Тема 7. Реализация ООП

Парадигмы программирования. C#

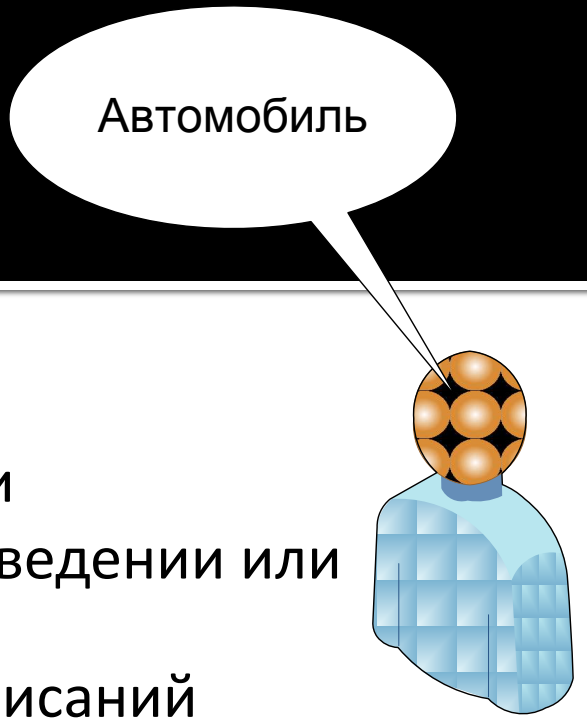
План

- Классы и объекты
- Инкапсуляция
- ООП в C#
- Практика применения ООП

Классы и объекты

Часть 1

Что такое класс?



- С точки зрения философа
  - Артефакт человеческой классификации
  - Классификация основана на общем поведении или атрибутах
  - Соглашение о соответствии именам описаний используется для обозначения класса
  - Мы создаём словари, общаемся, думаем с использованием классов
- С точки зрения программиста
  - Именованная синтаксическая конструкция, описывающая общее поведение и атрибуты
  - Структура данных, включающая данные и функции их обработки

Что такое объект?

- Объект – это реализация класса
- Объекты выражают
  - Уникальность. Объекты отличаются один от другого
  - Поведение. Объекты выполняют задания
  - Состояние. Объекты хранят информацию

Сравнение классов и структур

- Структура – это проект значения
  - Нет уникальности, состояние доступно, нет поведения

```
struct Time
{
    public int hour;
    public int minute;
}
```

- Класс – это проект объекта
  - Присутствует уникальность, состояние недоступно, есть поведение

```
class BankAccount
{
    // Class
    // contents
}
```

Абстракция

- Абстракция – это селектированное незнание
  - Определите, что важно, а что не важно
  - Фокусируйтесь и устанавливайте зависимости от того, что важно
  - Игнорируйте и не устанавливайте зависимости от того, что не важно
  - Используйте инкапсуляцию для укрепления абстракции

Совершенство достигается не тогда, когда уже нечего прибавить, но когда уже ничего нельзя отнять.

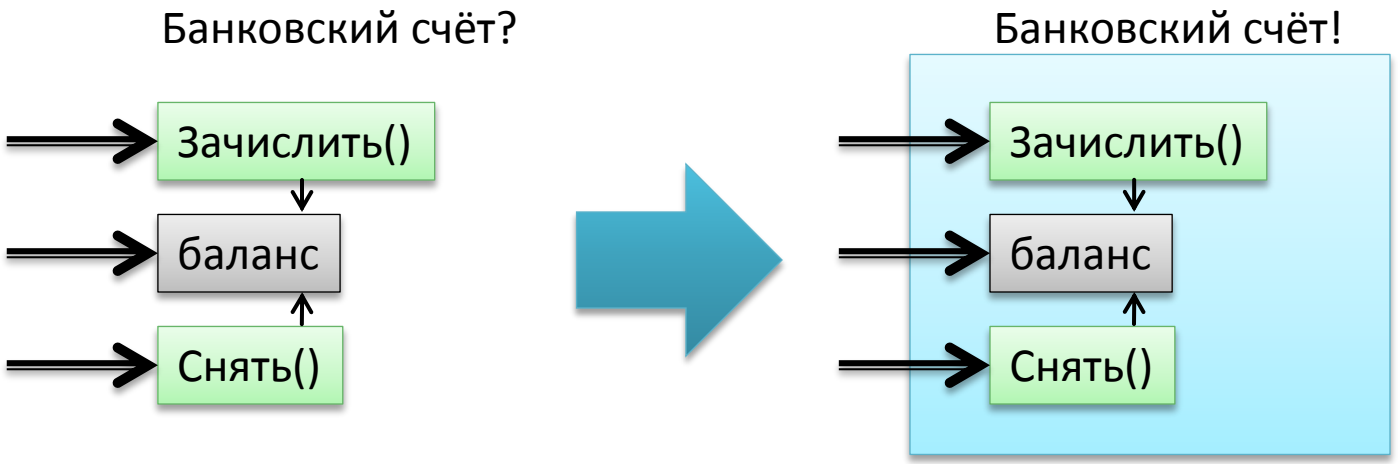
Антуан де Сент-Экзюпери «Ветер, песок и звезды»;

