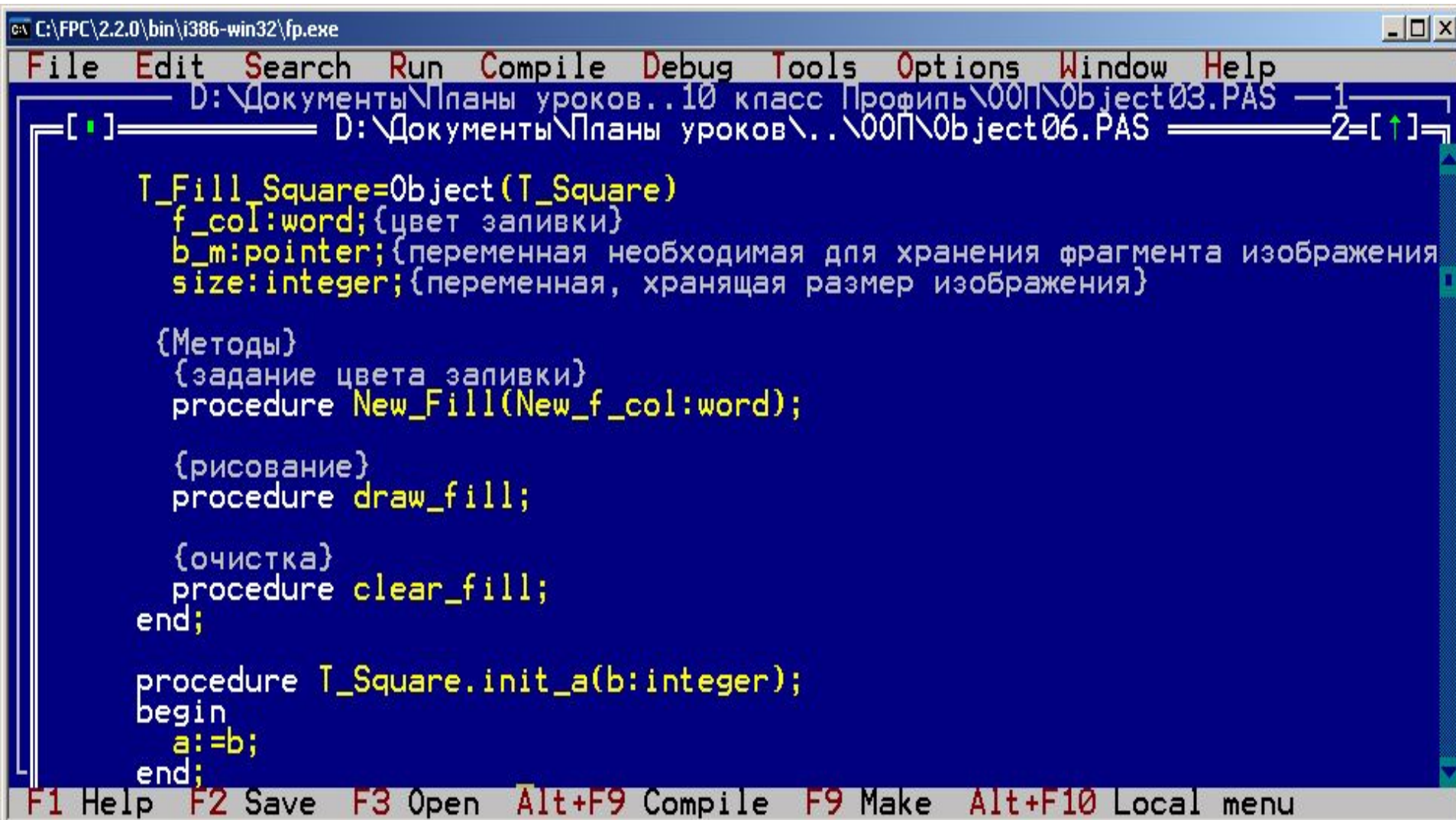
procedure T_Square.init_a(b:integer);
begin
  a:=b;
end;

procedure T_Square.new_coord(new_x,new_y:integer);
begin
  x:=new_x;
  y:=new_y;
end;
```

Наследование

Наследованием называется возможность породить один класс от другого с сохранением всех свойств и методов класса-предка (прародителя, иногда его называют суперклассом) и добавляя, при необходимости, новые свойства и методы. Набор классов, связанных отношением наследования, называют иерархией. Наследование призвано отобразить такое свойство реального мира, как иерархичность.

Наследование



```
C:\FPC\2.2.0\bin\i386-win32\fp.exe
File Edit Search Run Compile Debug Tools Options Window Help
D:\Документы\Планы уроков..10 класс Профиль\00П\Object03.PAS —1—
D:\Документы\Планы уроков\..\00П\Object06.PAS —2=[↑]—

T_Fill_Square=Object(T_Square)
  f_color:word;{цвет заливки}
  b_m:pointer;{переменная необходимая для хранения фрагмента изображения}
  size:integer;{переменная, хранящая размер изображения}

{Методы}
  {задание цвета заливки}
  procedure New_Fill(New_f_color:word);

  {рисование}
  procedure draw_fill;

  {очистка}
  procedure clear_fill;
end;

procedure T_Square.init_a(b:integer);
begin
  a:=b;
end;
```

F1 Help F2 Save F3 Open Alt+F9 Compile F9 Make Alt+F10 Local menu

Полиморфизм

Полиморфизмом называют явление, при котором функции (методу) с одним и тем же именем соответствует разный программный код (полиморфный код) в зависимости от того, объект какого класса используется при вызове данного метода. Полиморфизм обеспечивается тем, что в классе-потомке изменяют реализацию метода класса-предка с обязательным сохранением сигнатуры метода. Это обеспечивает сохранение неизменным интерфейса класса-предка и позволяет осуществить связывание имени метода в коде с разными классами — из объекта какого класса осуществляется вызов, из того класса и берётся метод с данным именем. Такой механизм называется динамическим (или поздним) связыванием — в отличие от статического (раннего) связывания, осуществляемого на этапе компиляции.