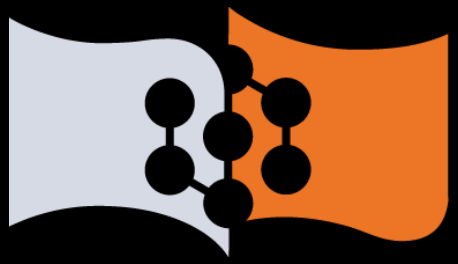


СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ
SIBERIAN FEDERAL UNIVERSITY

Институт космических и информационных технологий

Объектно-ориентированное программирование

# Введение. Понятия объекта и класса

к.т.н. Якунин Юрий Юрьевич

# Задачи курса

- Развить навыки объектного моделирования программных систем
- Познакомиться с унифицированным языком моделирования (UML) объектно-ориентированных программных систем
- Научиться читать и создавать диаграммы классов, а также понимать связь с программным кодом
- Научиться применять шаблоны проектирования

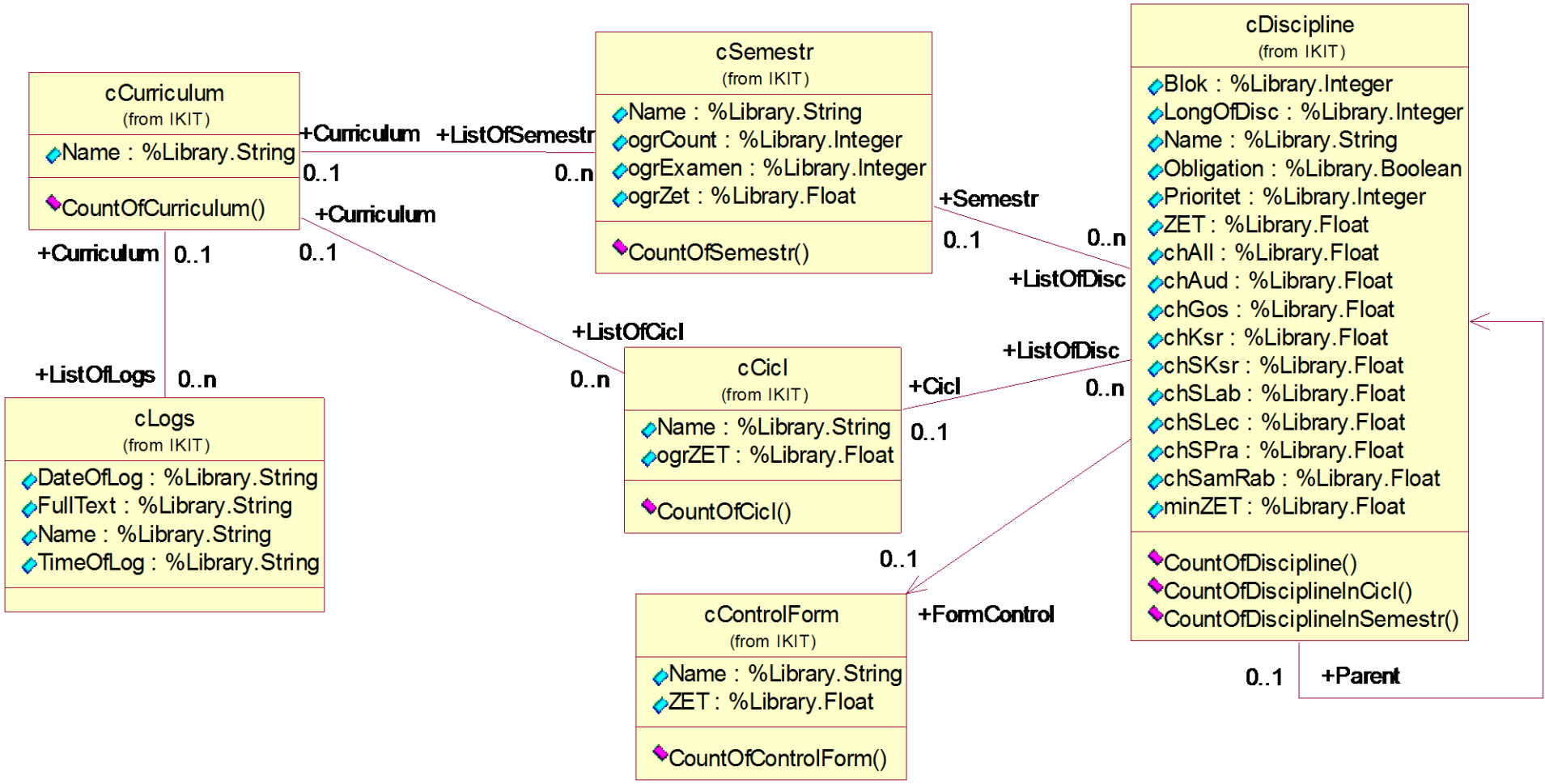
# Эволюция методов

- Хаотичное программирование
  - Данные и функции плохо структурированы
- Структурное программирование
  - Декомпозиция на самостоятельные модули
  - Неограниченный доступ функций к глобальным данным
  - Разделение данных и функций плохо отображает картину реально-го мира
- Объектно-ориентированное программирование
  - Инкапсуляция, наследование, полиморфизм
  - Unified model language (UML)

