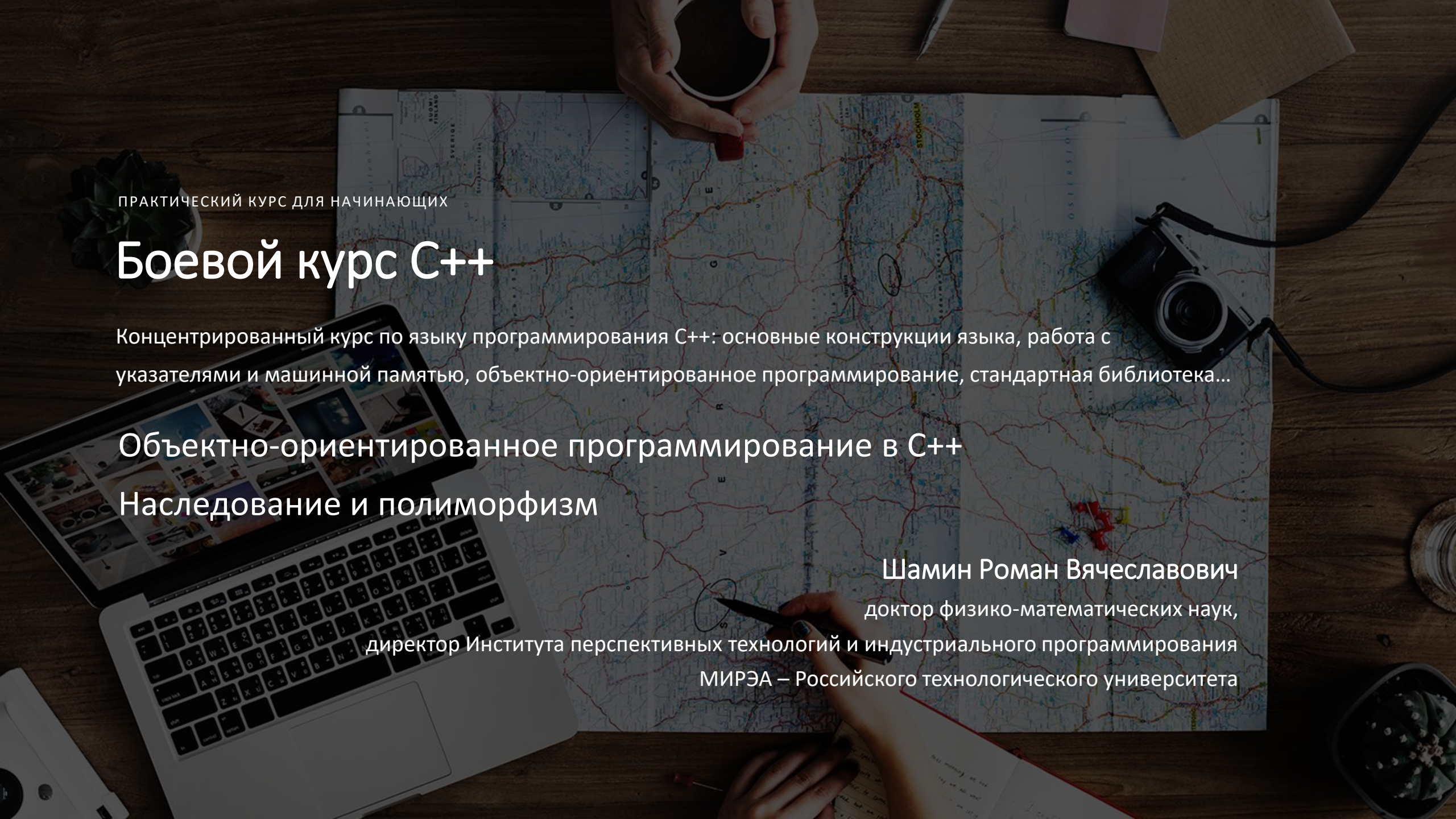


ПРАКТИЧЕСКИЙ КУРС ДЛЯ НАЧИНАЮЩИХ

Боевой курс C++

Концентрированный курс по языку программирования C++: основные конструкции языка, работа с указателями и машинной памятью, объектно-ориентированное программирование, стандартная библиотека…

