

Объекты классов и полиморфизм

Наумов Д.А., доц. каф. КТ

Основы программной инженерии, 2019

Содержание лекции

- 1 Статическое и динамическое связывание типов
- 2 Таблица виртуальных функций
- 3 Чисто виртуальные функции и абстрактные классы
- 4 Приведение типов

Связывание

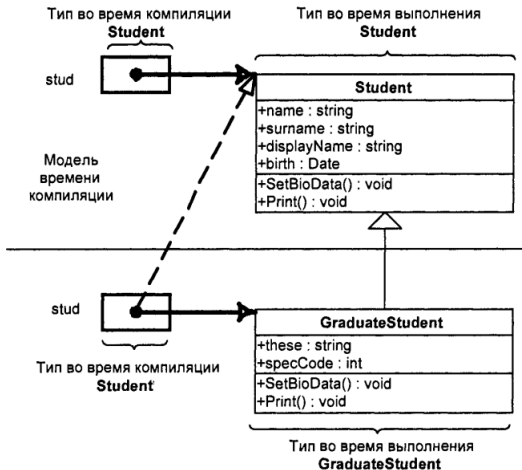
привязка тела функции к месту ее вызова.

В процедурных языках задача связывания решается:

- на этапе компиляции (если определение функции находится в том же модуле, что и обращение к ней);
- на этапе компоновки (если определение функции находится в другом модуле).

В объектно-ориентированных языках переменная-указатель на объект базового типа может в действительности быть связана с объектом производного типа, что делает в общем случае невозможным установление действительного экземпляра метода, который нужно вызвать, в момент трансляции или компоновки.

Статическое связывание - установление типов во время компиляции



Статическое связывание - установление типов во время компиляции

```

procedure CallPrint(s: TStudent);
begin
  s.Print();
end;

var
  s: TStudent;
  gs1, gs2: TGraduateStudent;

  parr: array[1..3] of TStudent;
  i: integer;

begin
  s := TStudent.Create;
  s.SetBioData('Ivan', 'Petrov', 1, 1, 1980);

  gs1 := TGraduateStudent.Create;
  gs1.SetBioData('Oleg', 'Sidorov', 2, 2, 1981,
    'Space objects detection', '09.04.01');

  gs2 := TGraduateStudent.Create;
  gs2.SetBioData('Alex', 'Ivanov', 3, 3, 1982,
    'Image classification', '09.04.02');

  writeln;

  parr[1] := s;
  parr[2] := gs1;
  parr[3] := gs2;

  for i := 1 to 3 do
    CallPrint(parr[i]); //вызывается процедура TStudent.Print
  end;
end.

```

Статическое связывание

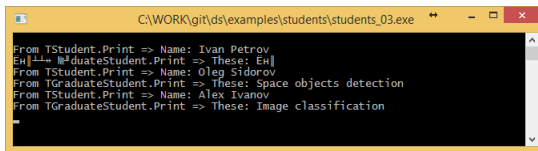
- Во время компиляции невозможно установить, на объект какого типа будет указывать *stud* во время выполнения: на объект типа *student* или на объект производного типа (например, *GraduateStudent*);
- Гарантируется, что объект, передаваемый функции *CaliPrint()* в качестве аргумента, может быть безопасно и корректно преобразован к типу *TStudent*
- Модель времени компиляции в этом случае является моделью времени выполнения.
- Даже если в действительности *stud* связан с объектом типа *GraduateStudent*, в соответствии с механизмом статического связывания вызываться всегда будет версия функции *Print*, объявленная в классе *TStudent*

Статическое связывание

Можно попытаться «уговорить» компилятор, воспользовавшись явным преобразованием типа:

```
procedure CallPrint(s: TStudent);  
begin  
    TGraduateStudent(s).Print();  
end;
```

