Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Институт информационных технологий

Кафедра ИС

# ОТЧЕТ

по лабораторной работе №4

ИССЛЕДОВАНИЕ СПОСОБОВ ИНТЕГРАЦИОННОГО ТЕСТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Выполнил:

ст. гр. ИС/б-21-2-о

Мовенко К. М.

Проверил:

Петраков В. А.

Севастополь

2023

# ЦЕЛЬ РАБОТЫ

Исследовать основные принципы интеграционного тестирования программного обеспечения. Приобрести практические навыки организации интеграционных тестов для объектно-ориентированных программ.

# ЗАДАНИЕ

* 1. Выбрать в качестве тестируемого один из классов, спроектированных в лабораторных работах №1-2;
  2. Составить спецификацию тестового случая для одного из методов выбранного класса;
  3. Реализовать тестируемый класс и необходимое тестовое окружение на языке С#;
  4. Выполнить тестирование с выводом результатов на экран и сохранением в log-файл;
  5. Проанализировать результаты тестирования, сделать выводы;

# ХОД РАБОТЫ

Для тестирования было рассмотрено взаимодействие двух классов: “Car” и “Driver”, способ взаимодействия которых − общедоступная операция с формальным параметром объектного типа Car (отношение агрегации).

Была составлена спецификация тестового случая:

1. **Названия взаимодействующих классов**: Car, Driver.
2. **Название теста**: CarDriverTest1.
3. **Описание теста**: тест проверяет возможность создания объекта типа Driver и правильное взаимодействие между машиной и водителем посредством вывода сообщения о том, кто завел автомобиль и какой марки он был.
4. **Начальные условия**: автомобиль и водитель не определены.
5. **Ожидаемый результат**: вывод сообщения о запуске и остановке авто, содержащее информацию об имени водителя и марке автомобиля.

На основе этой спецификации был разработан тестовый драйвер – класс CarDriverTester, который наследуется от класса Tester. Тестируемые классы и необходимое тестовое окружение были реализованы на языке С#.

public class Car

{

private string \_model;

public string getModel()

{

return \_model;

}

public Car(string model)

{

\_model = model;

}

public string Start()

{

return $"The {\_model} has started.\n";

}

public string Stop()

{

return $"The {\_model} has stopped.\n";

}

}

public class Driver

{

private string \_name;

private Car \_car;

public Driver(string name, Car car)

{

\_name = name;

\_car = car;

}

public string StartCar()

{

string ret = $"{\_name} is starting the car {\_car.getModel()}.\n" + \_car.Start() + "\n";

return ret;

}

public string StopCar()

{

string ret = $"{\_name} is stopping the car {\_car.getModel()}.\n" + \_car.Stop() + "\n";

return ret;

}

public string DataToString()

{

return $"DRIVER DATA:\n======================\nDriver: {\_name}\nCar: {\_car.getModel()}\n";

}

}

public class Logger

{

static private StreamWriter log = new StreamWriter("log.log");

static public void Add(string message) => log.WriteLine(message);

static public void Close() => log.Close();

}

abstract class Tester

{

protected void LogMessage(string message)

{

Logger.Add(message);

}

}

class CarDriverTester : Tester

{

Car car;

Driver driver;

public CarDriverTester()

{

Run();

}

private void Run()

{

CarDriverTest1();

}

public void Init()

{

car = new Car("Toyota Camry");

driver = new Driver("Konstantin Movenko", car);

}

private void CarDriverTest1()

{

Init();

LogMessage("///////// CarDriverTest1 /////////////");

LogMessage("Проверяем, создается ли объект типа Driver");

string driverData = driver.DataToString();

Console.WriteLine(driverData);

LogMessage(driverData);

string res = driver.StartCar() + driver.StopCar();

Console.WriteLine(res);

LogMessage($"{res}");

LogMessage("Тест успешно выполнен");

}

}

public static void Main(string[] args)

{

CarDriverTester tester = new CarDriverTester();

Logger.Close();

}

Было выполнено тестирование с выводом результатов на экран (Рисунок 1) и сохранением в log-файл (Рисунок 2).

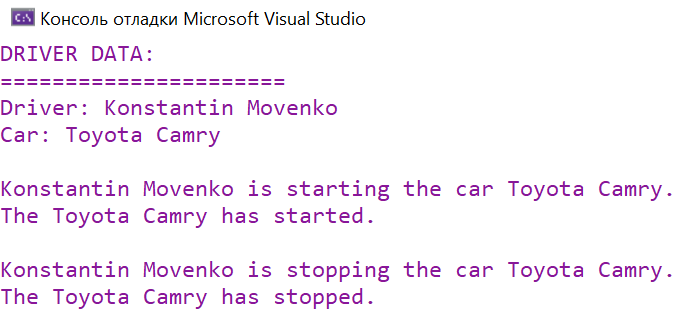


Рисунок 1 – Вывод результатов тестирования на экран

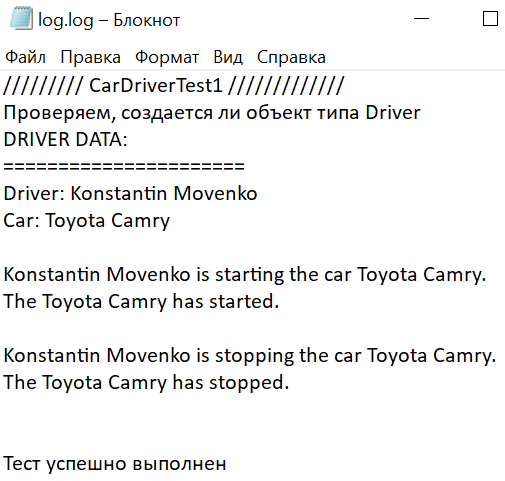


Рисунок 2 – Сохранение результатов тестирования в log-файл

Были проанализированы результаты тестирования. Класс Driver правильно определяет автомобиль (класс Car) своего объекта и корректно использует его данные для вывода сообщения о марке машины и её водителе. Программа успешно справляется со своей задачей.

# ВЫВОД

В ходе работы были получены навыки составления интеграционных тестов для тестирования взаимодействия двух или нескольких модулей. Данный метод позволяет отслеживать ошибки, возникаемые при совместной работе нескольких модулей системы.