Инкапсуляция

Часть 2

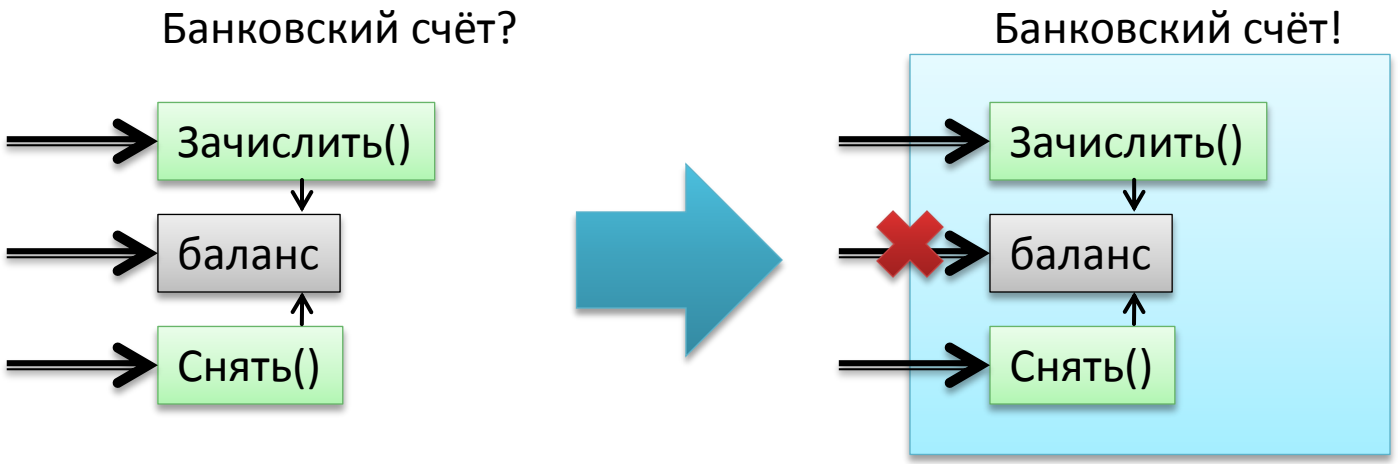
Объединение данных и методов

- Объединение данных и методов в единой капсуле
- Границы капсулы формируют внутреннее устройство и внешний мир



