МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Российский государственный университет им. А.Н. Косыгина   
(Технологии. Дизайн. Искусство)»**

Институт ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И ЦИФРОВЫЕ ТРАНСФОРМАЦИИ

**Отчет по лабораторной работе № 4**

**по дисциплине «Объектно-ориентированное программирование»**

**Тема: «Классы. Инкапсуляция. Реализация класса PayRoad»**

Выполнил: Сидоров Д. С., группа ИТС-123

Проверил: к.т.н., доц. Семёнов А. А.

Москва, 2025г

**Классы. Инкапсуляция. Реализация класса PayRoad**

Изучить и повторить проиллюстрированный ниже пример, демонстрирующий реализацию класса платной дороги с тарифом 500,50 за проезд. В примере разрабатывается визуальное приложение (Windows VCL Application), в котором необходимо создать класс с именем PayRoad, содержащий три поля:

int Cars;

float Cash; // наличные

float NonCash; // безналичные

Они предназначены для хранения кол-ва машин, наличных и безналичных платежей. Нулевой конструктор класса должен инициализировать поля нулевыми значениями. Необходимо создать четыре метода класса, которые будут увеличивать счетчик проехавших машин, выводить в LabeledEdit кол-во проехавших машин, выводить суммы платежей (наличных и безналичных), считать и выводить сводные данные. Также необходимо Заблокировать кнопки Оплата. Вместо них гененрировать случайный проезд машин за наличные и безналичные. Создать визуализатор проехавшей машины.

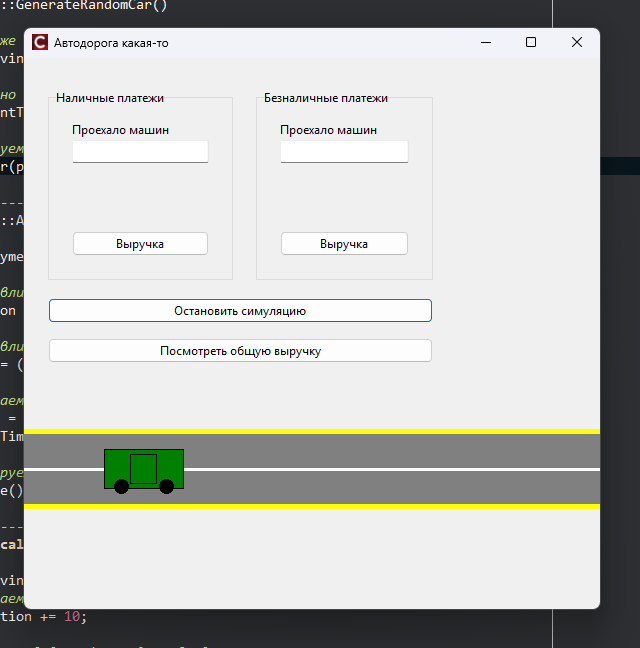


Рисунок 1. Работающая программа

Были произведены доработки относительно демонстрационного примера. Программа случайно определяет по какой оплате проехала машина (Цвет машины также бывает разный, зелёный или синий.) Добавлена автомобильная дорогая снизу. Когда машина доезжает до конца, производится оплата.

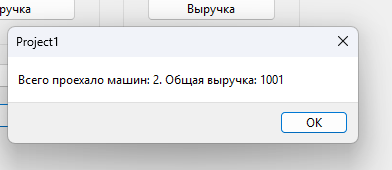


Рисунок 2. Подсчёт общей выручки

**Листинг кода**

**Unit1.cpp**

///---------------------------------------------------------------------------

#include <vcl.h>

#pragma hdrstop

#include "Unit1.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm1 \*Form1;

PayRoad CarCash, CarNonCash, TotalCars;

void PayRoad::paying(int flag){ // увеличиваем счетчик проехавших

Cars++;

if(flag==1) Cash+=500.50;

if(flag==2) NonCash+=500.50;

}

void PayRoad::show(int flag){

if(flag==1) Form1->LabeledEdit1->Text = Cars;

if(flag==2) Form1->LabeledEdit2->Text = Cars;

}

void PayRoad::sum(int flag) {

if(flag==1) ShowMessage(Cash);

if(flag==2) ShowMessage(NonCash);

}

void PayRoad::total(PayRoad car1, PayRoad car2) {

Cars = car1.Cars + car2.Cars;

Cash = car1.Cash + car2.Cash;

NonCash = car1.NonCash + car2.NonCash;

AnsiString s = "Всего проехало машин: " + IntToStr(Cars) + "." + " Общая выручка: " + FloatToStr(Cash+NonCash);

