Множественное наследование

Александр Смаль

Академический университет
20 декабря 2013
Санкт-Петербург

Множественное наследование

Множественное наследование (multiple inheritance) — возможность наследовать сразу несколько классов.

```
struct Student {
    string name()       const { return name_; }
    string university() const { return university_; }
private:
    string name_, university_;
};

struct FullTimeEmployee {
    string name()    const { return name_; }
    string company() const { return company_; }
private:
    string name_, company_;
};

struct BadStudent: Student, FullTimeEmployee {
    string name() const { return Student::name(); }
};
```

Интерфейсы

Во многих языках множественное наследование заменяется возможностью реализовывать интерфейсы.

```
struct Person {
    string name() const { return name_; }
private:
    string name_;
};
struct IStudent {
    virtual string name()       const = 0;
    virtual string university() const = 0;
    virtual ~IStudent() {}
};
struct IFullTimeEmployee {
    virtual string name()    const = 0;
    virtual string company() const = 0;
    virtual ~IFullTimeEmployee() {}
};
struct BadStudent: Person, IStudent, IFullTimeEmployee {
    string name()       const { return Person::name(); }
    string university() const { return university_; }
    string company()    const { return company_; }
private:
    string university_, company_;
};
```

Проблема с переопределением одинаковых функций

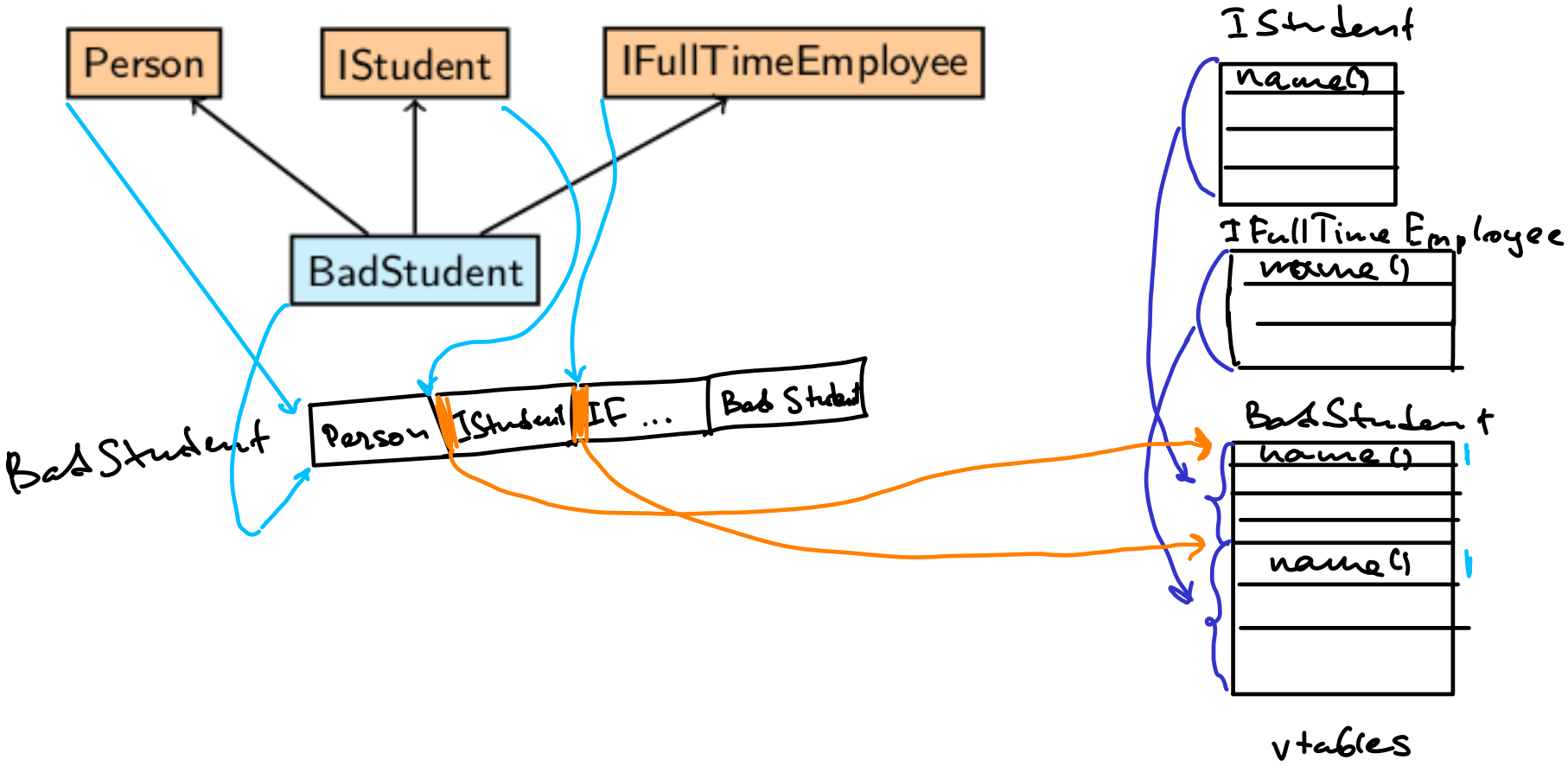
```
struct IStudent {
    virtual string name()         const = 0;
    virtual string university() const = 0;
    virtual ~IStudent() {}
};

struct IPlayer {
    virtual string name()         const = 0;
    virtual ~IPlayer() {}
};

struct TypicalStudent: Person, IStudent, IPlayer {
    ...
    string name()          const { return Person::name(); }
    ...
};
```