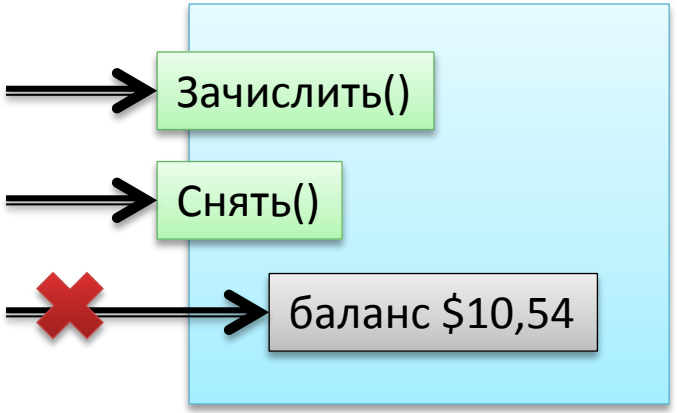
Контроль доступа

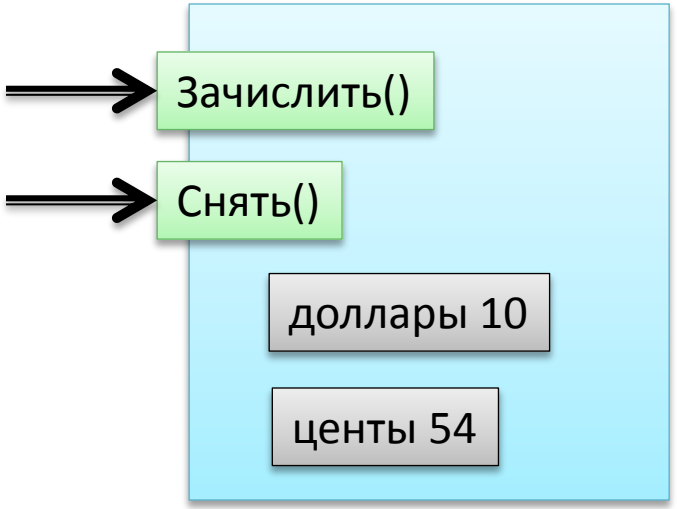
- Методы являются публичными (public), доступными для внутреннего устройства и внешнего мира
- Данные являются частными (private), доступными только для внутреннего устройства



Зачем инкапсулировать?

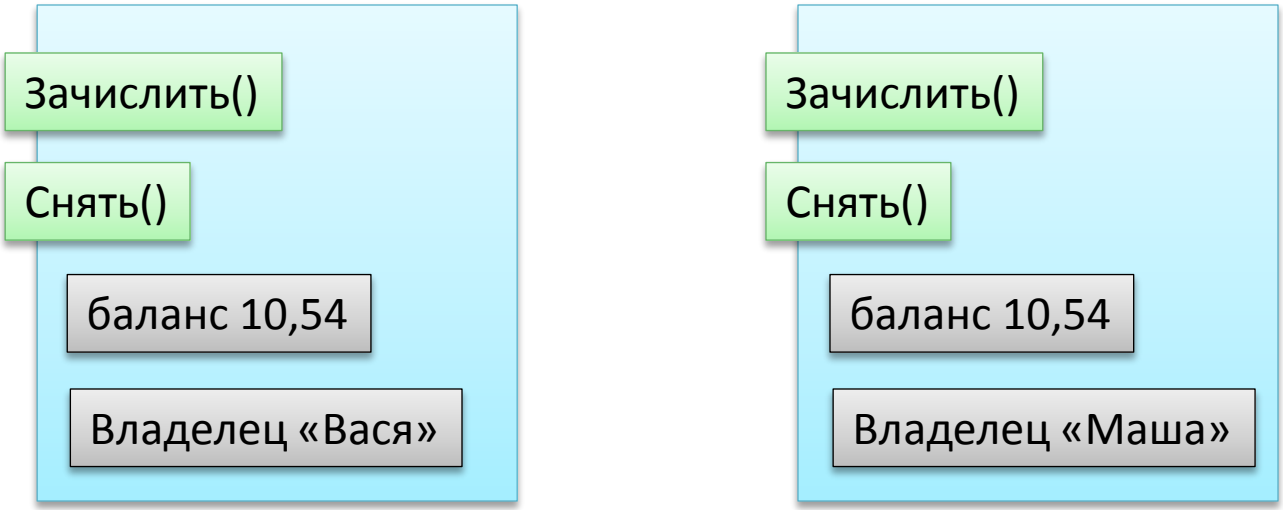
- Для контроля
  - Доступ к объекту осуществляется только через публичные методы
- Для изменения
  - Изменение представления данных не влияет на работу с объектом, если данные частные





