МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВО «НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

КАФЕДРА АВТОМАТИЗИРОВАННЫХ СИСТЕМ УПРАВЛЕНИЯ



**Отчет по лабораторной работе №3**

**Дисциплина**: Программирование

Вариант **№7**

|  |  |  |
| --- | --- | --- |
| **Выполнили:** |  | **Проверил:** |
| Студенты | *Федотов И.В.*  *Шестаков К.Д.* | Старший преподаватель кафедры АСУ Эстрайх И.В |
| Факультет | *АВТФ* |
| Направление (специальность) подготовки | *09.03.03 – Прикладная информатика* |
| Группа | *АП-227* |

Новосибирск

2023

**Оглавление**

[**Постановка задачи** 3](#_Toc154837137)

[**Описание разработанных классов** 4](#_Toc154837138)

[Класс Program 4](#_Toc154837139)

[Класс GameBoard 5](#_Toc154837140)

[Класс Player 6](#_Toc154837141)

[Класс Unit 7](#_Toc154837142)

[Класс Ability 8](#_Toc154837143)

[Класс GameState 8](#_Toc154837144)

[**Модульное тестирование** 9](#_Toc154837145)

[Класс GameBoard 9](#_Toc154837146)

[Класс Player 9](#_Toc154837147)

[Класс Unit 10](#_Toc154837148)

[Класс Ability 10](#_Toc154837149)

[Класс GameState 10](#_Toc154837150)

[**Инструкция пользования приложением:** 11](#_Toc154837151)

[**Тестирование программы** 13](#_Toc154837152)

[**Скриншоты работы программы** 14](#_Toc154837153)

[**Вывод** 16](#_Toc154837154)

# **Постановка задачи**

**Цель работы:** разработать консольное приложение с использованием командной работы и принципов объектно-ориентированного программирования.

**Задание**

Вы должны работать в команде. Для этого начните с проектирования будущего приложения, разделите классы между членами бригады, договоритесь об интерфейсах, через которые ваши классы будут взаимодействовать. Исходный код должен находиться в репозитории Git, быть покрыт модульными тестами и документирован. Объекты должны поддерживать сериализацию и десериализацию. Должна быть возможность закрыть программу с сохранением состояния и затем запустить её из сохранённого состояния. ГПИ в этой ЛР не нужен.

**Вариант**

Минималистичная настольная военная игра (типа https://en.wikipedia.org/wiki/Warhammer\_40,000). Прямоугольное игровое поле с преградами. У существ есть как активные, так и пассивные способности.

# **Описание разработанных классов**

## Класс Program

|  |  |  |
| --- | --- | --- |
| **Метод** | **Тип возвращаемого значения** | **Описание** |
| Main | void | Точка входа в программу. Отвечает за вывод начального меню и обработку выбора пользователя. |
| StartNewGame | void | Инициализирует и запускает новую игру. |
| LoadExistingGame | void | Загружает сохранённое состояние игры, если оно существует, иначе начинает новую игру. |
| RunGameLoop | void | Запускает основной игровой цикл, управляющий игровыми событиями. |
| SetupPhase | void | Управляет фазой расстановки юнитов на игровом поле. |
| GetUserChoice | int | Возвращает выбор пользователя в заданном диапазоне. |
| IsGameFinished | bool | Проверяет условия окончания игры. |
| SaveGame | void | Сохраняет текущее состояние игры в файл. |
| LoadGame | GameState | Загружает сохранённое состояние игры из файла. |

## Класс GameBoard

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип** | **Описание** |
| width | int | Ширина игрового поля. |
| height | int | Высота игрового поля. |
| grid | char[,] | Двумерный массив символов, представляющий игровое поле. |
| units | List<Unit> | Список юнитов на игровом поле. |

|  |  |  |
| --- | --- | --- |
| **Метод** | **Тип возвращаемого значения** | **Описание** |
| GameBoard | Конструктор | Инициализирует игровое поле заданными размерами и инициализирует grid. |
| InitializeGrid | void | Инициализирует grid, заполняя его начальными символами. |
| SetObstacles | void | Расставляет препятствия на игровом поле. |
| SetUnit | void | Добавляет юнит на игровое поле. |
| DisplayAllUnits | void | Отображает информацию о всех юнитах на игровом поле. |
| IsValidPosition | bool | Проверяет, является ли позиция на поле допустимой для размещения юнита. |
| IsObstacle | private bool | Проверяет, является ли заданная позиция препятствием. |
| IsCellEmpty | bool | Проверяет, пуста ли заданная клетка на поле. |
| MoveUnit | void | Перемещает юнит на новую позицию на игровом поле. |
| AttackUnit | void | Осуществляет атаку одного юнита на другого. |
| IsValidMove | private bool | Проверяет, является ли перемещение юнита допустимым. |
| RemoveUnit | private void | Удаляет юнит с игрового поля. |
| Display | void | Отображает игровое поле и его содержимое в консоли. |

