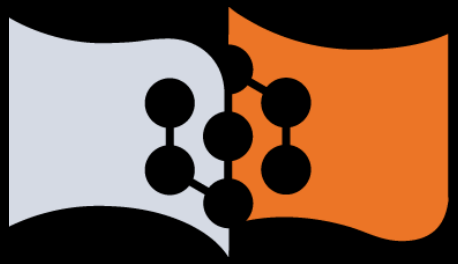


СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ
SIBERIAN FEDERAL UNIVERSITY

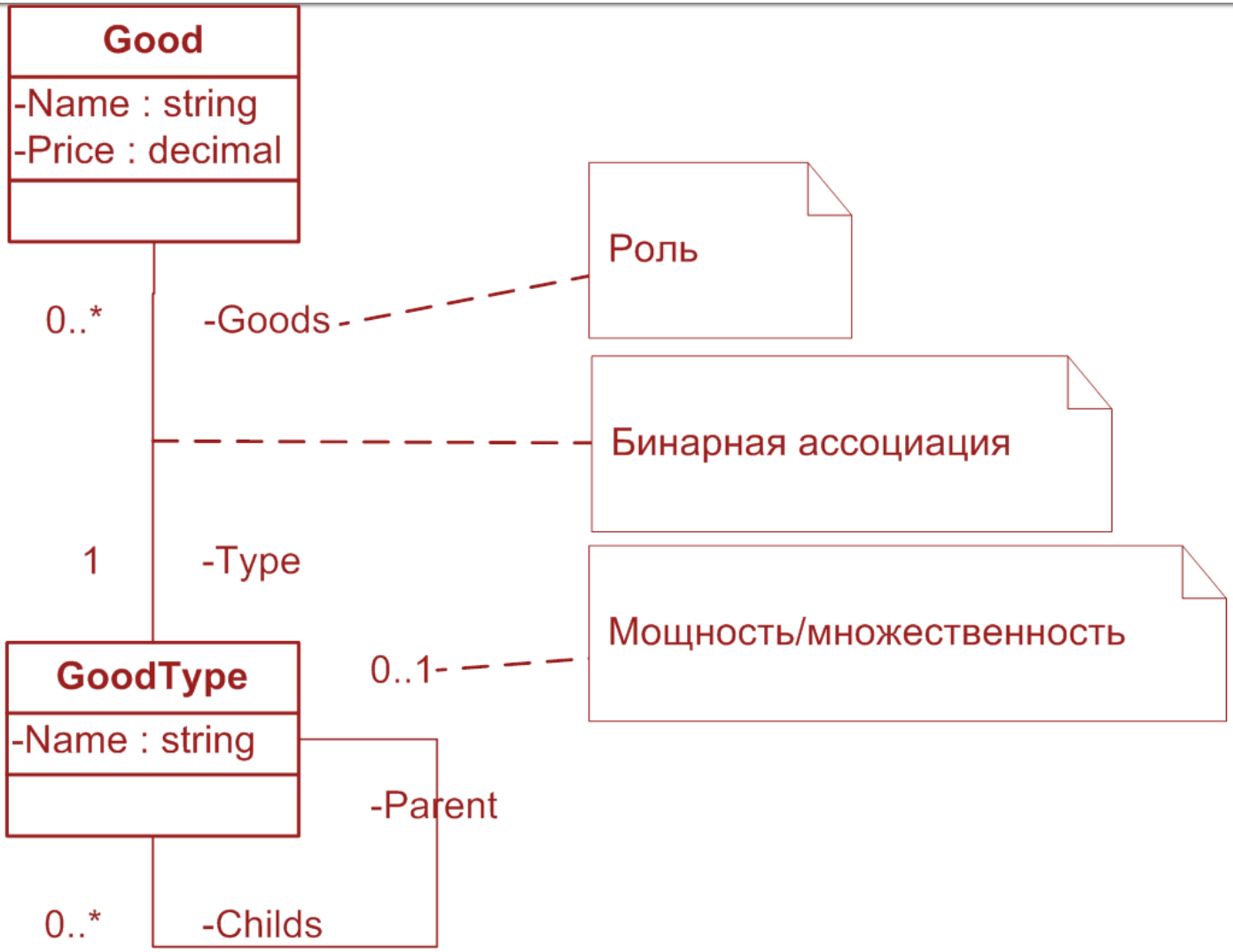
Институт космических и информационных технологий