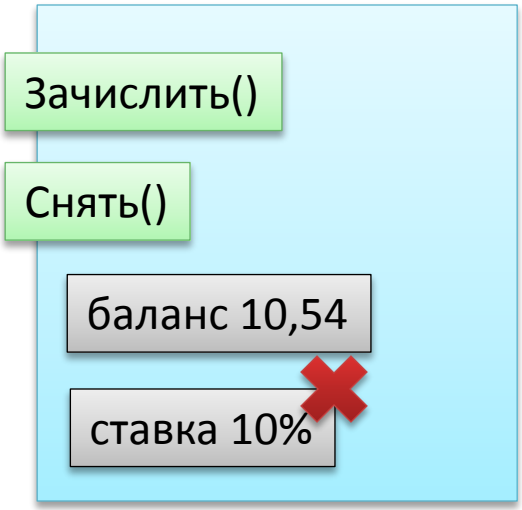
Данные объектов

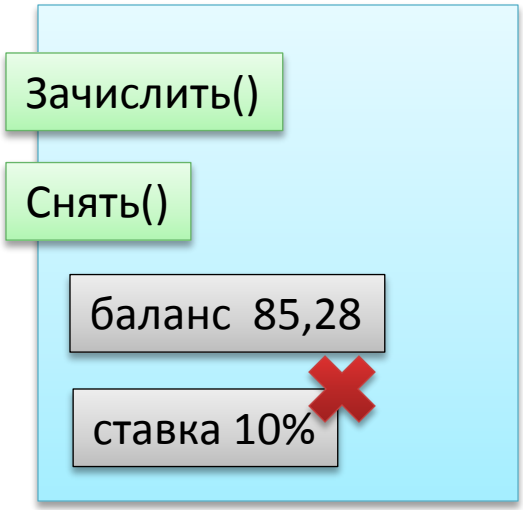
- Данные объектов описывают информацию индивидуальных объектов
- Пример
  - Каждый банковский счёт имеет свой баланс
  - Если два банковских счёта имеют одинаковый баланс, то это просто совпадение



Использование статических данных

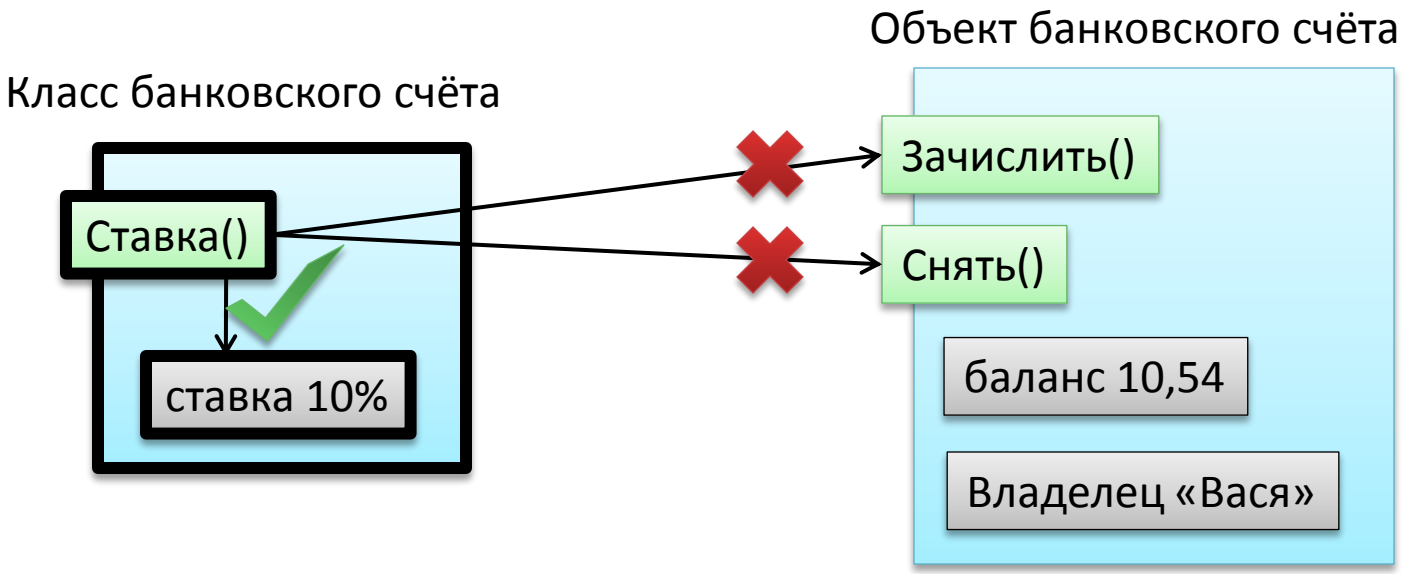
- Статические данные описывают информацию для всех объектов класса
- Пример
  - Все банковские счета разделяют общую процентную ставку
  - Почему хранение процентной ставки в каждом банковском счёте нецелесообразно?





