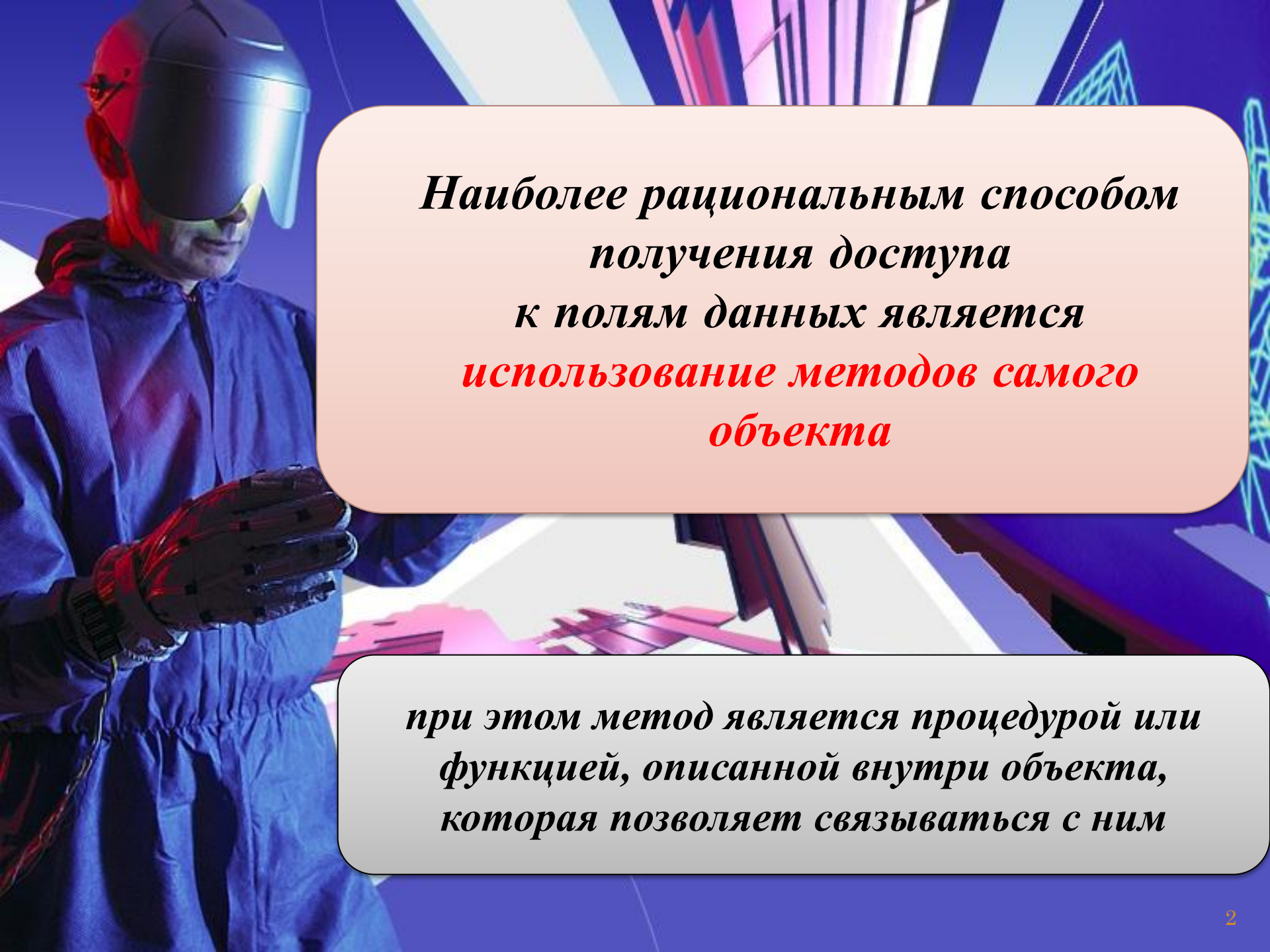


Методы



*Наиболее рациональным способом
получения доступа
к полям данных является
использование методов самого
объекта*

*при этом метод является процедурой или
функцией, описанной внутри объекта,
которая позволяет связываться с ним*

ИНИЦИАЛИЗАЦИЯ ПОЛЕЙ ОБЪЕКТОВ



Обычно при работе с записями возникает проблема инициализации полей записей.