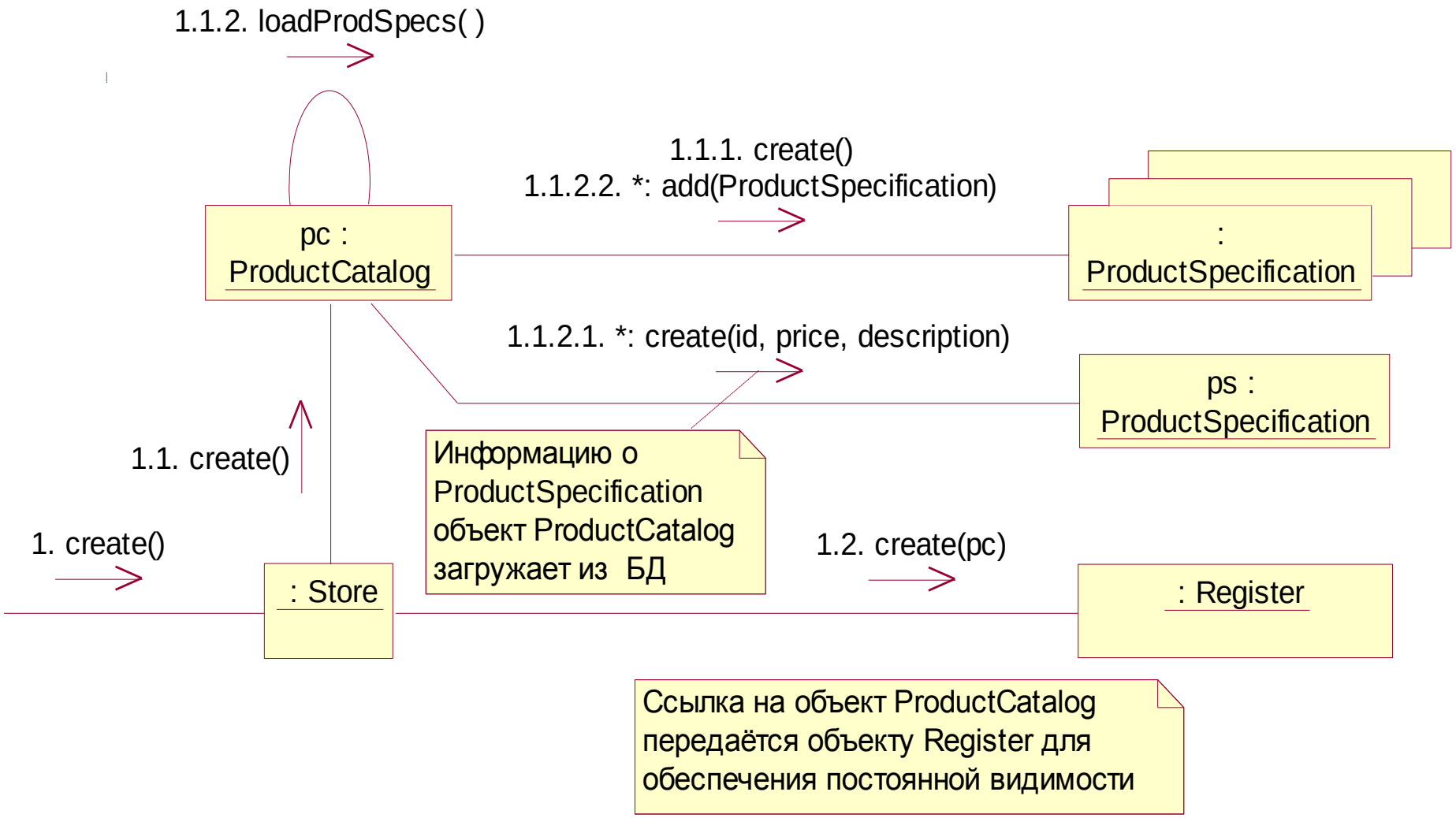
# Модель программной системы

| | | |
|---|---|---|
| **Динамическое** | Диаграмма: последовательностей, кооперации, деятельности, состояний и переходов, вариантов использования | |
| **Статическое** | Диаграмма: классов, компонент | Диаграмма развёртывания |
| *Представле-ния* | **Логическое** | **Физическое** |

# Пример диаграммы классов

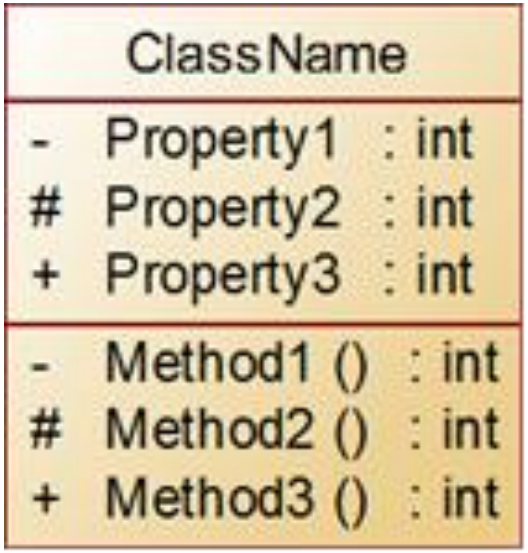


# Пример диаграммы кооперации

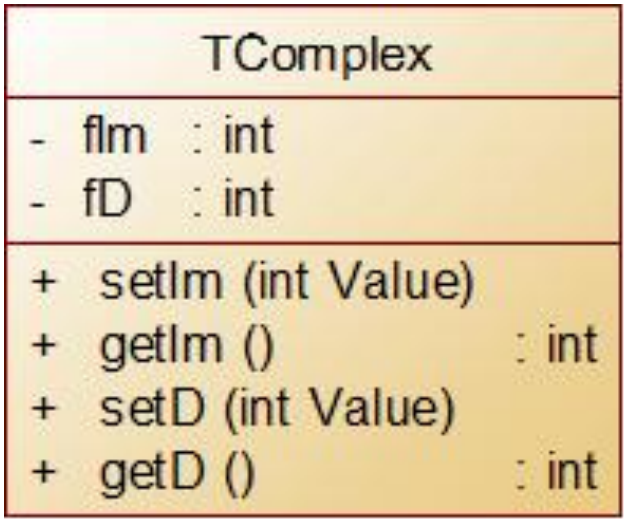


# Классы и объекты

- **Объект** представляет собой конкретный опознаваемый предмет, единицу или сущность (реальную или абстрактную), имеющую чётко определённое функциональное назначение в данной предметной области
- Объект характеризуется
  - Состоянием
  - Поведением
- **Класс** описывает группу объектов с одинаковыми свойствами, одинаковым поведением, типами отношений и семантикой
- Уровни защищённости элементов класса:
  - public
  - private
  - protected