Использование статических методов

- Статические методы имеют доступ только к статическим данным
- Статические методы вызываются в классе, а не в объекте



ООП в C#

Часть 3

Пересмотр простейшей программы

```
using System;
class Hello
{
    public static int Main()
    {
        Console.WriteLine("Hello, World");
        return 0;
    }
}
```

Программа с несколькими методами Main

```
// TwoEntries.cs
using System;
class EntranceOne
{
    public static void Main()
    {
        Console.Write("EntranceOne.Main( )");
    }
}
class EntranceTwo
{
    public static void Main()
    {
        Console.Write("EntranceTwo.Main( )");
    }
}
// End of file
c:\> csc /main:EntranceOne TwoEntries.cs
c:\> twoentries.exe
EntranceOne.Main( )
c:\> csc /main:EntranceTwo TwoEntries.cs
c:\> twoentries.exe
EntranceTwo.Main( )
c:\>
```

Определение простых классов

- Данные и методы объединяются внутри класса
- Методы публичные, данные частные

```
class BankAccount
{
    public void Withdraw(decimal amount)
    {
        ...
    }
    public void Deposit(decimal amount)
    {
        ...
    }
    private decimal balance;
    private string name;
}
```

Создание новых объектов

- Определение переменной класса не приводит к созданию объекта
  - Для создания объекта следует использовать оператор new

```
class Program
{
    static void Main()
    {
        Time now;
        now.hour = 11;
        BankAccount yours = new BankAccount();
        yours.Deposit(999999M);
    }
}
```

Использование ключевого слова this

- Ключевое слово this является ссылкой на объект, используемой для обращения к его членам
- Удобно использовать для одинаковых переменных из разных областей видимости

```
class BankAccount
{
    ...
    public void SetName(string name)
    {
        this.name = name;
    }
    private string name;
}
```

Использование вложенных классов

- Классы могут быть вложены в другие классы

```
class Program
{
    static void Main()
    {
        Bank.Account yours = new Bank.Account();
    }
}
class Bank
{
    ... class Account { ... }
}
```

Доступ к вложенным классам

- Вложенные классы могут быть объявлены, как публичные или частные

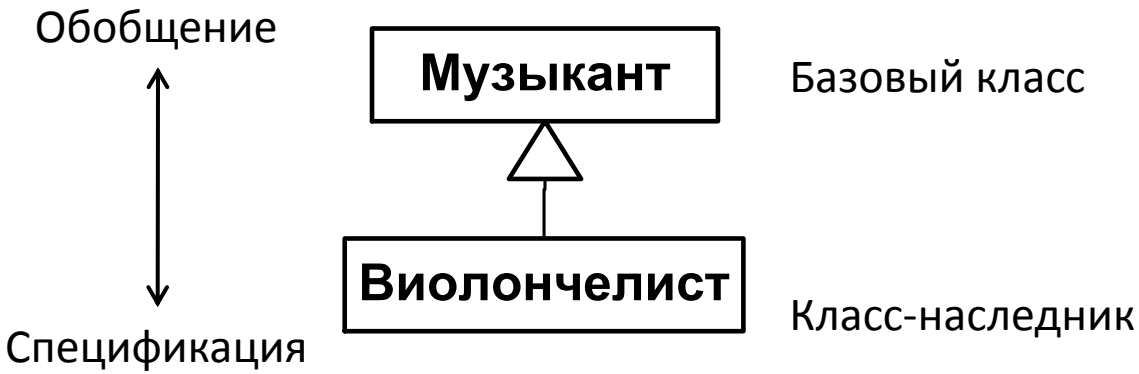
```
class Bank
{
    public class Account { ... }
    private class AccountNumberGenerator { ... }
}
class Program
{
    static void Main()
    {
        Bank.Account accessible;
        Bank.AccountNumberGenerator inaccessible;
    }
}
```