Объектно-ориентированное программирование

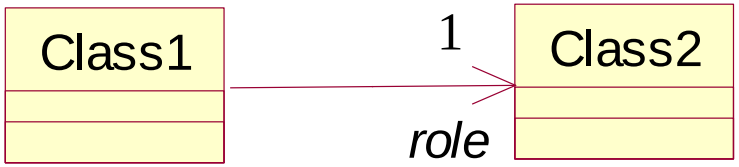
Ассоциации. Полиморфные объекты

к.т.н. Якунин Юрий Юрьевич

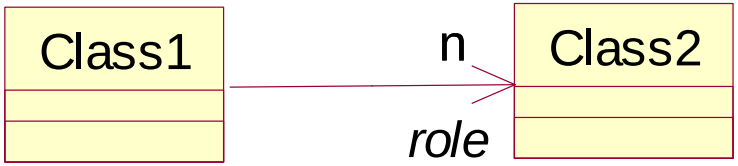
Ассоциации



Однонаправленные ассоциации

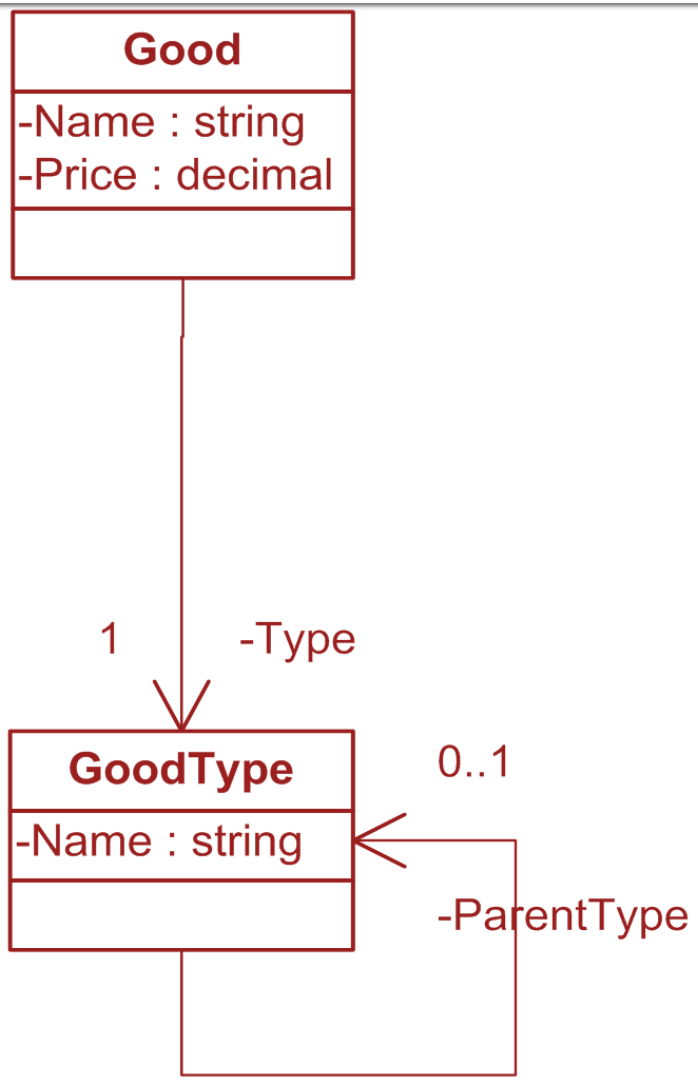



```
type
  Class1 = class
    private
    role : Class2;
end;
```




```
type
  Class1 = class
    private
    role : TObjectList;
end;
```

