УДК 372.862

О РАЗРАБОТКЕ КЛАССОВ В СРЕДЕ ПРОГРАММИРОВАНИЯ DELPHI ПРИ ОБУЧЕНИИ ПРОГРАММИРОВАНИЮ В ШКОЛЕ И ВУЗЕ

Дмитриев В.Л.
Стерлитамакский филиал Башкирского государственного университета
Российская Федерация, г. Стерлитамак

В объектно-ориентированном программировании базовым понятием выступает понятие объекта, который имеет определенные свойства [3, 4]. Состояние объекта задается перечислением значений его признаков (полей). Кроме того, обычно объект располагает набором методов, призванных решать разнообразные задачи. При написании программы, содержащей объекты, можно легко организовать взаимодействие объектов между собой.

Главными достоинствами объектно-ориентированного программирования являются: во-первых, использование абстрагирования и инкапсуляции структуры; во-вторых, то, что объекты представляют собой хорошую модель для большинства сущностей реального мира. Вместе с тем иногда становится сложно понять и представить разрабатываемый алгоритм в целом, поскольку его фрагменты могут быть распределены по многим методам, ассоциированным с различными типами объектов. Поэтому в объектно-ориентированном программировании при написании больших программ в большинстве практически важных случаев также необходимо крайне аккуратно документировать методы. Следует отметить, что преимущества объектно-ориентированного программирования в полной мере проявляются лишь при разработке достаточно сложных программ.

Особое внимание следует уделять обучению объектно-ориентированному программированию в школе. Дело в том, что в настоящее время методика изучения в школе любых видов объектно-ориентированного программирования совершенно недостаточно разработана, возможно, даже можно сказать, что этот процесс находится на начальной стадии. В основном это связано, очевидно, с недостаточным количеством часов, отводимых на изучение информатики – для изучения объектного программирования совершенно не хватает времени. Тем не менее, изучение объектно-ориентированного программирования должно стать привычным в планировании учебной деятельности и организации занятий в современной школе, т.к. именно оно дает возможность учащемуся перейти с уровня простого исполнителя на уровень стратега, позволяющего максимально раскрыть свои творческие способности.

В большинстве случаев в школьном курсе информатики при обучении школьников объектно-ориентированному программированию используется язык Pascal, в пользу которого говорит и тот факт, что в настоящее время очень распространены Pascal ABC и Free Pascal, ориентированные на школьников и студентов младших курсов. При этом Pascal ABC.NET представляет собой систему программирования, которая содержит все основные элементы современных языков программирования: модули, классы, перегрузку операций, обобщенные классы, интерфейсы, дизайнер форм, средства параллельного программирования, а также целый ряд расширений языка Pascal.

С другой стороны, в настоящее время язык программирования Pascal все еще является базовым в большинстве педагогических университетов нашей страны. Как следствие, практически все учителя информатики владеют основами программирования на языке Pascal. Кроме того, во многих случаях наиболее приемлемым переходом к изучению визуальных языков программирования в школе является переход именно от Turbo Pascal к Delphi.

В представленной работе рассматривается поэтапная разработка классов, предназначенных для работы с графическими объектами (на примере прямоугольников), в среде программирования Delphi. Приведенная последовательность этапов позволяет использовать ее для обучения основам объектно-ориентированного программирования школьников старших классов, а также студентов вузов. О важности поэтапной разработки при проектировании больших проектов рассказывается в работе [1], а пример разработки объекта в среде программирования Turbo Pascal рассмотрен в работе [2].

В зависимости от конечного результата, который планируется получить в ходе разработки проекта, будет зависеть количество этапов. Мы поставим целью разработку проекта, в котором можно будет создавать объекты – прямоугольники различных размеров с возможностью организации последующего движения любого из них от их текущего положения до указанного, а также с возможностью изменения их цвета. Также нужно будет предусмотреть возможность удаления объектов.

На первом этапе разработки опишем класс, содержащий следующие поля: координаты прямоугольника, цвет границы и цвет заливки, стиль заливки, толщина линий границы. Методов пока будет только 2: конструктор класса и метод отображения объекта на компоненте типа TImage. При проектировании класса сразу учтем, что класс должен предоставлять возможность располагать объект на любом экземпляре TImage. В результате структура класса будет иметь следующий вид:

```
TRectangle = class
  public
    x1,y1,x2,y2: Integer;
    Color: TColor;
    Style: TBrushStyle;
    B_Color: TColor;
    Width: Byte;
    constructor Create(xx,yy: Integer);
    procedure Paint(Sender: TImage; xx,yy: Integer);
 end;
```

Методы класса TRectangle описаны следующим образом (здесь мы пока будем предполагать, что размеры сторон прямоугольника равны 50):

```
constructor TRectangle.Create(xx,yy: Integer);
begin;
 inherited Create;
 x1:=xx; y1:=yy; x2:=x1+50; y2:=y1+50;
end;
procedure TRectangle.Paint(Sender: TImage; xx,yy: Integer);
begin
 Sender.Canvas.Pen.Color:=Color;
 Sender.Canvas.Pen.Mode:=pmNotXOR;
 Sender.Canvas.Pen.Width:=Width;
 Sender.Canvas.Brush.Color:=B_Color;
 Sender.Canvas.Brush.Style:=Style;
 x1:=xx; y1:=yy; x2:=x1+50; y2:=y1+50;
 Sender.Canvas.Rectangle(x1,y1,x2,y2);
end;
```

После этого объявим экземпляр класса – объект A, а также сразу создадим его и отобразим на компоненте Image1 (предварительно расположенном на форме), используя обработчик FormCreate:

```
procedure TForm1.FormCreate(Sender: TObject);
```

```
begin
  A:=Trectangle.Create(100,100);
  A.Color:=clBlue; A.Style:=bsSolid; A.B_Color:=clRed;
  A.Paint(Form1.Image1,100,100);
end;
```

В обработчике Image1MouseUp напишем одну лишь строчку, позволяющую выводить объект A в точке, где будет отпущена мышь (при этом изображение объекта на старом месте не будет уничтожаться):

```
A.Paint(Form1.Image1,x,y);
```

На этом первый этап разработки завершен, и можно смотреть результат работы программы.