Практика применения ООП

Часть 4

Наследование

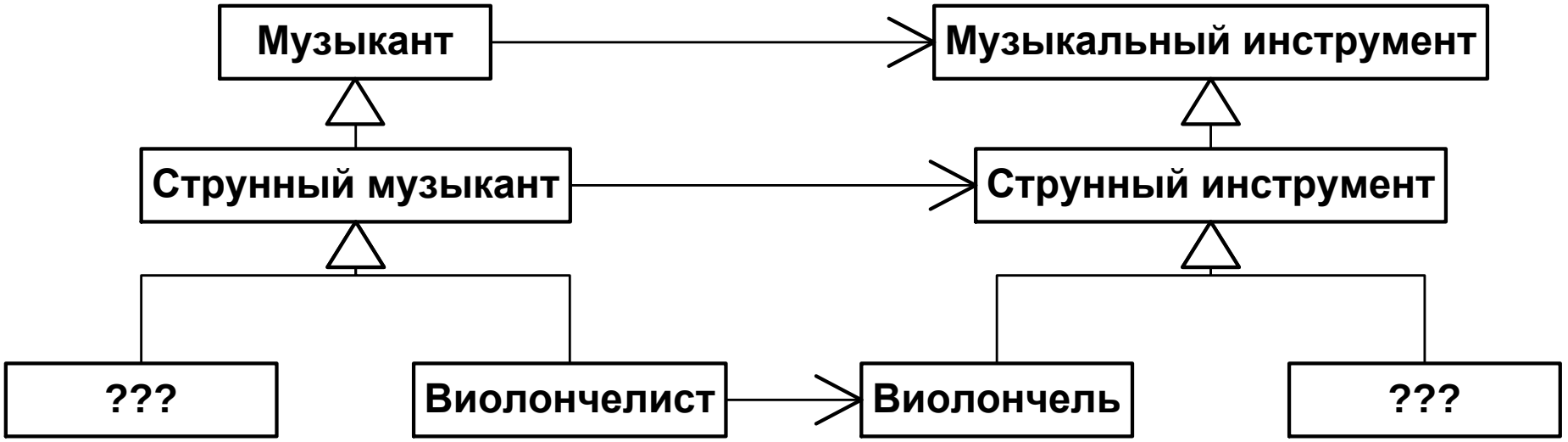
- Наследование определяет связь «является видом чего-либо»
- Наследование указывает на родство классов
- Классы-наследники конкретизируют классы-предки



Эквивалентны ли биологическое наследование и наследование в ООП?