Объектно-ориентированное программирование в C++

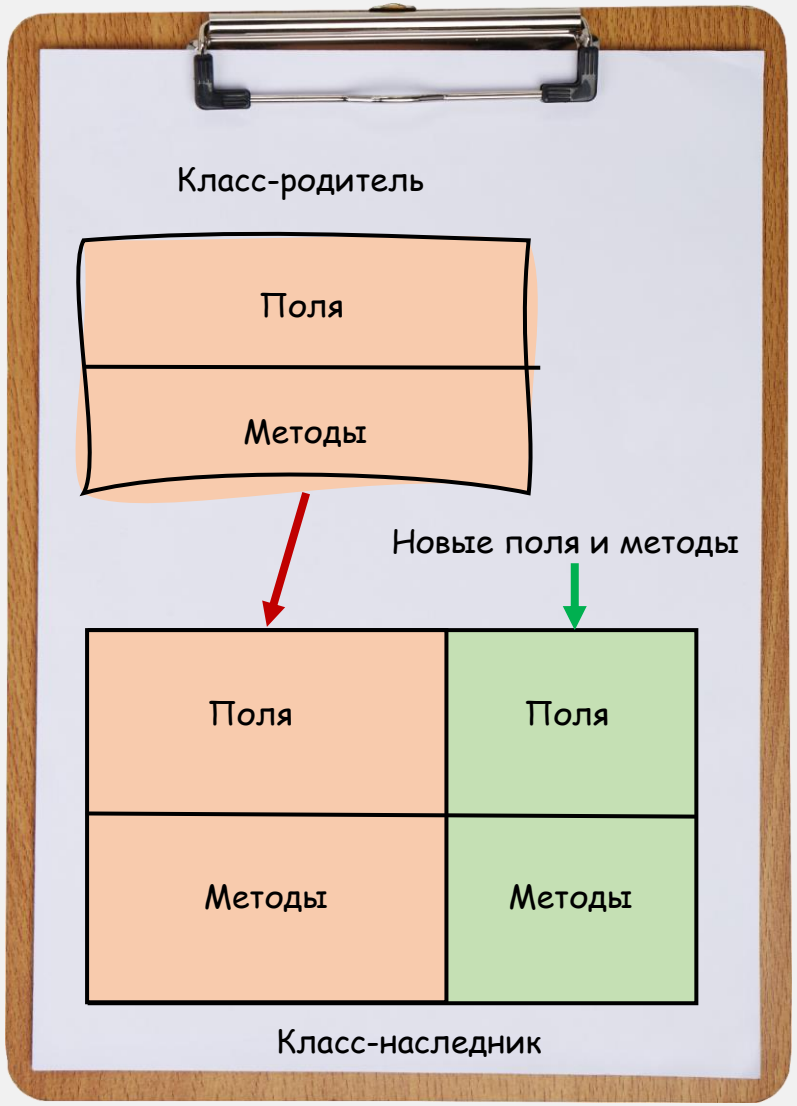
Наследование и полиморфизм

Шамин Роман Вячеславович
доктор физико-математических наук,
директор Института перспективных технологий и индустриального программирования
МИРЭА – Российского технологического университета

Объектно-ориентированное программирование основано на том, что создается иерархия классов и основным механизмом является наследование классов. Язык C++ поддерживает множественное наследование, но современные языки программирования от множественного наследования отказываются.

Наследование работает следующим образом: при создании класса можно указать другой класс, который будет унаследован. В этом случае класс-наследник будет иметь все поля и методы класса-родителя.

```
class TPoint // объявляем класс
{
public:
   TPoint(double x, double y); // объявляем конструктор
   double Dist(); // объявляем метод
private:
   double XY[2]; // координаты (x, y)
};
TPoint::TPoint(double x, double y) // реализуем конструктор
{
   XY[0] = x;
   XY[1] = y;
}
double TPoint::Dist() // реализуем метод
{
   return sqrt(XY[0] * XY[0] + XY[1] * XY[1]);
}
```



Посмотреть видео

Теперь создадим класс-наследник.

```
class TCircle : public TPoint // создаем класс, наследующий TPoint
{
public:
    TCircle(double x, double y, double r) : TPoint(x, y) // вызываем конструктор предка
    {
        this->r = r; // сохраняем переменную
    }

    bool IsO() // метод для проверки содержится ли начало координат в круге
    {
        return (Dist() < r); // возвращаем логическое значение
    }

private:
    double r; // скрытое поле – радиус круга
};
```