На втором этапе рассмотрим управление объектами на примере двух экземпляров класса TRectangle. Первый объект будет реагировать на отпускание левой кнопки мыши, а второй – правой кнопки. В самом начале на Image1 будут расположены два объекта – A и B. Организуем работу с объектами так, чтобы каждый из них мог плавно двигаться к точке, в которой будет отпущена клавиша мыши. Для этого в классе TRectangle нужно будет сделать существенные изменения. В частности, потребуется ввести экземпляр TTimer, некоторые дополнительные поля и методы, отвечающие за перемещение объекта. После соответствующих изменений описание класса примет следующий вид:

```
TRectangle = class
   private
     T: Ttimer;
     kx,ky,px,py: real;
     xn: integer;
     S: TImage;
     procedure start(Sender: TObject);
   public
     x1,y1,x2,y2: Integer;
     Color: TColor;
     Style: TBrushStyle;
     B_Color: TColor;
     Width: Byte;
     constructor Create(xx,yy: Integer);
     procedure Paint(Sender: TImage; xx,yy: Integer);
     procedure Move(Sender: TImage; xx,yy: Integer);
  end;
```

Здесь метод start будет являться методом, выполняющим роль обработчика OnTimer для экземпляра T типа TTimer. В метод Paint добавится удаление объекта на старом месте. В итоге описание методов класса будет следующим:

```
constructor TRectangle.Create(xx,yy: Integer);
begin;
  inherited Create;
  x1:=xx;  y1:=yy; x2:=x1+50;  y2:=y1+50; T:=TTimer.Create(Form1);
  T.Interval:=10; T.Enabled:=false; T.OnTimer:=start;
end;
procedure TRectangle.Paint(Sender: TImage; xx,yy: Integer);
begin
  Sender.Canvas.Pen.Color:=clwhite; Sender.Canvas.Pen.Mode:=pmcopy;
  Sender.Canvas.Pen.Width:=Width; Sender.Canvas.Brush.Color:=clwhite;
```

```
  Sender.Canvas.Brush.Style:=bssolid; Sender.Canvas.Rectangle(x1,y1,x2,y2);
  Sender.Canvas.Pen.Color:=Color; Sender.Canvas.Pen.Mode:=pmNotXOR;
  Sender.Canvas.Pen.Width:=Width; Sender.Canvas.Brush.Color:=B_Color;
  Sender.Canvas.Brush.Style:=Style;
  x1:=xx; y1:=yy; x2:=x1+50; y2:=y1+50; Sender.Canvas.Rectangle(x1,y1,x2,y2);
end;
procedure TRectangle.Move(Sender: TImage; xx,yy: Integer);
begin
  px:=x1; py:=y1; xn:=xx; kx:=xx-x1; ky:=yy-y1;
  if (abs(kx)>abs(ky)) then begin ky:=ky/abs(kx);
                                  if kx>0 then kx:=1 else kx:=-1; end else
                            begin kx:=kx/abs(ky);
                                  if ky>0 then ky:=1 else ky:=-1; end;
  S:=Sender; T.Enabled:=true;
end;
procedure TRectangle.start(Sender: TObject);
begin
  px:=px+kx; py:=py+ky; Paint(S,round(px),round(py));
  if (trunc(px)=xn) then t.Enabled:=false;
end;
```

В обработчике FormCreate теперь описываем создание двух объектов – A и B, а в обработчике Image1MouseUp теперь будет содержаться строка

```
if Button=mbleft then A.Move(Form1.Image1,x,y) else B.Move(Form1.Image1,x,y);
```

Теперь можно посмотреть результат работы программы, и убедиться, что объекты будут перемещаться так, как мы и хотели. Однако можно заметить, что если один из объектов перемещается по другому, то он затирает его частично или полностью (в случае неподвижности последнего). Поэтому на одном из последующих этапов нужно предусмотреть возможность исключения такого недочета.

На следующих двух этапах предлагается организовать работу сначала с массивом объектов класса TRectangle, а затем перейти к использованию списка объектов. Для работы со списком объектов необходимо описать еще один класс – TlistRectangle, основанный на стандартном классе Delphi – Tlist. Это позволит наиболее удобным образом добавлять новые объекты, управлять ими, а также удалять при необходимости. Готовый пример, выполняющий указанные действия, написан автором, однако его полное представление здесь выходит за рамки объемов данной статьи.

В заключение отмечу, что одной из главных целей при изучении объектно-ориентированного программирования должно стать приобретение навыков конструктивного мышления, умения видеть различные варианты реализации задачи.

Список литературы

1. Дмитриев В.Л. Поэтапная разработка программы в среде Turbo Pascal на примере поиска пути с использованием волнового алгоритма // Информатика и образование. № 8, 2013. – С. 29-33.
2. Дмитриев В.Л. Развитие представлений об объектном программировании на примере разработки объектов в среде программирования Turbo Pascal // Информатика в школе. № 2 (95), 2014. – С. 54-59.
3. Дмитриев В.Л. Теория и практика программирования на С++. – Стерлитамак: РИО СФ БашГУ, 2013. – 308 с.
4. Поляков К.Ю., Еремин Е.А. Информатика. Углубленный уровень: учебник для 11 класса: в 2 ч., ч. 2. – М.: "БИНОМ. Лаборатория знаний", 2013. – 304 с.