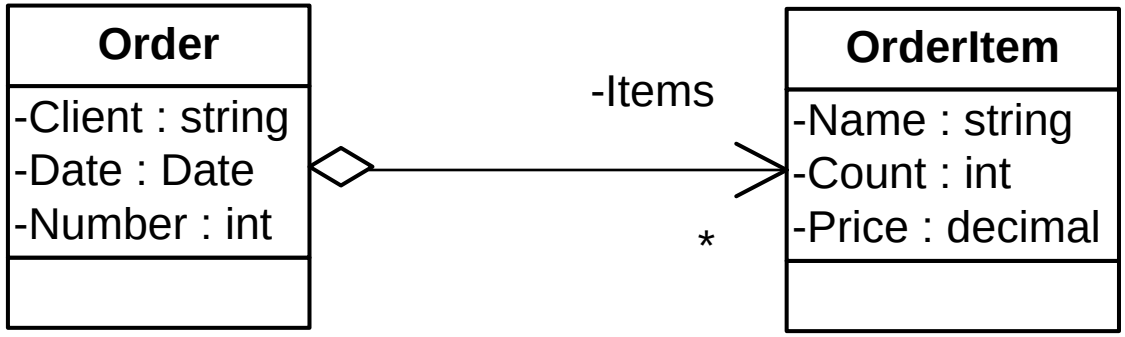
Самоассоциация



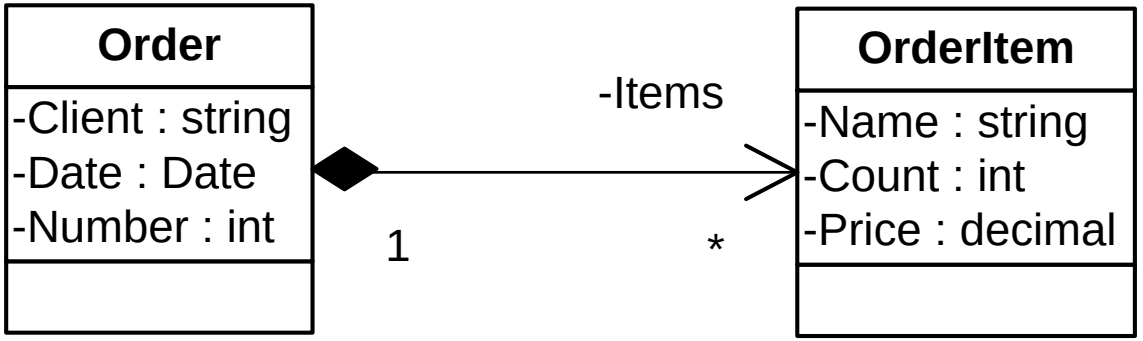
Пример данных для GoodType

- Тип продукции
  - Бытовая техника
    - Компьютерная техника
    - Холодильники
  - Продовольственная продукция
    - Кисломолочная продукция
      - Молоко
      - …
    - ..
  - ..