Предположим, имеется следующая структура:

```
TPerson = OBJECT
    Name : STRING[30];
    Date  : STRING[10];
    Rate  : REAL;
end;
```

Начинающие программисты часто используют оператор **WITH** для присвоения полям *Name*, *Date* и *Rate* начальных значений:

```
VAR  
  MyPerson : TPerson;  
  
  WITH MyPerson DO  
    BEGIN  
      Name:='Иванов, Николай';  
      Date:='25-06-1995';  
      Rate:=40000;  
    END;
```

Инициализация полей объектов



Такое действие будет корректным, но не идеальным.

*При необходимости инициализировать более одной записи типа **TPerson** придется использовать большое число операторов **WITH**, которые будут выполнять одни и те же действия.*

Естественным решением проблемы является создание инициализирующей процедуры, которая обобщает применение оператора **WITH** к любому экземпляру типа *TPerson*, передаваемого в качестве параметра:

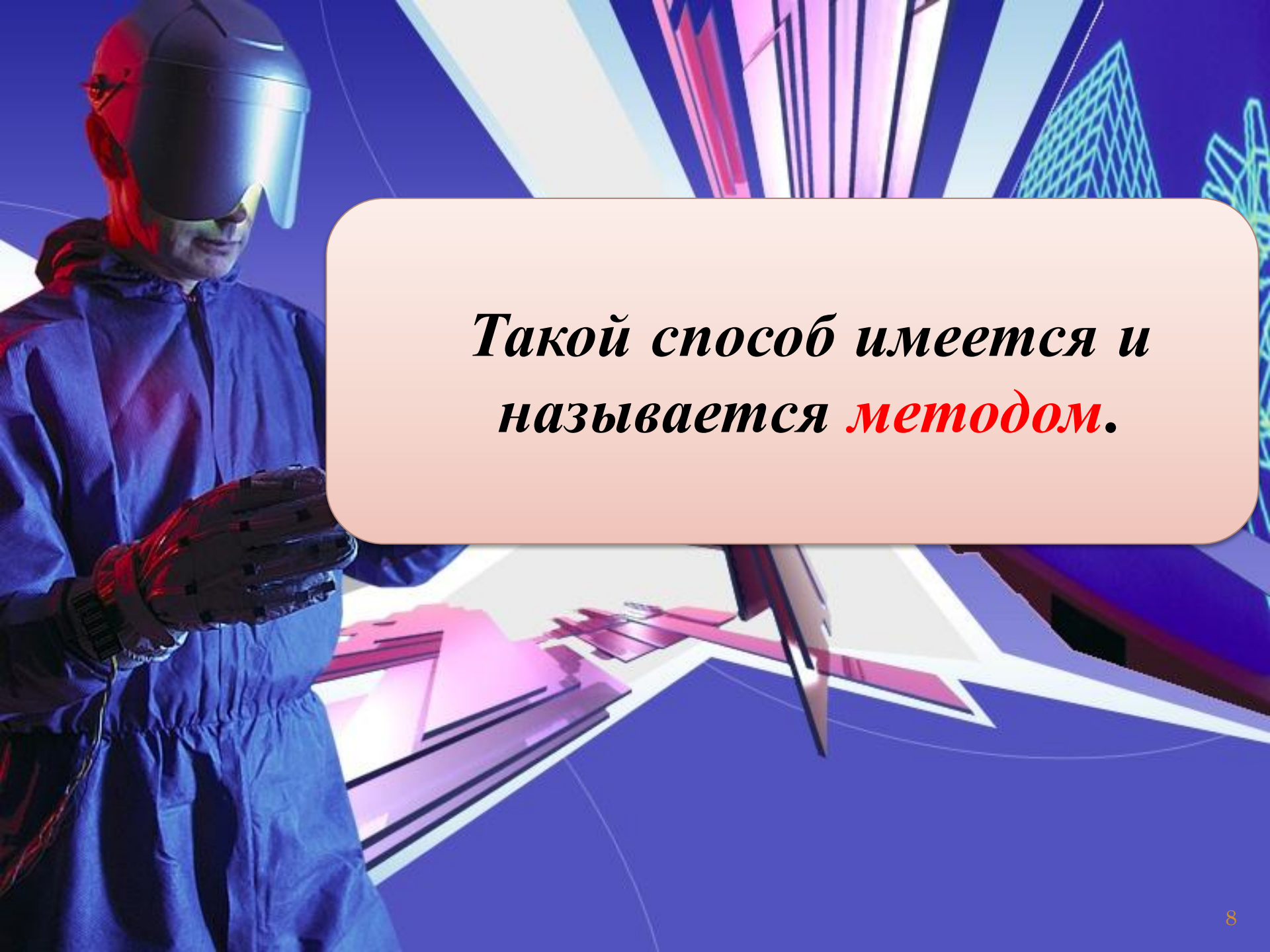
```
PROCEDURE Init(VAR Person:TPerson; Nm,Dt:STRING; Rt:REAL);  
BEGIN  
    WITH Person DO  
        BEGIN  
            Name:=Nm;  
            Date:=Dt;  
            Rate:=Rt;  
        END;  
END;
```