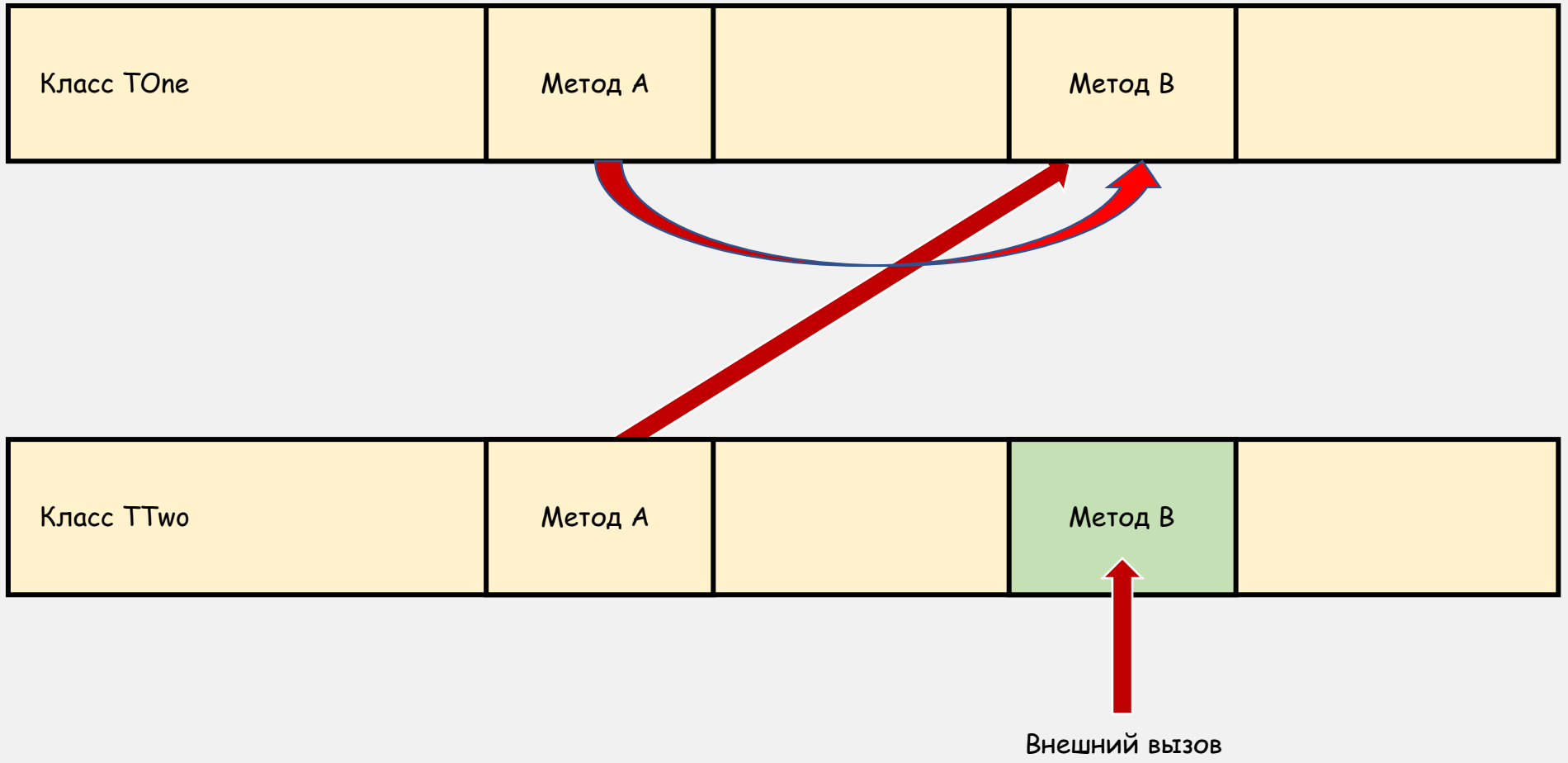
Используем класс-наследник.

```
TCircle* C = new TCircle(3, 4, 10); // создаем объект и передаем параметры

cout << C->Dist() << endl; // выводим расстояние от центра до нуля
cout << C->IsO() << endl; // выводим результат проверки
```