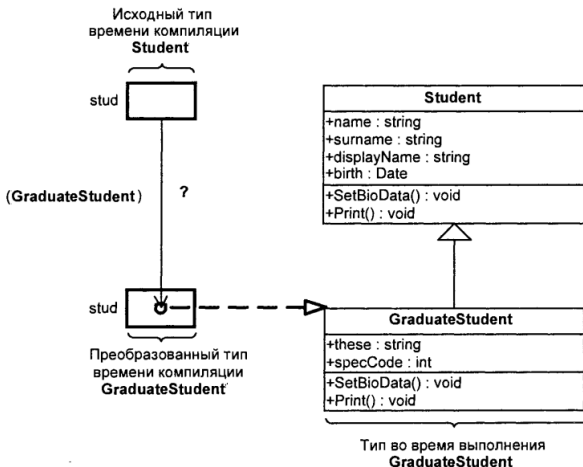
Но для переменной типа TStudent результат будет непредсказуем:



```
C:\WORK\git\ds\examples\students\students_03.exe  
From TStudent.Print => Name: Ivan Petrov  
From TGraduateStudent.Print => These: Eh|  
From TStudent.Print => Name: Oleg Sidorov  
From TGraduateStudent.Print => These: Space objects detection  
From TStudent.Print => Name: Alex Ivanov  
From TGraduateStudent.Print => These: Image classification
```

Статическое связывание

Небезопасное преобразование типа:



Динамическое связывание - установление типов во время выполнения

- Механизм реализации динамического связывания основан на использовании виртуальных (перезаписываемых, замещающих) функций.
- Для того чтобы объявить для некоторого метода необходимость реализовать применительно к этому методу модель позднего связывания, используется ключевое слово `virtual`.

```
type
  //дата
  TDate = record Day, Month, Year: integer; end;

  //сведения о студенте
  TStudent = class
  public
    Name,                //имя
    Surname: string[80]; //фамилия
    DisplayName: string[255]; //имя для печати
    Birth: TDate;         //дата рождения

    procedure SetBioData(AName, ASurname: string; ADay, AMonth, AYear: integer);

    procedure Print(): virtual;
  end;
```

Динамическое связывание - установление типов во время выполнения

- Для того чтобы метод оставался виртуальным в классе-потомке, используется ключевое слово `override`.

```
//сведения об аспиранте
TGraduateStudent = class(TStudent)
public
    //только для аспиранта
    These: string[255];      //тема диссертации
    SpecCode: string[10];    //код специальности

    procedure SetBioData(
        AName, ASurname: string;
        ADay, AMonth, AYear: integer;
        AThese: string;
        ASpecCode: string);

    procedure Print(); override;
end;
```

Динамическое связывание - установление типов во время выполнения

```
procedure CallPrint(s: TStudent);
begin
    s.Print();
end;

var
    s: TStudent;
    gs1, gs2: TGraduateStudent;

    parr: array[1..3] of TStudent;
    i: integer;

begin
    s := TStudent.Create;
    s.SetBioData('Ivan', 'Petrov', 1, 1, 1980);

    gs1 := TGraduateStudent.Create;
    gs1.SetBioData('Oleg', 'Sidorov', 2, 2, 1981,
        'Space objects detection', '09.04.01');

    gs2 := TGraduateStudent.Create;
    gs2.SetBioData('Alex', 'Ivanov', 3, 3, 1982,
        'Image classification', '09.04.02');

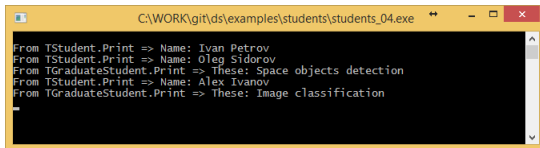
    writeln;

    parr[1] := s;
    parr[2] := gs1;
    parr[3] := gs2;

    for i := 1 to 3 do
        CallPrint(parr[i]); //вызывается процедура TStudent.Print
    end;

    readln;
end.
```