ShowMessage(s);

}

//---------------------------------------------------------------------------

\_\_fastcall TForm1::TForm1(TComponent\* Owner)

: TForm(Owner)

{

carPosition = -100;

carMoving = false;

currentPaymentType = 0;

carColor = clRed;

// Инициализация для визуализации

RandomCarTimer->Enabled = false;

RandomCarTimer->Interval = 2000; // 2 секунды между случайными машинами

AnimationTimer->Enabled = false;

AnimationTimer->Interval = 30; // Интервал для анимации движения

// Настраиваем дорогу - она будет в нижней части формы

roadHeight = 80;

// roadY будет рассчитан в FormCreate

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::FormCreate(TObject \*Sender)

{

// Рассчитываем положение дороги (внизу формы, но выше кнопок)

roadY = ClientHeight - roadHeight - 100;

// Начальная позиция машины по Y (по центру дороги)

carY = roadY + roadHeight/2 - 20; // 20 - половина высоты машины

// Устанавливаем DoubleBuffered для устранения мерцания при анимации

DoubleBuffered = true;

}

//---------------------------------------------------------------------------

// Отрисовка всей сцены

void \_\_fastcall TForm1::FormPaint(TObject \*Sender)

{

// Рисуем дорогу

DrawRoad(Canvas);

// Рисуем машину, если она видима

if (carMoving || carPosition > -100) {

DrawCar(Canvas);

}

}

//---------------------------------------------------------------------------

// Рисуем дорогу

void TForm1::DrawRoad(TCanvas \*Canvas)

{

// Рисуем серую дорогу

Canvas->Brush->Color = clGray;

Canvas->FillRect(TRect(0, roadY, ClientWidth, roadY + roadHeight));

// Рисуем желтые обочины

Canvas->Brush->Color = clYellow;

Canvas->FillRect(TRect(0, roadY, ClientWidth, roadY + 5));

Canvas->FillRect(TRect(0, roadY + roadHeight - 5, ClientWidth, roadY + roadHeight));

// Рисуем пунктирную разметку по центру дороги

Canvas->Pen->Color = clWhite;

Canvas->Pen->Width = 3;

Canvas->Pen->Style = psDot;

int centerY = roadY + roadHeight/2;

Canvas->MoveTo(0, centerY);

Canvas->LineTo(ClientWidth, centerY);

// Возвращаем стиль пера в сплошной для остальной отрисовки

Canvas->Pen->Style = psSolid;

Canvas->Pen->Width = 1;

}

//---------------------------------------------------------------------------

// Рисуем машину

void TForm1::DrawCar(TCanvas \*Canvas)

{

// Размеры машины

int carWidth = 80;

int carHeight = 40;

// Основная часть машины

Canvas->Brush->Color = carColor;

Canvas->Pen->Color = clBlack;

Canvas->Rectangle(carPosition, carY, carPosition + carWidth, carY + carHeight);

// Кабина машины

Canvas->Rectangle(carPosition + carWidth/3, carY + 5, carPosition + 2\*carWidth/3, carY + carHeight - 5);

// Колеса

Canvas->Brush->Color = clBlack;

Canvas->Ellipse(carPosition + 10, carY + carHeight - 10, carPosition + 25, carY + carHeight + 5);

Canvas->Ellipse(carPosition + carWidth - 25, carY + carHeight - 10, carPosition + carWidth - 10, carY + carHeight + 5);

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Button3Click(TObject \*Sender)

{

CarCash.sum(1);

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Button4Click(TObject \*Sender)

{

CarNonCash.sum(2);

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Button5Click(TObject \*Sender)

{

TotalCars.total(CarCash, CarNonCash);

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::RandomButtonClick(TObject \*Sender)

{

// Запускаем или останавливаем таймер случайных машин

RandomCarTimer->Enabled = !RandomCarTimer->Enabled;

if (RandomCarTimer->Enabled) {

RandomButton->Caption = "Остановить симуляцию";

GenerateRandomCar(); // Сразу генерируем первую машину

} else {

RandomButton->Caption = "Продолжить симуляцию";

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::RandomCarTimerTimer(TObject \*Sender)

{

GenerateRandomCar();

}

//---------------------------------------------------------------------------

void TForm1::GenerateRandomCar()

{

// Если уже идет анимация, не запускаем новую

if (carMoving) return;

// Случайно выбираем тип оплаты (1 = наличные, 2 = безналичные)

int paymentType = (rand() % 2) + 1;

// Анимируем машину

AnimateCar(paymentType);

}

//---------------------------------------------------------------------------

void TForm1::AnimateCar(int paymentType)

