## Практическая работа №12. Проектирование классов

Цель: разработать консольное приложение с классами, язык программирования Python.

***Проект «Библиотека книг»***

*Цель проекта*: создание программы, которая позволяет управлять каталогом книг в библиотеке. Пользователи должны иметь возможность добавлять новые книги, удалять существующие, искать книги по различным критериям, а также просматривать список всех доступных книг.

*Функциональные требования*:

* Добавление новой книги в каталог: пользователь должен иметь возможность ввести информацию о книге (название, автор, жанр и т.д.) и сохранить ее в каталоге.
* Удаление книги из каталога: пользователь должен иметь возможность выбрать книгу из списка и удалить ее из каталога.
* Поиск книги по различным критериям: пользователь должен иметь возможность искать книги по названию, автору, жанру и другим характеристикам.
* Просмотр списка всех книг: пользователь должен иметь возможность просматривать полный список всех доступных книг в каталоге.

*Внешний интерфейс*. Программа должна предоставлять текстовый интерфейс для взаимодействия с пользователем. При запуске программа должна выводить приветствие и предлагать доступные команды (например, «Добавить книгу», «Удалить книгу», «Найти книгу» и т.д.). Пользователь должен вводить команды с клавиатуры для выполнения соответствующих действий.

*Предлагаемый дизайн классов*:

* Класс Book: представляет книгу с полями, такими как название, автор, жанр и т.д.
* Класс Library: представляет библиотеку и содержит список экземпляров класса Book. Методы класса Library должны реализовывать функции добавления, удаления, поиска и просмотра книг в каталоге.

***Проект «Таск-менеджер»***

*Цель проекта*: создание программы, которая позволяет пользователям создавать новые задачи, устанавливать им приоритет, отмечать выполнение, а также просматривать и редактировать список задач.

*Функциональные требования*:

* Создание новой задачи: пользователь должен иметь возможность ввести описание задачи, установить ей приоритет (высокий, средний, низкий) и добавить в список задач.
* Установка выполнения задачи: пользователь должен иметь возможность отметить задачу как выполненную.
* Редактирование задачи: пользователь должен иметь возможность изменить описание задачи или ее приоритет.
* Просмотр списка задач: пользователь должен иметь возможность просматривать список всех задач с указанием их приоритета и статуса выполнения.

*Внешний интерфейс*. Программа должна предоставлять текстовый интерфейс для взаимодействия с пользователем. При запуске программа должна выводить приветствие и предлагать доступные команды (например, «Создать задачу», «Отметить задачу выполненной», «Редактировать задачу» и т.д.). Пользователь должен вводить команды с клавиатуры для выполнения соответствующих действий.

*Предлагаемый дизайн классов*:

* Класс Task: представляет задачу с полями, такими как описание, приоритет и статус выполнения.
* Класс TaskManager: представляет менеджер задач и содержит список экземпляров класса Task. Методы класса TaskManager должны реализовывать функции создания, установки выполнения, редактирования и просмотра задач.

***Проект «Крестики-нолики»***

*Цель проекта*: предоставление игрокам возможности сыграть в "Крестики-нолики" и насладиться увлекательными схватками за линии крестиков или ноликов.

*Игровой процесс*:

* Начало игры: игроки запускают игру и видят текстовую доску 3x3, расположенную в консоли. Один игрок играет за крестики (Х), а другой за нолики (О).
* Ходы игроков: игроки совершают свои ходы поочередно. Ход осуществляется путем ввода координат клетки на доске, куда игрок хочет поставить свой символ (например, «1 2» для выбора клетки в первой строке и втором столбце).
* Правила игры: игра должна соблюдать все правила «Крестики-нолики», такие как запрет на повторное заполнение клетки и определение победителя, если в одной строке, столбце или диагонали собирается линия из трех одинаковых символов.
* Конец игры: игра продолжается до тех пор, пока не наступит один из концов: победа одного из игроков, ничья или запрос на завершение игры со стороны игроков.

*Внешний интерфейс*. Программа должна предоставлять текстовый интерфейс для взаимодействия с пользователем, который реализует пункты, описанные в игровом процессе.