Инициализация полей объектов

ИНИЦИАЛИЗАЦИЯ ПОЛЕЙ ОБЪЕКТОВ

✓ Возникает вопрос, почему при разработке процедуры **Init** специально для обслуживания типа **TPerson** необходимо снова указывать, какой тип записи и какое его поле обрабатывает процедура **Init**?

✓ Нет ли какого-нибудь способа объединения типа записи и обслуживающего кода в одно целое?



*Такой способ имеется и
называется **методом**.*

Метод — это процедура или функция, объявленные внутри объявления элемента типа объект и имеющая доступ к полям данных объекта, не требуя передачи их ему в виде параметров.

- Объявление метода внутри объявления объектного типа содержит только заголовок.
- Тело метода определяется вне объявления объекта.
- Заголовок метода должен содержать имя объекта, которому принадлежит метод.

Поля и методы являются двумя составными частями новой структуры, называемой **объектом**.

TYPE

TPerson = OBJECT

Name : STRING[30];

Date : STRING[10];

Rate : REAL;

PROCEDURE Init(Nm,Dt:STRING; Rt:REAL);

END;

PROCEDURE TPerson.Init(Nm,Dt:STRING; Rt:REAL);

BEGIN

Name:=Nm; {Поле Name объекта TPerson }

Date:=Dt; {Поле Date объекта TPerson }

Rate:=Rt; {Поле Rate объекта TPerson }

END;

Методы

Теперь для инициализации экземпляра типа *TPerson* достаточно просто вызвать его метод.

```
VAR  
  Person : TPerson;  
Begin  
  Person.Init('Николай Иванов','25-06-1995',40000);  
END.
```


Процесс определения методов объектов напоминает модули Турбо Паскаля.

Внутри объекта метод определяется заголовком процедуры или функции, действующей как метод:

```
TYPE  
  TPerson = OBJECT  
    Name   : STRING[30];  
    Date   : STRING[10];  
    Rate   : REAL;  
    PROCEDURE Init(Nm,Dt:STRING; Rt:REAL);  
    FUNCTION GetName      : STRING;  
    FUNCTION GetDate     : STRING;  
    FUNCTION GetRate     : REAL;  
End;
```

Заметьте, что поля данных должны быть объявлены перед объявлением методов.

Определение методов.

- Подобно интерфейсной части модуля описание методов внутри объекта только указывает действия, но не определяет, каким образом они будут выполнены.
- Сами методы описываются вне определения объекта как отдельная процедура или функция.
- При определении метода его имени должно предшествовать имя типа объекта, которому принадлежит данный метод, с последующей точкой

Определение методов.