{

currentPaymentType = paymentType;

// Устанавливаем начальную позицию машины

carPosition = -100;

// Устанавливаем цвет машины в зависимости от типа оплаты

carColor = (paymentType == 1) ? clGreen : clBlue;

// Запускаем анимацию

carMoving = true;

AnimationTimer->Enabled = true;

// Инициируем перерисовку

Invalidate();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::AnimationTimerTimer(TObject \*Sender)

{

if (carMoving) {

// Двигаем машину

carPosition += 10;

// Перерисовываем форму для обновления позиции машины

Invalidate();

// Если машина достигла правого края формы

if (carPosition > ClientWidth) {

// Останавливаем анимацию

carMoving = false;

AnimationTimer->Enabled = false;

// Регистрируем проезд и оплату

if (currentPaymentType == 1) {

CarCash.paying(1);

CarCash.show(1);

} else if (currentPaymentType == 2) {

CarNonCash.paying(2);

CarNonCash.show(2);

}

// Сбрасываем позицию машины

carPosition = -100;

// Перерисовываем форму для скрытия машины

Invalidate();

}

}

}

//---------------------------------------------------------------------------

**Unit1.h**

/

//---------------------------------------------------------------------------

#ifndef Unit1H

#define Unit1H

//---------------------------------------------------------------------------

#include <System.Classes.hpp>

#include <Vcl.Controls.hpp>

#include <Vcl.StdCtrls.hpp>

#include <Vcl.Forms.hpp>

#include <Vcl.ExtCtrls.hpp>

#include <Vcl.Mask.hpp>

#include <Vcl.Graphics.hpp>

//---------------------------------------------------------------------------

class TForm1 : public TForm

{

\_\_published: // IDE-managed Components

TGroupBox \*GroupBox1;

TLabeledEdit \*LabeledEdit1;

TGroupBox \*GroupBox2;

TLabeledEdit \*LabeledEdit2;

TButton \*Button3;

TButton \*Button4;

TButton \*Button5;

TTimer \*AnimationTimer;

TTimer \*RandomCarTimer;

TButton \*RandomButton;

void \_\_fastcall Button3Click(TObject \*Sender);

void \_\_fastcall Button4Click(TObject \*Sender);

void \_\_fastcall Button5Click(TObject \*Sender);

void \_\_fastcall AnimationTimerTimer(TObject \*Sender);

void \_\_fastcall RandomButtonClick(TObject \*Sender);

void \_\_fastcall RandomCarTimerTimer(TObject \*Sender);

void \_\_fastcall FormCreate(TObject \*Sender);

void \_\_fastcall FormPaint(TObject \*Sender);

private: // User declarations

int carPosition; // X-позиция машины

int carY; // Y-позиция машины

bool carMoving;

int currentPaymentType; // 1 - наличные, 2 - безналичные

int roadY; // Верхняя координата дороги

int roadHeight; // Высота дороги

TColor carColor; // Цвет машины

void AnimateCar(int paymentType);

void GenerateRandomCar();

void DrawRoad(TCanvas \*Canvas); // Рисование дороги

void DrawCar(TCanvas \*Canvas); // Рисование машины

public: // User declarations

\_\_fastcall TForm1(TComponent\* Owner);

};

class PayRoad { // класс платной дороги

private:

int Cars;

float Cash;

float NonCash;

public:

PayRoad() { Cars=0; Cash=0.0; NonCash=0.0; }

void paying(int flag); // увеличиваем счетчик проехавших

void sum(int flag); // суммируем платежи

void total(PayRoad car1, PayRoad car2); // считаем итого по нал. и безнал. платежам

void show(int flag); // выводим showmessage

};

//---------------------------------------------------------------------------

extern PACKAGE TForm1 \*Form1;

//---------------------------------------------------------------------------

#endif

**Вывод:** В рамках данной лабораторной работы была разработана программа моделирования платной дороги с визуальным отображением процесса движения автомобилей. Реализован механизм случайной генерации транспортных средств и их анимированное перемещение по дорожному полотну. Программа позволяет отслеживать количество проехавших автомобилей и собранную оплату, разделяя их на наличные и безналичные платежи. Система визуализации построена на прямой отрисовке элементов на холсте формы, что обеспечивает точный контроль над расположением объектов и устраняет проблемы с перекрытием компонентов. Использование таймеров позволило создать плавную анимацию и автоматизировать генерацию случайных событий. Данная реализация демонстрирует применение объектно-ориентированного подхода к программированию, где класс PayRoad инкапсулирует данные о транспортном потоке и собранных средствах.