Проблема с переопределением одинаковых функций

```
struct IStudent {
    virtual string name()         const = 0;
    virtual string university()   const = 0;
    virtual ~IStudent() {}
};
struct IStudentX : IStudent {
    string name() const { return studentName(); }
    virtual string studentName() const = 0;
};

struct IPlayer {
    virtual string name()         const = 0;
    virtual ~IPlayer() {}
};
struct IPlayerX : IPlayer {
    string name() const { return playerName(); }
    virtual string playerName() const = 0;
};

struct TypicalStudent: Person, IStudentX, IPlayerX {
    ...
    string studentName() const { return Person::name(); }
    string playerName() const  { return "DarkEvil666"; }
    ...
};
```

Представление в памяти

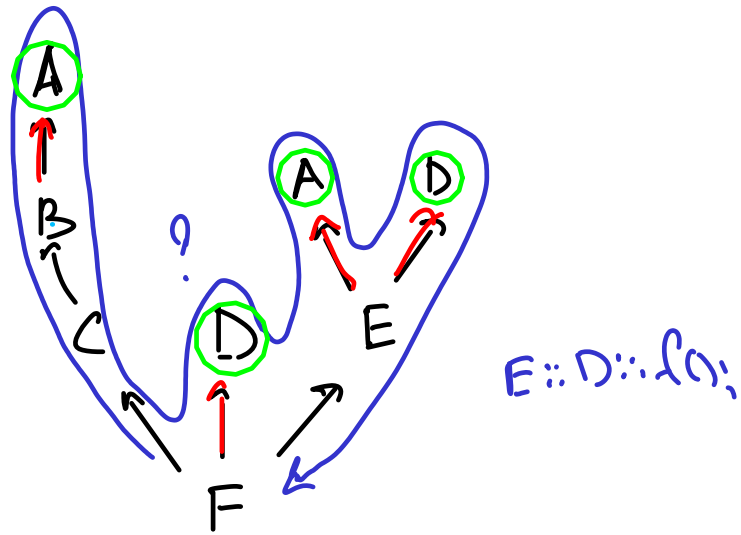
Во многих языках множественное наследование заменяется возможностью реализовывать интерфейсы.



Важно: помните про преобразование указателей.