## Класс Player

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип** | **Описание** |
| Name | string | Имя игрока. |
| units | List<Unit> | Приватный список юнитов игрока. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Метод** | **Тип Возвращаемого Значения** | | **Описание** |
| Player | Конструктор | Инициализирует новый экземпляр класса Player с заданным именем. | |
| Units | List<Unit> | Свойство для доступа к списку юнитов игрока. | |
| GetUnitByName | Unit | Возвращает юнит по имени, если он существует в списке юнитов игрока. | |
| MoveUnit | void | Позволяет игроку перемещать выбранный юнит на игровом поле. | |
| AttackUnit | void | Позволяет игроку атаковать цель выбранным юнитом. | |
| UseActiveAbility | void | Активирует способность выбранного юнита. | |
| SetPassiveAbility | void | Устанавливает пассивную способность выбранного юнита. | |

## Класс Unit

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип** | **Описание** |
| Name | string | Имя юнита. |
| X | int | Горизонтальная координата юнита на поле. |
| Y | int | Вертикальная координата юнита на поле. |
| Health | double | Здоровье юнита. |
| Attack | double | Атака юнита. |
| Endurance | int | Выносливость юнита. |
| UnitType | string | Тип юнита. |
| Abilities | List<Ability>? | Список способностей юнита. |
| Symbol | char | Символ, представляющий юнита на игровом поле. |
| Owner | Player | Игрок, владеющий юнитом. |
| AbilityFlag | bool | Флаг, указывающий на активность способностей. |

|  |  |  |
| --- | --- | --- |
| **Метод** | **Тип Возвращаемого Значения** | **Описание** |
| Unit | Конструктор | Инициализирует новый экземпляр класса Unit с заданными параметрами. |
| DisplayAbilities | void | Отображает способности юнита в консоли. |
| GetActiveAbility | void | Активирует указанную активную способность юнита. |
| GetPassiveAbility | void | Активирует указанную пассивную способность юнита. |

## Класс Ability

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип** | **Описание** |
| Name | string | Название способности. |
| Description | string | Описание способности. |
| IsActive | bool | Флаг, указывающий, активна ли способность. |
| AbilityID | int | Уникальный идентификатор способности. |

|  |  |  |
| --- | --- | --- |
| **Метод** | **Тип Возвращаемого Значения** | **Описание** |
| Ability | Конструктор | Инициализирует новый экземпляр класса Ability с заданными параметрами. |

## Класс GameState

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип** | **Описание** |
| GameBoard | GameBoard | Игровое поле, содержащее информацию о расположении юнитов и препятствий. |
| Player1 | Player | Первый игрок в игре. |
| Player2 | Player | Второй игрок в игре. |
| IsSetupPhase | bool | Флаг, указывающий находится ли игра в фазе расстановки. |
| Player1Turn | bool | Флаг, указывающий, чей сейчас ход (Первого игрока или Второго). |

# **Модульное тестирование**

## Класс GameBoard

|  |  |  |
| --- | --- | --- |
| **Название Теста** | **Цель Теста** | **Ожидаемые Результаты** |
| GameBoard\_Initialization\_  CreatesCorrectSize | Проверить, что доска инициализируется с правильными размерами. | Утверждение равенства (Assert.AreEqual) размеров доски заданным значениям (10x10). |
| SetUnit\_  AddsUnitToGameBoard | Проверить, что метод SetUnit корректно добавляет юнит на доску. | Утверждение наличия (Assert.IsNotNull) добавленного юнита и правильного символа в соответствующей ячейке доски. |
| SetObstacles\_  CorrectlyPlacesObstacles | Проверить, что метод SetObstacles корректно размещает препятствия на доске. | Утверждение наличия препятствий в списке юнитов (Assert.IsTrue) и корректного размещения символов препятствий на доске (Assert.AreEqual). |

## Класс Player

|  |  |  |
| --- | --- | --- |
| **Название Теста** | **Цель Теста** | **Ожидаемые Результаты** |
| Constructor\_  AssignsNameCorrectly | Проверить, корректно ли инициализируется имя игрока. | Утверждение равенства (Assert.AreEqual) имени игрока с заданным при создании объекта. |
| AddUnit\_  IncreasesUnitCount | Проверить, что добавление юнита увеличивает количество юнитов у игрока. | Утверждение равенства (Assert.AreEqual) количества юнитов после добавления. |
| GetUnitByName\_  ReturnsCorrectUnit | Проверить, что метод GetUnitByName возвращает корректный юнит. | Утверждение не пустоты (Assert.IsNotNull) и равенства имени возвращенного юнита заданному. |
| GetUnitByName\_  ReturnsNullForUnknownName | Проверить, что метод GetUnitByName возвращает null для несуществующего имени юнита. | Утверждение пустоты (Assert.IsNull) возвращенного объекта для несуществующего имени. |