Например:

```
PROCEDURE TPerson.Init(Nm,Dt:STRING; Rt: REAL);
  BEGIN
    Name: =Nm;
    Date:=Dt;
    Rate:=Rt;
  END;
FUNCTION TPerson.GetName:  STRING;
  BEGIN
    GetName:=Name;
  END;
FUNCTION TPerson.GetDate:  STRING;
  BEGIN
    GetDate:=Date;
  END;
FUNCTION TPerson.GetRate:  REAL;
  BEGIN
    GetRate:=Rate;
  END;
```

В данной программе при необходимости можно определить уже существующую функцию, например *GetName*, не связывая ее с типом *TPerson*.

Определение методов.

Область действия метода и параметр Self

Относящийся к *TPerson* метод *GetName* приблизительно эквивалентен описанию:

```
FUNCTION TPerson.GetName (VAR Self: TPerson): STRING;  
BEGIN  
    GetName := Self.Name;  
END;
```

хотя такое описание не совсем корректно.

Фактически, параметр **Self** является как бы невидимым полем объекта типа *TPerson* (это относится к любому типу), доступ к которому осуществляется так же, как и к любому другому полю объекта.

Область действия метода и параметр Self

Обычно нет необходимости в использовании параметра **Self**

- т.к. генерируемый Турбо Паскалем код обрабатывает его автоматически.

Однако в некоторых ситуациях, когда, например, объекты разных типов имеют поля с совпадающими именами, можно использовать параметр **Self** явно.

Параметр **Self** является частью области стека при всех вызовах методов.

Методы, используемые как внешние, написанные на языке Ассемблера, должны учитывать этот параметр при получении доступа к параметрам метода в стеке.

Область действия метода и параметр **Self**



Скрытие данных в объектах



*Хороший стиль
программирования
требует,
чтобы доступ к полям
объекта
осуществлялся только
через методы, работающие
с данными полями.*

СКРЫТИЕ ДАННЫХ В ОБЪЕКТАХ



Иногда при использовании объектов внутри модулей могут встретиться части описаний объектов, которые экспортировать нежелательно, например в коммерческих приложениях.

- Необходимо предусмотреть объекты, методы которых доступны, но непосредственный доступ к данным объекта запрещен.*
- В Турбо Паскале для этих целей используются скрытые (частные) поля и методы.*

- Скрытые поля и методы доступны только внутри того модуля, в котором описан объект.
- Поля и методы, которые следуют непосредственно за заголовком объектного типа или директивой **PUBLIC**, не имеют никаких ограничений на область действий.
- Поля и методы, объявленные после директивы **PRIVATE**, считаются частными (скрытыми) и ограничены использованием в пределах модуля.

Скрытие данных в объектах

Полное описание объекта будет выглядеть следующим образом:

```
TYPE  
  NewObject = ОБЪЕКТ(родитель)  
    поля;      {общедоступные}  
    методы;    {общедоступные}  
  PRIVATE  
    поля;      {частные}  
    методы;    {частные}  
  PUBLIC  
    поля;      {общедоступные}  
    методы;    {общедоступные}  
END;
```

Скрытие данных в объектах

Добавление методов косвенного обращения к полям типа *TStudent* несколько увеличивает объем результирующего кода, однако развитый компоновщик Турбо Паскаля отбрасывает код метода, который ни разу не вызывается в программе.

- Поэтому не следует сомневаться в том, включать или не включать в объект метод, который может быть, как использован, так и не использован в программе, в которой задействован данный тип объекта.

Неиспользуемые методы не влияют ни на быстродействие программы, ни на размер EXE-файла: если они не используются в программе, то они просто не включаются в него.

Имеется возможность полностью скрыть данные объекта типа *TStudent* от внешнего доступа.

- Если вне объекта его внутреннее представление неизвестно, то можно изменять части этого представления, не изменяя при этом логику работы программы, до тех пор, **пока заголовок метода остается неизменным.**