Иерархии классов

- Классы связываются наследованием в иерархии классов

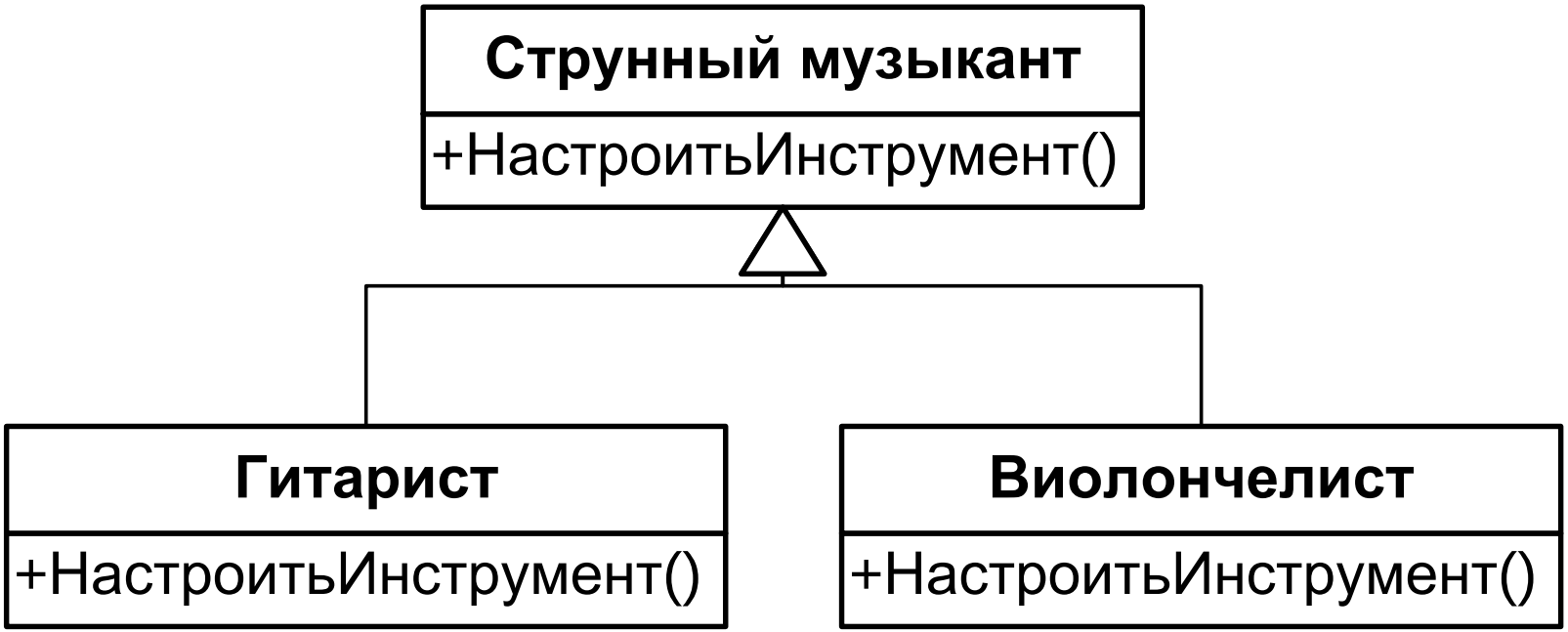


Единичное и множественное наследование

- Единичное наследование: класс наследуется от одного базового класса
  - Используется в C#
- Множественное наследование: класс наследуется от двух и более базовых классов
  - Запрещено в C#

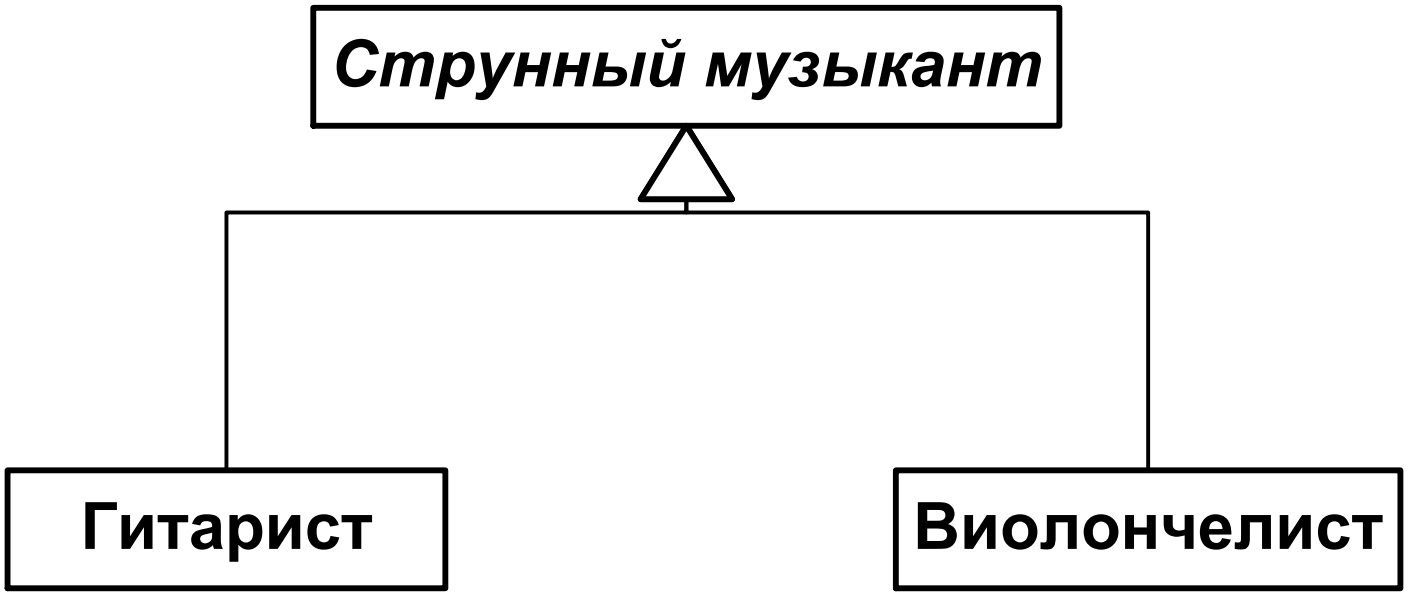
Полиморфизм

- Имя метода присутствует в базовом классе
- Реализации метода присутствуют в классах-наследниках