Создание и удаление объекта

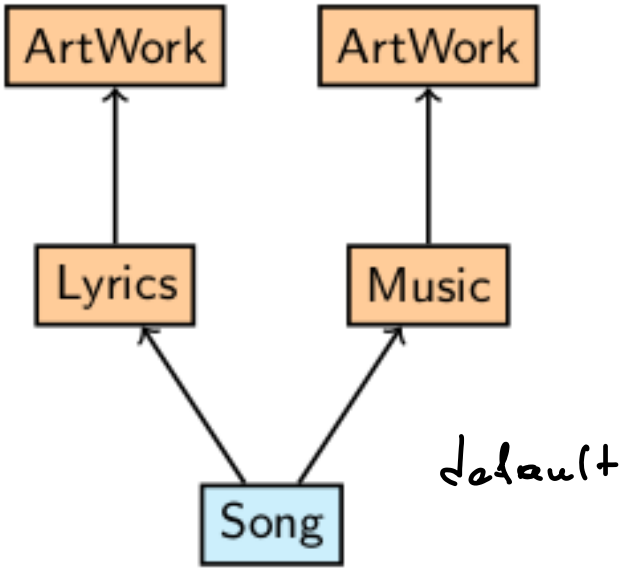
```
struct A { };

struct B : A { };

struct C : B { };

struct D { };

struct E : A, D { };

struct F : C, D, E { };
```

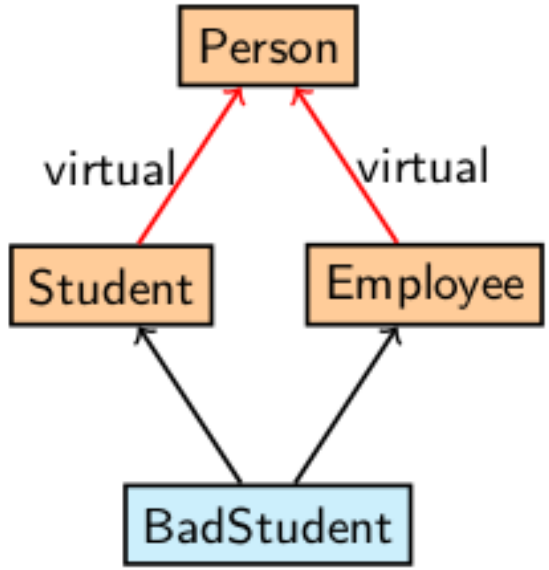


Конструкторы вызываются в порядке A, B, C, D, A, D, E, F.
Деструкторы вызываются в обратном порядке.
Проблемы:

1. Дублирование A и D.
2. Недоступность первого D.

Виртуальное наследование






```
struct Person {};
struct Student   : virtual Person {};
struct Employee  : virtual Person {};
struct BadStudent : Student, Employee {};
```

Виртуальное наследование: вопросы

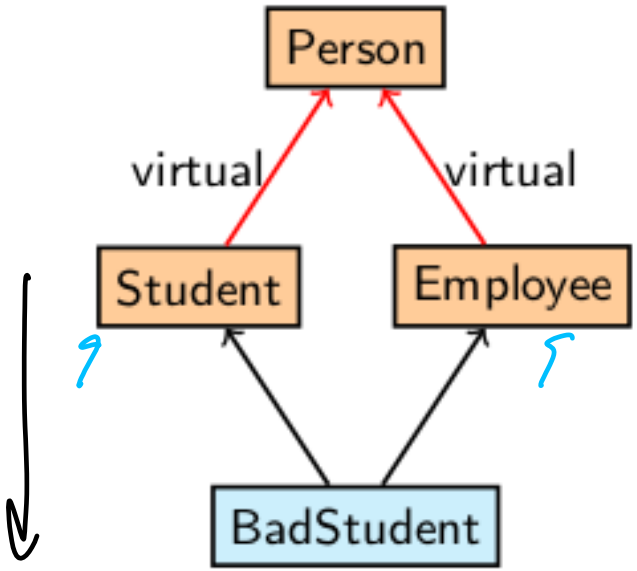
Кто вызывает конструктор базового класса?