## Класс Unit

|  |  |  |
| --- | --- | --- |
| **Название Теста** | **Цель Теста** | **Ожидаемые Результаты** |
| Constructor\_  SetsPropertiesCorrectly | Проверить, что конструктор корректно инициализирует свойства юнита. | 1. Утверждение равенства (Assert.AreEqual) для каждого свойства юнита (имя, координаты, здоровье, атака, выносливость, тип, символ). |
| 2. Утверждение идентичности (Assert.AreSame) для свойства Owner. |
| 3. Утверждение наличия (Assert.IsTrue) добавленной способности в списке способностей юнита. |
| 4. Утверждение истинности (Assert.IsTrue) флага активности способностей. |

## Класс Ability

|  |  |  |
| --- | --- | --- |
| **Название Теста** | **Цель Теста** | **Ожидаемые Результаты** |
| Constructor\_SetsPropertiesCorrectly  Название Теста | Проверить, что конструктор корректно инициализирует свойства способности. | 1. Утверждение равенства (Assert.AreEqual) для свойства Name. |
| 2. Утверждение истинности (Assert.IsTrue) для свойства IsActive. |

## Класс GameState

|  |  |  |
| --- | --- | --- |
| **Название Теста** | **Цель Теста** | **Ожидаемые Результаты** |
| Constructor\_SetsPropertiesCorrectly  Название Теста | Проверить, что свойства GameState корректно устанавливаются.  Цель Теста | 1. Утверждение идентичности (Assert.AreSame) для свойств GameBoard, Player1 и Player2. |
| 2. Утверждение равенства (Assert.AreEqual) для булевых свойств IsSetupPhase и Player1Turn. |

# **Инструкция пользования приложением**

Начало игры:

* При запуске приложения появится приветственное сообщение с тремя опциями: "1) Начать игру", "2) Загрузить игру", "3) Выход".
* Выберите соответствующую опцию, введя число от 1 до 3.

Новая игра:

* Если вы выбрали "1) Начать игру", создается новая игровая доска размером 10x10 и предлагается вам расставить свои юниты.
* За игрока выступают два участника: "Игрок1" и "Игрок2".
* Фаза расстановки: каждый игрок имеет 3 хода для размещения своих юнитов на доске.

Расстановка юнитов:

* Поочередно выбирайте тип юнита и указывайте координаты (x, y) для размещения на доске.
* Доступные типы юнитов: Воин, Лучник, Маг, Паладин, Берсерк, Целитель.
* Вводите имя и символ для обозначения юнита.
* Следите за правилами размещения, координаты зависят от выбранного игрока.

Игровой цикл:

* После завершения фазы расстановки начинается основной игровой цикл.
* Каждый игрок поочередно выполняет действия:

1. Ходить (перемещение юнита).
2. Атаковать (атака другого юнита).
3. Использовать способность (активация активной способности).
4. Сохранить игру.
5. Выйти из игры.

Действия в игре:

* При выборе действия введите соответствующий номер.
* При ходе укажите направление (1-Вверх, 2-Вправо, 3-Вниз, 4-Влево) и количество клеток для перемещения.
* При атаке выберите атакующего юнита, затем цель атаки.
* При использовании способности выберите юнита и активируйте активную способность.

Сохранение и загрузка игры:

* В любой момент игры можно сохранить текущее состояние.
* Для сохранения игры выберите опцию "4 - Сохранить".
* Для загрузки игры выберите "2) Загрузить игру" при старте приложения.

Выход из игры:

* Вы можете выйти из игры в любой момент, выбрав опцию "5 - Выход".
* Прогресс сохраняется при выходе, и вы можете загрузить его при следующем запуске.

Окончание игры:

* Игра завершается, когда у одного из игроков заканчиваются юниты.
* При окончании игры вы увидите сообщение о завершении и приложение закроется.

Обработка ошибок:

* Если введены неверные данные, программа запросит их повторный ввод.
* В случае ошибок при сохранении или загрузке игры будет выведено соответствующее сообщение.