Динамическое связывание - установление типов во время выполнения

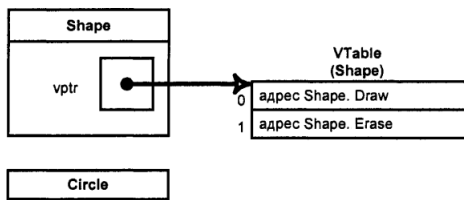


A screenshot of a Windows command prompt window. The title bar shows the file path "C:\WORK\git\ds\examples\students\students_04.exe". The window contains the following text output:

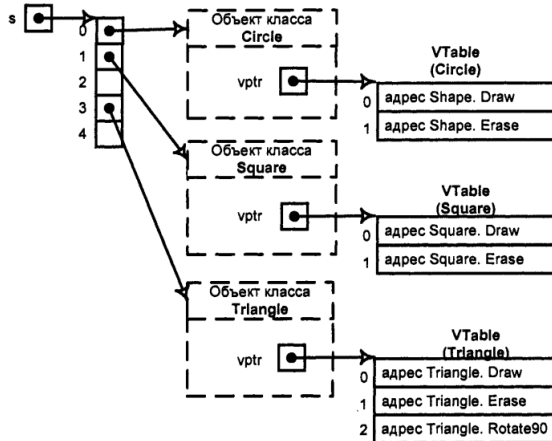
```
From TStudent.Print => Name: Ivan Petrov  
From TStudent.Print => Name: Oleg Sidorov  
From TGraduateStudent.Print => These: Space objects detection  
From TStudent.Print => Name: Alex Ivanov  
From TGraduateStudent.Print => These: Image classification  
-
```

Таблица виртуальных функций

- для каждого класса создается специальная таблица, содержащая адреса виртуальных функций, объявленных в этом классе;
- в определение класса добавляется скрытый член класса — указатель на таблицу виртуальных функций.
- адреса одноименных виртуальных функций разных классов, находящихся в отношении наследования, помещаются в ячейки таблицы VTable с одинаковыми индексами.
- если какая-либо виртуальная функция не перезаписывается в производном классе, в таблицу виртуальных функций помещается адрес соответствующей функции базового класса.



Выбор правильной виртуальной функции



Чисто виртуальные функции и абстрактные классы

- В ряде случаев уровень абстрагирования, представляемый базовым классом, не подразумевает какую-либо практическую реализацию некоторых (или всех) методов класса.
- В этом случае базовый класс рассматривается как выразитель некоторой абстрактной концепции, предоставляя не реализацию методов, а общий интерфейс, подходящий для работы с элементами такого типа.

```
type
  TShape = class
    procedure Show; virtual; abstract;
    procedure Hide; virtual; abstract;
  end;
```

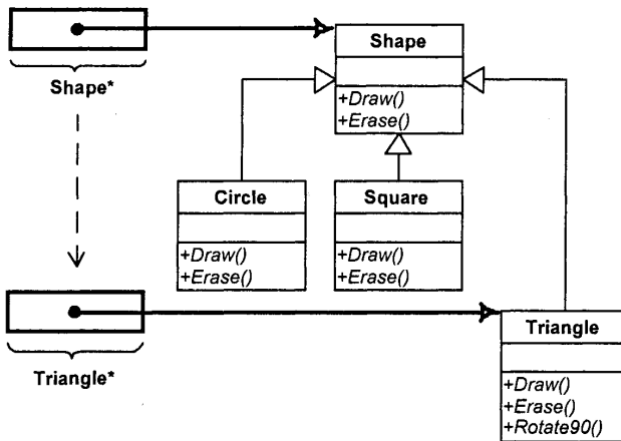
Абстрактный класс

Класс, содержащий хотя бы одну чисто виртуальную функцию.

Объекты абстрактного класса создавать нельзя.

Приведение типов

Использование **механизма приведения типов** позволяет осуществить преобразование указателя на базовый тип к указателю на производный тип.



Приведение типов

```
for i := 1 to 10 do  
begin  
  f[i].Show;  
  
  if f[i] is TTriangle then  
    (f[i] as TTriangle).Rotate90;  
end;
```

В отличие от повышающего преобразования, попытка понижающего преобразования может закончиться неудачей, поэтому требуется проверка.