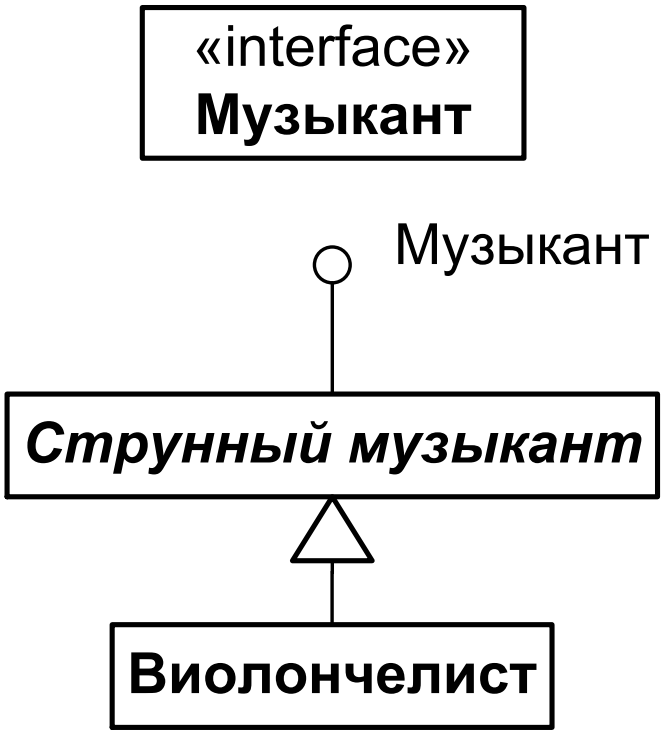
Абстрактные базовые классы

- Некоторые классы существуют только для того, чтобы от них наследовались другие классы
  - Запрещено создавать экземпляры таких классов
  - Эти классы являются абстрактными



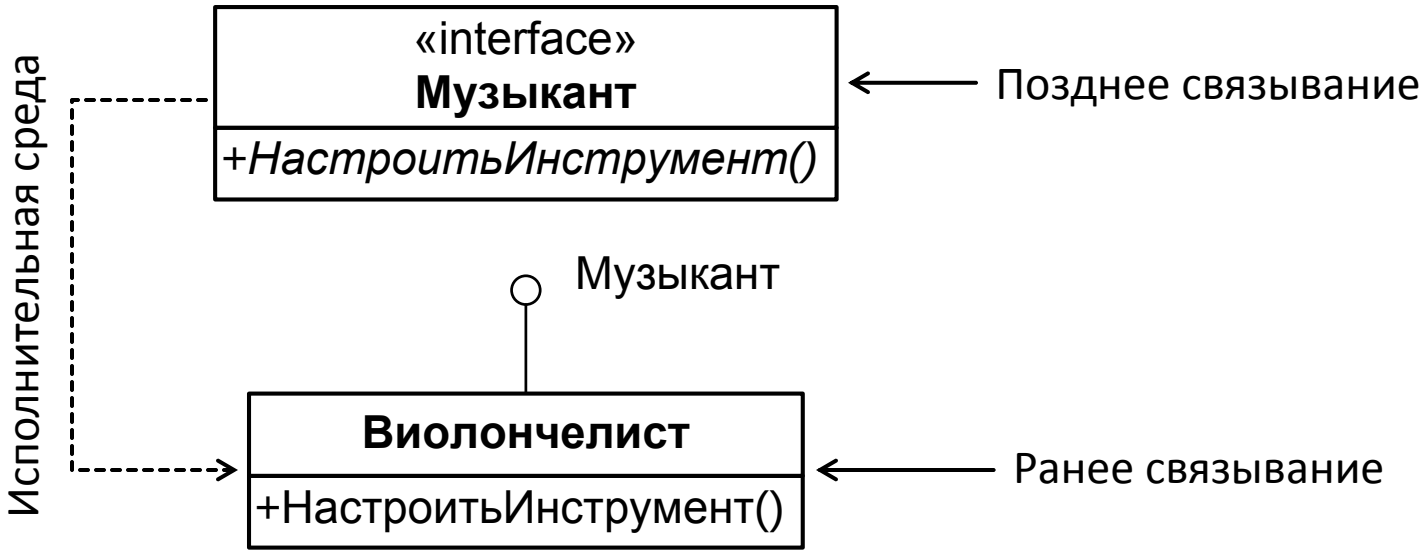
Интерфейсы

- Интерфейсы содержат только операции без их реализации



Раннее и позднее связывание

- Обычные вызовы методов могут быть определены во время компиляции
- Вызовы полиморфических методов определяются во время выполнения программы



Заключение

- Спасибо за внимание!