Агрегация и композиция



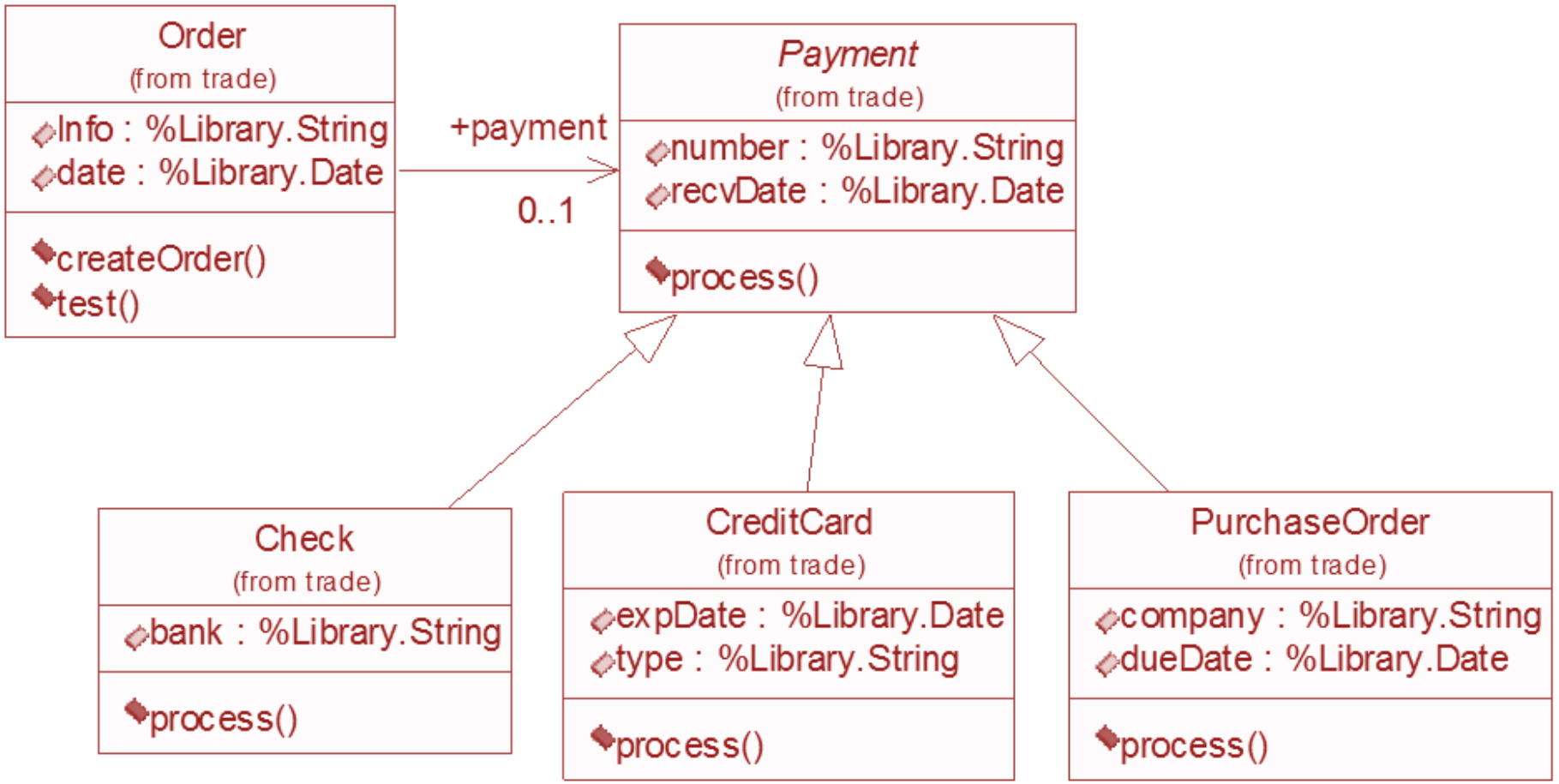
- Агрегация – ассоциация, показывающая отношение часть – целое



- Композиция – усиленная агрегация, где агрегат отвечает за создание и уничтожение своих частей в процессе жизненного цикла объекта

Наследование и полиморфизм

Payment – абстрактный класс



Методы и свойства класса. Интерфейс

```
interface
uses System.Contnrs;
Type
  TPerson = class abstract
  private
    …
    class var FPersons: TObjectList;
    class function GetListOfPersons(Index: Integer): TPerson;
static;
    class procedure SetListOfPersons(Index: Integer; const
Value: TPerson); static;
  public
    …
    class property ListOfPersons[Index: Integer]: TPerson read
GetListOfPersons write SetListOfPersons;
    class function getInfoAll(): String;
  end;
```

Методы и свойства класса. Реализация

```
implementation
uses System.SysUtils;
{ TPerson }
class function TPerson.getInfoAll: String;
var
  str: String;
  I: Integer;
begin
  if (FPersons = nil) then Exit('Общее количество людей в списке составляет: 0 человек.');

  str := 'Общее количество людей в списке составляет: ' + IntToStr(FPersons.Count) + 'чел.!
';
  for I := 0 to FPersons.Count-1 do
    str := str + IntToStr(I+1) + ') ' + ListOfPersons[I].FIO + '; ';
  result := str;
end;
class function TPerson.GetListOfPersons(Index: Integer): TPerson;
begin
  if (FPersons = nil) then Exit;
  result := TPerson(FPersons.Items[Index]);
end;
class procedure TPerson.SetListOfPersons(Index: Integer; const Value: TPerson);
begin
  if (FPersons = nil) then FPersons := TObjectList.Create;
  FPersons.Insert(Index, Value);
end;
end.
```

