Модульное тестирование

1. Цель работы

1. Обеспечение корректности отдельных модулей программы.
2. Выявление ошибок на ранних этапах разработки.
3. Снижение риска регрессий при внесении изменений в код.
4. Ускорение процесса разработки за счёт автоматизированной проверки функциональности.
5. Повышение качества и поддерживаемости кода.
6. Автоматизация тестирования для экономии времени и ресурсов.

2. Описание проекта

Проект, разработанный в рамках лабораторной работы №1, представляет собой приложение для управления резервированием. Основные компоненты проекта:

* Класс Reservation — описывает резервирование (имя клиента, дата начала и окончания резервирования).
* Класс ReservationManager — класс, управляющий сохраненными данными по резервации разных пользователей.
* Класс ReservationForm — описывает интерфейс программы.

3. Разработанные модульные тесты

3.1. Тесты для класса Reservation

Проверены следующие сценарии:

* Метод Reservation\_Constructor\_InitializesPropertiesCorrectly() проверяет конструктор класса Reservation.
* Метод Reservation\_UpdateStatus\_ChangesStatusCorrectly() проверяет корректность резервирования метода UpdateStatus().
* Метод Reservation\_ToString\_ReturnsCorrectFormat() проверяет, что метод ToString() формирует читаемую строку с ключевой информацией о резервировании.

3.2. Тесты для класса ReservationForm

Проверены следующие сценарии:

* Метод ReservationForm\_Constructor\_InitializesCorrectly() проверяет конструктор класса ReservationForm.
* Метод ReservationForm\_Controls\_AreCreated() проверяет, что на форме присутствуют все необходимые элементы управления.
* Метод ReservationForm\_AddReservation\_ValidData() проверяет, что при вводе корректных данных и нажатии кнопки «Добавить» создаётся новое бронирование и сохраняется в менеджере.
* Метод ReservationForm\_UpdateReservationsList\_WorksCorrectly() проверяет, что метод обновления списка бронирований (UpdateReservationsList) корректно заполняет ListBox.
* Метод ReservationForm\_GetSelectedReservation\_ReturnsCorrectReservation() проверяет, что метод GetSelectedReservation возвращает правильный объект бронирования, соответствующий выбранному элементу в списке.

3.3. Тесты для класса ReservationManager

Проверены следующие сценарии:

* Метод ReservationManager\_AddReservation\_AddsSuccessfully() проверяет, что метод AddReservation корректно добавляет новое бронирование в список.
* Метод ReservationManager\_AddReservation\_NullReservation\_ThrowsException() проверяет, что попытка добавить null вызывает исключение.
* Метод ReservationManager\_RemoveReservation\_RemovesSuccessfully() проверяет корректность удаления бронирования из списка.
* Метод ReservationManager\_UpdateReservationStatus\_UpdatesSuccessfully() проверяет, что статус бронирования можно изменить через менеджер.
* Метод ReservationManager\_SaveAndLoadReservations\_WorksCorrectly() проверяет сохранение бронирований в файл и последующую загрузку при создании нового менеджера.
* Метод ReservationManager\_AddReservation\_OverlappingTime\_ThrowsException() проверяет, что система не позволяет создавать пересекающиеся бронирования.
* Метод ReservationManager\_IsTimeSlotOccupied\_ReturnsCorrectValue() проверяет работу приватного метода IsTimeSlotOccupied, который определяет, занят ли указанный временной интервал.

4. Результаты тестирования

| **Категория тестов** | **Количество тестов** | **Успешно пройдено** | **Не пройдено** |
| --- | --- | --- | --- |
| ReservationFormTests | 5 | 2 | 3 |
| ReservationManagerTests | 7 | 2 | 5 |
| ReservationTests | 4 | 3 | 1 |
| Итого | 16 | 7 | 9 |

5. Вывод

В ходе выполнения лабораторной работы были разработаны и реализованы модульные тесты для всех ключевых компонентов приложения. Тесты охватывают функциональную логику, обработку ошибок.