# Объявление класса




```
unit uComplex;

interface
type
 TComplex = class
 private
  fIm, fD: Integer;
  function GetIm: Integer;
  procedure SetIm(const Value: Integer);
  function GetD: Integer;
  procedure SetD(const Value: Integer);

 public
  property Im: Integer read GetIm write SetIm;
  property D: Integer read GetD write SetD;
  constructor Create(); overload;
  constructor Create(vIm,vD:Integer); overload;
 end;
```

# Реализация класса

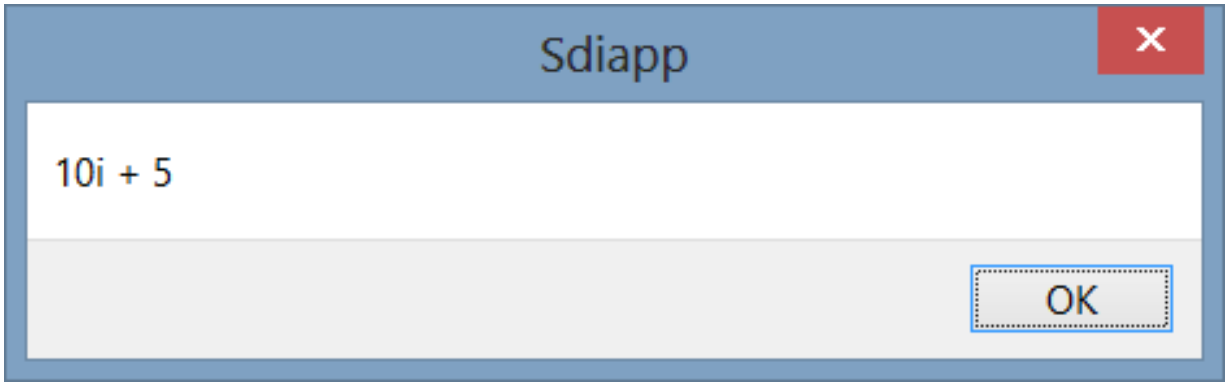
```
Implementation

constructor TComplex.Create(vIm, vD:
Integer);
begin
  fIm := vIm;
  fD := vD;
end;

function TComplex.GetIm: Integer;
begin
  result := fIm;
end;
procedure TComplex.SetIm(const Value:
Integer);
begin
  fIm := Value;
end;
end.
```

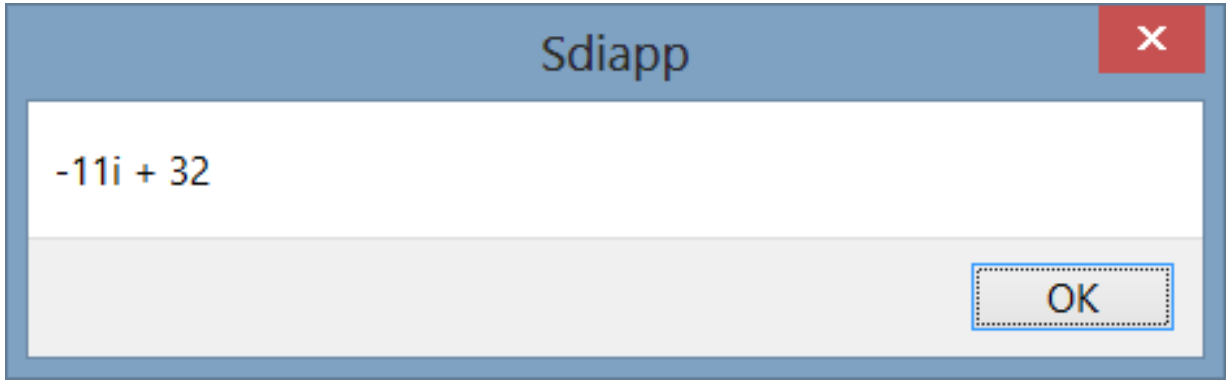
# Создание объекта

```
procedure TSDIAppForm.btCreateObjectClick(Sender: TObject);
var
  cmplx: TComplex; //Объявляем переменную
begin
  //Создаем объект cmplx класса TComplex посредствам метода класса Create
  cmplx := TComplex.Create;
  //Задаем значения свойств объекта (определяем состояние объекта)
  cmplx.Im := 10;
  cmplx.D := 5;
  //Выводим информацию об объекте в оконное сообщение
  ShowMessage(IntToStr(cmplx.Im) + 'i + ' + IntToStr(cmplx.D));
end;
```