Методы и свойства класса. Обращение (вызов)

```
procedure TSDIAppForm.btOrdClassMethodClick(Sender: TObject);
var
  str: String;
  per: TPerson;
begin
  per := TPerson.Create('Кулибин Иван', Now());
  per.ListOfPersons[0] := per; //Обращение к свойству класса
через объект
  per := TPerson.Create('Иванов Александр', Now());
  //Обращение к свойству класса через имя класса
  TPerson.ListOfPersons[1] := per;
  str := TPerson.getInfoAll; //Вызов метода класса через имя
класса
  ShowMessage(str);
end;
```

Статические методы и свойства класса

```
class function GetListOfPersons(Index: Integer): TPerson; static;
```

- Доступны только через имя класса

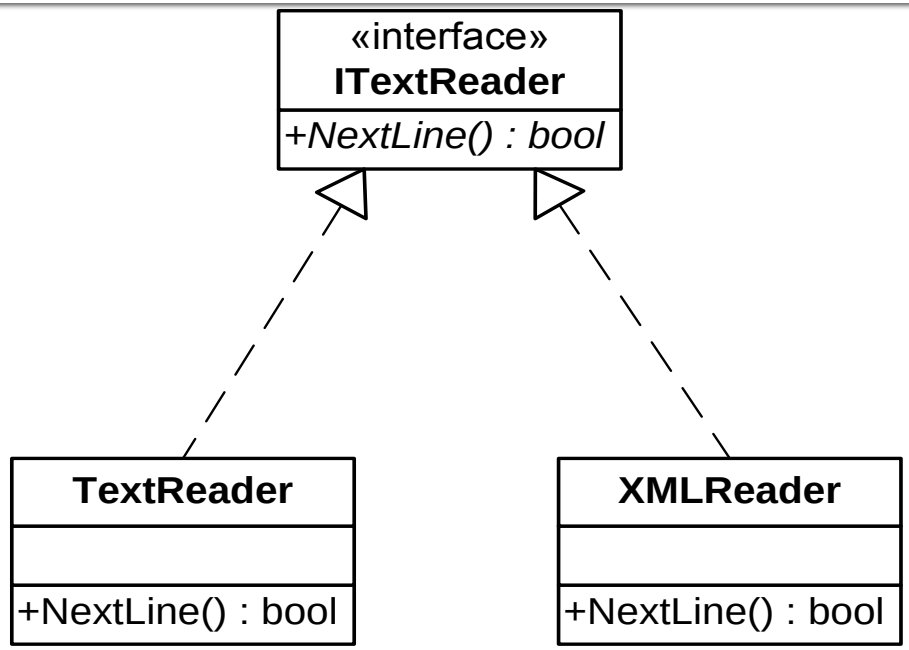
Интерфейс

Интерфейс = класс – реализация

- не может содержать конструкторы и деструкторы
- все атрибуты интерфейса являются общедоступными (public)
- все методы интерфейса являются абстрактными (virtual, abstract)

```
type
  ITextReader = interface(IInterface)
    // Методы
    function NextLine: Boolean;
    ...
    // Свойства
    property Active: Boolean;
    property ItemCount: Integer;
    property Items[Index: Integer]: string;
    property EndOfFile: Boolean;
  end;
```

Реализация интерфейса




```
type
  TTextReader = class(TInterfacedObject, ITextReader)
    ...
  end;

  TIteratableStringList = class(TInterfacedObject, ITextReader)
    ...
  end;
```

Работа с интерфейсом

```
interface
type
  TText = class
    protected
      // интерфейсная переменная
      source: ITextReader;
    public
      procedure getText(sr:Integer);
  end;

 implementation
 procedure TText.getText(sr:Integer);
 begin
   case sr of
     0: source := TTextReader.Create();
     1: source := TIteratableStringList.Create();
   end;
   source.Active := True;
   source.NextLine;
 end;
```