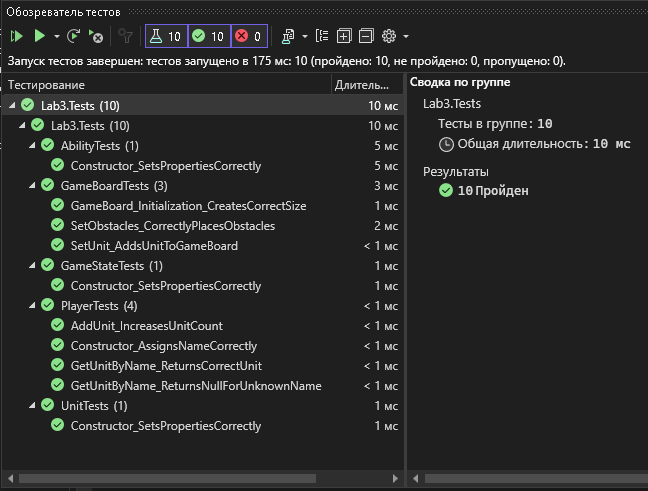
Завершение программы:

* По умолчанию, для выхода из программы используйте команду "3 - Выход".

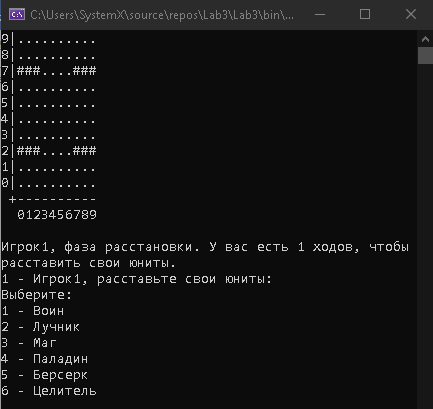
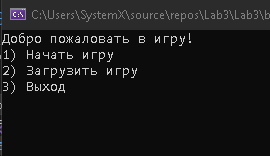
# **Тестирование программы**

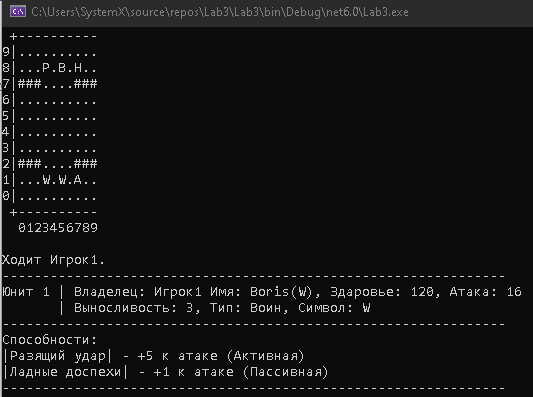
Все модульные тесты были успешно пройдены. Можно сделать вывод, что программа работает корректно и надежно. Это подтверждает, что основные функции программы, включая управление игровым процессом и взаимодействие с юнитами, соответствуют требованиям. Тесты также помогают устранить основные ошибки, обеспечивая стабильность программы. Это создает прочную основу для дальнейшего развития и упрощает дальнейшее расширение функциональности. В целом, успешные тесты говорят о готовности программы к следующим этапам разработки и использованию.

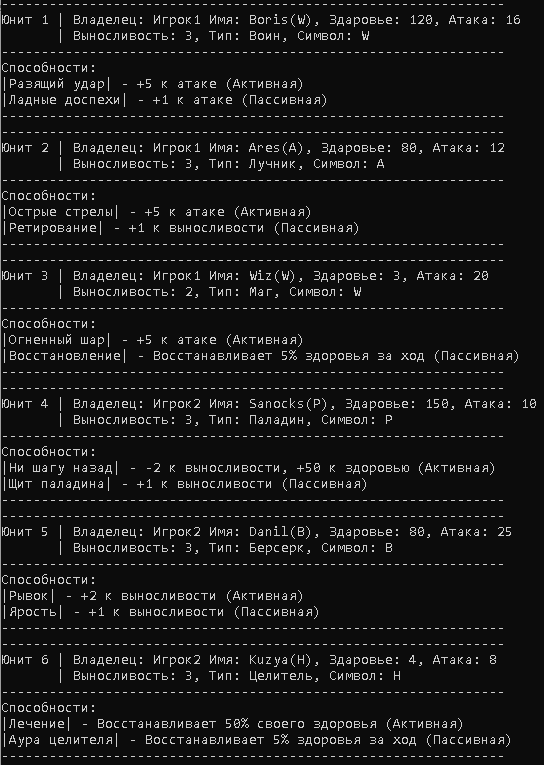
**Скриншот результата выполнения модульных тестов**

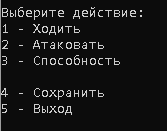


# **Скриншоты работы программы**









# **Вывод**

В процессе работы над программой были успешно разработаны основные компоненты, реализована функциональность сохранения и загрузки игры, и устранены ключевые ошибки. Модульное тестирование подтвердило надежность и корректность работы программы, обеспечивая ее стабильность. Подробная документация и анализ помогли в понимании структуры и логики программы, в результате чего была создана, готовая к дальнейшему использованию и развитию.