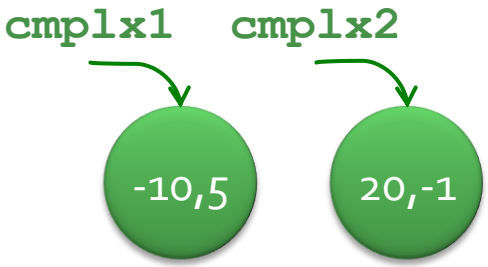
# Создание объекта

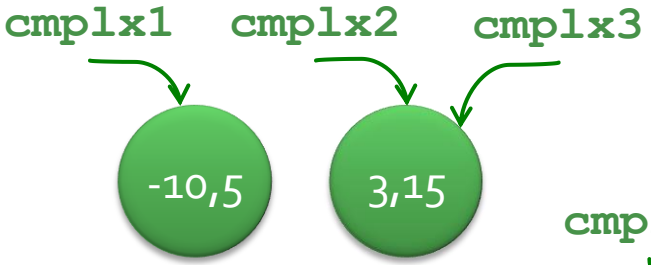
```
procedure TSDIAppForm.btCreateObjByConstrClick(Sender: TObject);
var
  cmplx: TComplex;
begin
  cmplx := TComplex.Create(-11,32);
  ShowMessage(IntToStr(cmplx.Im) + 'i + ' + IntToStr(cmplx.D));
end;
```

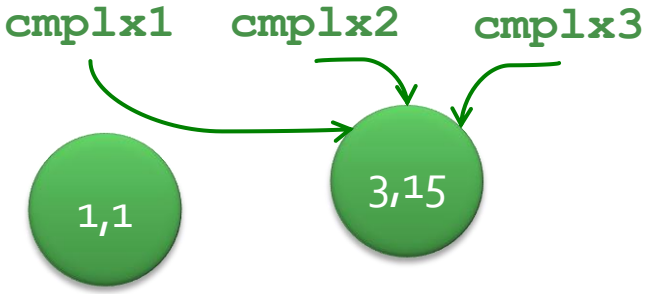


# Ссылки на объекты

```
procedure TComplexSet.Add;
var
   cmlx1, cmlx2, cmlx3 : TComplex;
begin
//А
   cmplx1 := TComplex.Create(-10, 5);
   cmplx2 := TComplex.Create(20, -1);
//Б
   cmplx3 := cmplx2;
   cmplx3.Im := 3;
   cmplx3.D := 15;
//В
   Cmplx1.Im := 1;
   Cmplx1.D := 1;
   cmplx1 := cmplx3;
end;
```



# Передача параметров в метод

- **Передача параметров по значению**
  - procedure Test(s: string);
- **Передача параметров по ссылке**
  - procedure ChangeMe(**var** x: longint);
- **Передача возвращаемого значения**
  - procedure ParamReturn(**out** x: Integer);
  - // function FuncAnalog : Integer;
- **Передача параметров констант**
  - procedure Test(**const** s: string );
- **Значения параметров по умолчанию**
  - procedure HasDefVal( s: string; i: integer = 0 );
- **Передача открытых массивов**
  - function AddEmUp(A: array of integer): integer;