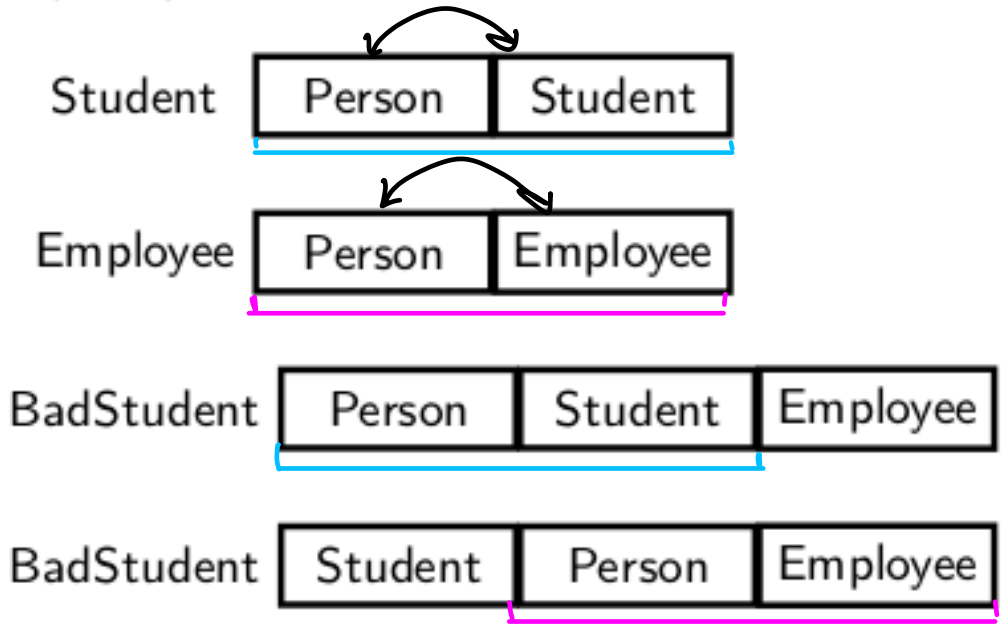
```
struct Person {
    explicit Person(string const& name)
        : name_(name)
    {}
private:
    string name_;
};
struct Student  : virtual Person {
    explicit Student(string const& name)
        : Person(name)
    {}
};
struct Employee : virtual Person {
    explicit Employee(string const& name)
        : Person(name)
    {}
};
struct BadStudent : Student, Employee {
    explicit BadStudent(string const& name)
        : Person(name)
        , Student(name)
        , Employee(name)
    {}
};
```

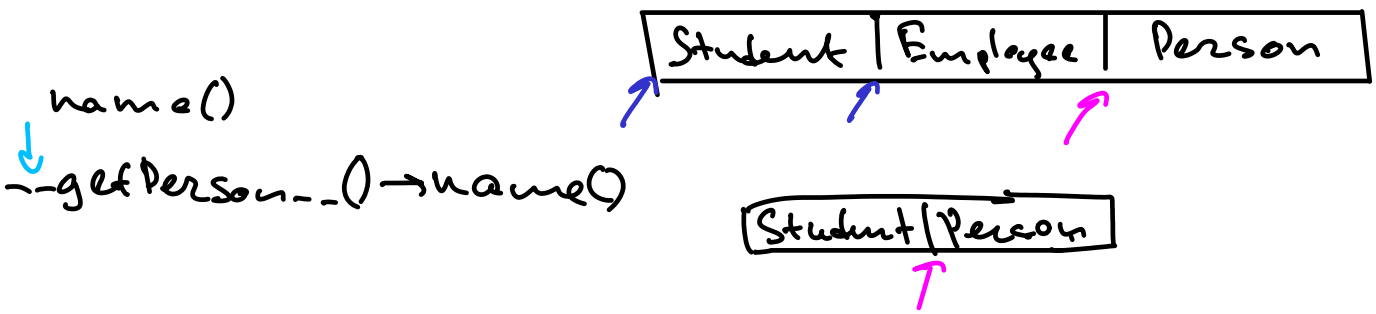
"Vasya"
"Mike"
"Victor"

Виртуальное наследование: вопросы

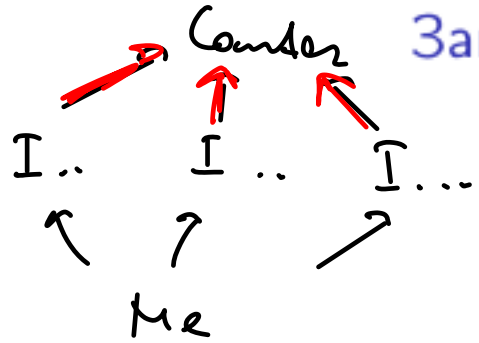
Как устроено расположение в памяти?







Заключение



1. Не используйте множественное наследование для наследования реализации.
2. Используйте интерфейсы.
3. Хорошо подумайте перед тем, как использовать виртуальное наследование.
4. Помните о неприятностях, связанных с множественным наследованием.
5. Помните о неприятностях, связанных с виртуальным наследование.