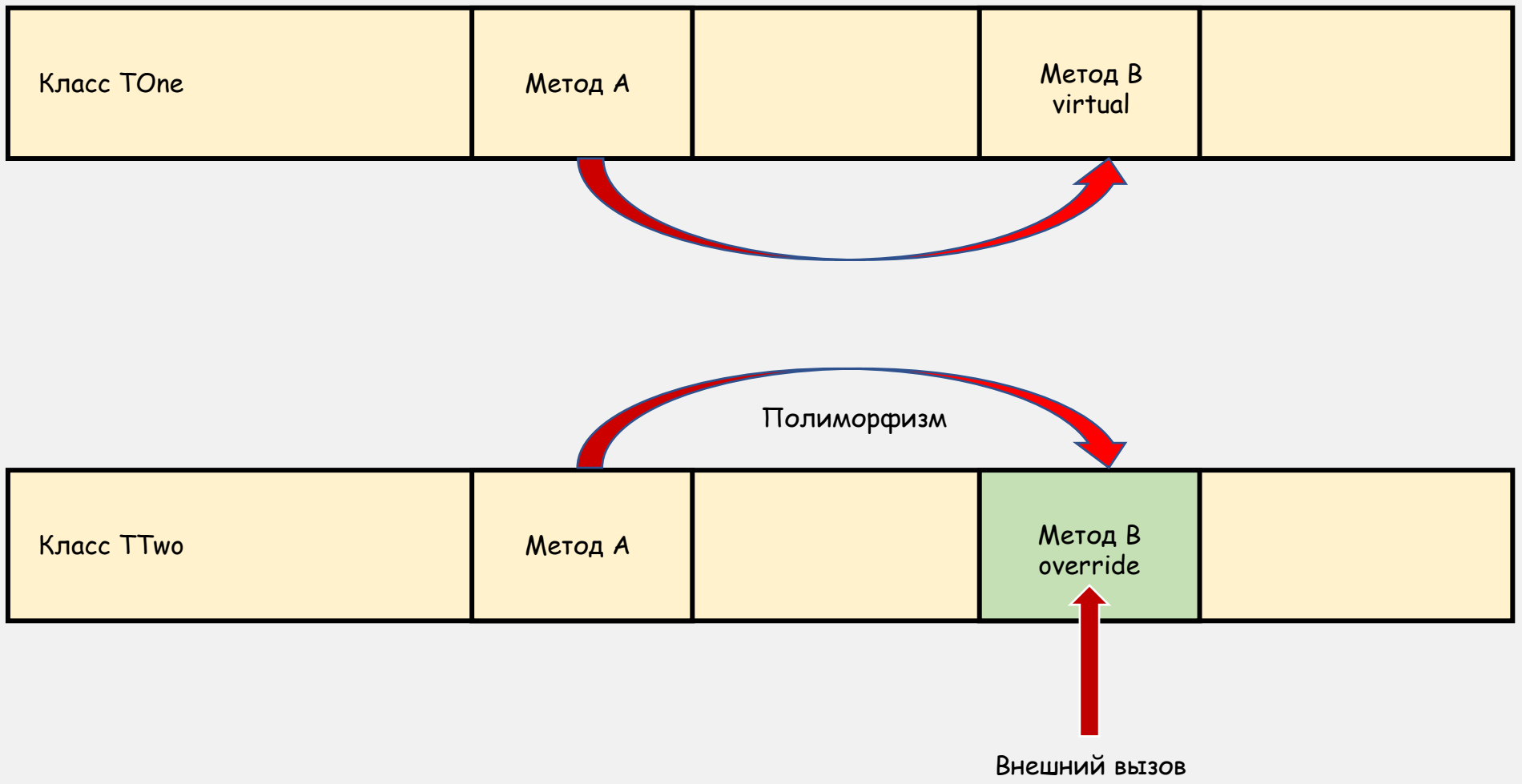
Если в наследном классе мы определим метод, который имеется в родительском классе, то при вызове его из наследного класса, он будет заменен на новый метод. Этот эффект называется сокрытием метода.

Но если мы заменим таким образом метод, который вызывается из других методов родительского класса, то при этих вызовах будет использоваться метод, определенный в родительском классе.



Идея полиморфизма

Если переопределяемый метод будет помечен как «виртуальный», то будет возможным сделать так, чтобы все методы в наследном классе будут вызывать переопределенный класс. Этот механизм называется полиморфизмом.



Программируем полиморфизм

```
class TOne // объявляем класс
{
public:
    TOne(int a) // конструктор
    {
        this->a = a;
    }
    int a;
    void ReCalc() // метод, изменяющий значение a
    {
        a = a * a;
    }
    void Print() // метод, использующий ReCalc
    {
        ReCalc(); // вызываем другой метод
        cout << endl << a << endl;
    }
};

class TTwo : public TOne // класс наследник
{
public:
    TTwo(int a) : TOne(a)
    {
    }
    void ReCalc() // переопределяем метод
    {
        a = a * a * a;
    }
};
```

```
TOne* One = new TOne(2);
One->Print(); // будет выведено 4

TTwo* Two = new TTwo(2);
Two->Print(); // будет выведено 4
```

Программируем полиморфизм

```
class TOne // объявляем класс
{
public:
    TOne(int a) // конструктор
    {
        this->a = a;
    }
    int a;
    virtual void ReCalc() // виртуальный метод, изменяющий значение a
    {
        a = a * a;
    }
    void Print() // метод, использующий ReCalc
    {
        ReCalc(); // вызываем другой метод
        cout << endl << a << endl;
    }
};

class TTwo : public TOne // класс наследник
{
public:
    TTwo(int a) : TOne(a)
    {
    }
    void ReCalc() override // переопределяем метод
    {
        a = a * a * a;
    }
};
```

```
TOne* One = new TOne(2);
One->Print(); // будет выведено 4

TTwo* Two = new TTwo(2);
Two->Print(); // будет выведено 8
```

Скачать текст программы

Посмотреть видео