*Предлагаемый дизайн классов*:

* Класс TicTacToeGame: представляет саму игру с атрибутами, такими как доска и текущий игрок. Методы класса TicTacToeGame должны реализовывать логику игры, обработку ходов игроков и проверку на завершение игры.
* Класс Board: представляет игровую доску с атрибутами, такими как клетки и расположенные на них символы (Х, О или пустое место). Методы класса Board должны реализовывать отрисовку доски и обновление её после каждого хода.

***Проект «Простой текстовый квест» (\*)***

*Цель проекта*: создание захватывающей истории, которая позволит игрокам погрузиться в увлекательный мир квестов и принимать важные решения, влияющие на исход игры.

*Игровой процесс*:

* Начало игры: игрок запускает игру и вводит своё имя. После этого он попадает в историю и начинает своё приключение.
* Перемещение по сюжету: игроку предлагаются различные сценарии и ситуации, и он должен выбирать, как действовать, вводя соответствующую команду (например, «1» или «2»). Каждое решение влияет на дальнейший ход сюжета и может привести к разным исходам игры.
* Взаимодействие с окружением: игрок может взаимодействовать с различными предметами и персонажами в игровом мире, вводя соответствующие команды (например, «осмотреть комнату», «поговорить с торговцем» и т.д.).
* Сбор предметов: в ходе игры игрок может собирать различные предметы, которые могут пригодиться в дальнейшем при прохождении испытаний.
* Завершение игры: игра продолжается до тех пор, пока игрок не достигнет цели или не пройдет все испытания. В конце игры игроку предлагается оценка его решений и достижений.

*Внешний интерфейс*. Программа должна предоставлять текстовый интерфейс для взаимодействия с пользователем, который реализует пункты, описанные в игровом процессе.

*Предлагаемый дизайн классов*:

* Класс Player: представляет игрока с полями, такими как имя и инвентарь (список собранных предметов). Методы класса Player должны реализовывать ввод команд игроком и обработку его решений.
* Класс Scene: представляет сцену игры с полями, такими как текстовое описание и список возможных вариантов действий. Методы класса Scene должны реализовывать вывод текста на экран и предложение игроку выбрать вариант действия.
* Класс Item: представляет предметы в игровом мире с полями, такими как название и описание. Методы класса Item должны реализовывать взаимодействие игрока с предметами.

***Проект «Текстовые шахматы» (\*)***

*Цель проекта*: создание игры классические шахматы для двух игроков, которая позволит игрокам сыграть в шахматы через текстовый интерфейс в консоли.

*Игровой процесс*:

Начало игры: игрок запускает игру и видит текстовую доску 8x8, расположенную в консоли. Игроку предоставляется возможность выбрать цвет фигур, за которыми он будет играть (белые или черные).

Ходы игроков: игроки совершают свои ходы поочередно. Ход осуществляется путем ввода команды с указанием начальной и конечной позиции фигуры (например, «e2 e4» для сдвига пешки). После каждого хода доска перерисовывается с учетом последних изменений.

Правила игры: игра должна соблюдать все правила шахмат, такие как возможность ходить только определенными фигурами, невозможность хода внутри/сквозь другие фигуры, правила рокировки, взятия на проходе и т.д.

Конец игры: игра продолжается до тех пор, пока не наступит один из концов: мат, пат, ничья или запрос на завершение игры со стороны игрока.

*Внешний интерфейс*. Программа должна предоставлять текстовый интерфейс для взаимодействия с пользователем, который реализует пункты, описанные в игровом процессе.

*Предлагаемый дизайн классов*:

* Класс ChessGame: представляет саму игру с полями, такими как доска, игроки и текущий игрок. Методы класса ChessGame должны реализовывать логику игры, обработку ходов игроков и проверку на завершение игры.
* Класс Board: представляет шахматную доску с полями, такими как клетки и расположенные на них фигуры. Методы класса Board должны реализовывать отрисовку доски и обновление её после каждого хода.
* Класс Piece и унаследованные от неё конкретные фигуры (Pawn, Bishop, King, Queen, Knight, Rock): представляют отдельные фигуры на доске с полями, такими как тип фигуры и её текущая позиция. Методы класса Piece и его подклассов должны реализовывать возможные ходы фигур и проверку их на корректность.