1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. **Институт компьютерных наук и кибербезопасности**

**ЛАБОРАТОРНАЯ РАБОТА №2**

1. «Рефакторинг программы»
2. по дисциплине «Объектно-ориентированное программирование»
3. Выполнил
4. студент гр.5151001/20202 Сергеева Л. С.

<*подпись*>

1. Преподаватель Завадский Е.В.

<*подпись*>

1. Санкт-Петербург
2. 2024

# Цель работы

Целью работы – освоение особенностей этапов анализа предметной области и проектирования архитектуры программного обеспечения в объектной модели

# Задачи

В рамках выполнения лабораторной работы необходимо решить следующие задачи:

⎯ анализ предметной области заданной программы, реализованной в процедурной парадигме программирования с выделением ключевых абстракций;

⎯ анализ недостатков архитектуры и кода исходного проекта;

⎯ проектирование новой архитектуры программы в объектной парадигме программирования (рефакторинг) с использованием UML диаграммы классов (см. UML.docx);

⎯ программная реализация новой архитектуры на языке программирования Python или C++;

⎯ реализовать unit-тесты для основного функционала программы.

# Ход работы

## Описание исходного проекта

Исходный проект для рефакторинга был взят по ссылке: <https://github.com/yuraxdrumz/tank-game-with-python/blob/master/TankGame.py> .

Данная программа реализует игру «Танки».

**Предметная область:** видеоигры

**Функционал проекта:**

1. *Игровой процесс:* Однопользовательская игра, в которой игрок управляет танком и сражается против танка врага.
2. *Графика:* Использование Pygame для создания векторной графики и отображения танков, пушек и других элементов игры.
3. *Управление:* Возможность управления танком с клавиатуры, включая перемещение, смену высоты пушки и выстрела.
4. *Сражения:* Логика сражений, включая вычисление урона и проверку попаданий снарядов.
5. *Интерфейс:* Отображение полосок здоровья для танков, главное меню.

## Описание недостатков исходного кода

**Недостатки исходного кода:**

1. *Отсутствие типизации:* Механизм type hinting (аннотация типов данных) не используется, что снижает читаемость кода и затрудняет его сопровождение.
2. *Отсутствие документации:* Комментарии и документация к методам отсутствует. Это усложняет понимание логики кода.
3. *Недостаток модульности:* Весь код размещен в одном большом файле, что усложняет его изучение.
4. *Непонятые переменные:* Используются различные переменные в разных частях кода, которые трудно искать. Использование разных стилей наименований переменных.
5. *Повторяющийся код:* Присутствует дублирование кода, что увеличивает общий объем

## Описание разработанной архитектуры

Была разработана новая архитектура проекта, включающая в себя различные классы проекта. Ниже представлена UML-диаграмма новой реализации:

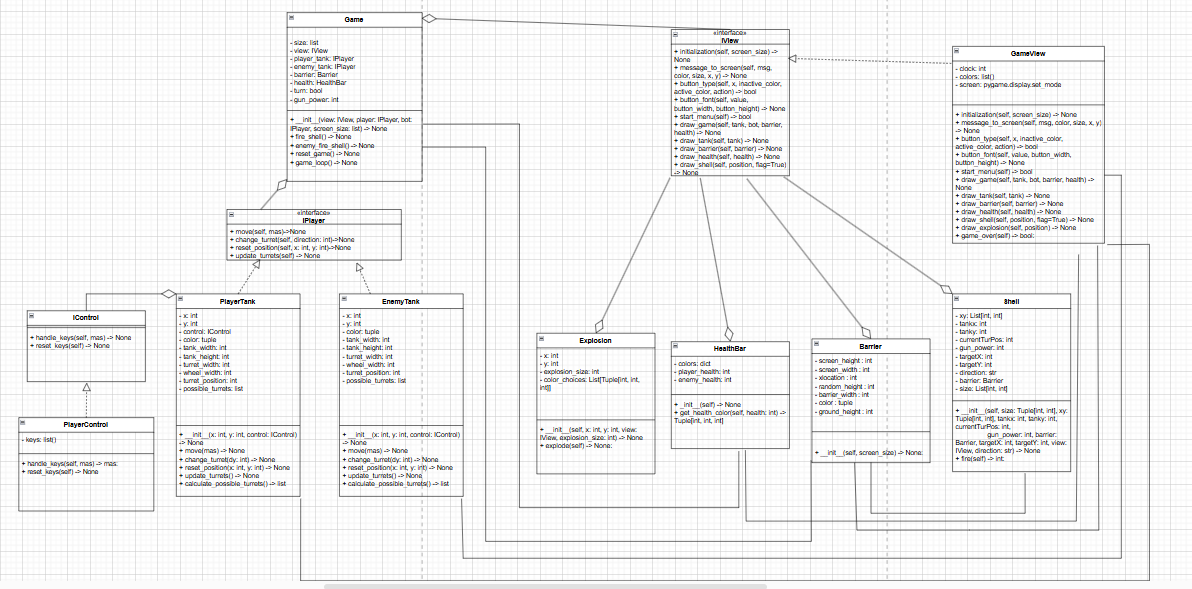


Рисунок 1 - UML – диаграмма

Таблица созданных классов и их описание приведена ниже:

Таблица 1. Классы проекта

|  |  |
| --- | --- |
| **Класс** | **Описание** |
| Menu (Game) | Класс Menu отвечает за отображение и управление меню игры. Он включает такие важные аспекты, как отрисовка элементов меню, обработка нажатий кнопок, а также взаимодействие с пользователем. |
| Barrier | Класс Barrier отвечает за создание преграды на игровом поле. Эта преграда взаимодействуют с танками и снарядами, добавляя элемент препятствия в игру. |
| HealthBar | Класс HealthBar отвечает за отображение и управление полосками здоровья для игрока и вражеского танка. Полоски здоровья показывают текущее состояние здоровья обоих участников и обновляются в реальном времени по мере получения урона. |
| PlayerTank | Класс PlayerTank наследует от базового класса Tank и реализует специфичное для игрока поведение в игре. Основная роль этого класса заключается в предоставлении методов и свойств, которые позволяют игроку управлять танком, взаимодействовать с игровым миром в процессе игры. |
| EnemyTank | Класс EnemyTank наследует от базового класса Tank и предназначен для представления вражеского танка в игре. Основная роль этого класса — моделировать поведение вражеского танка, добавляя элементы сложности и вызова для игрока. |
| Shell | Класс Shell отвечает за управление стрельбой снарядами в игре. Он моделирует поведение снарядов, их траекторию, взаимодействие с объектами на игровом поле и нанесение урона целям. |
| Explosion | Класс Explosion отвечает за визуальные и звуковые эффекты взрывов, которые происходят при попадании снарядов в цели или препятствия в игре. |

## Описание особенностей новой архитектуры

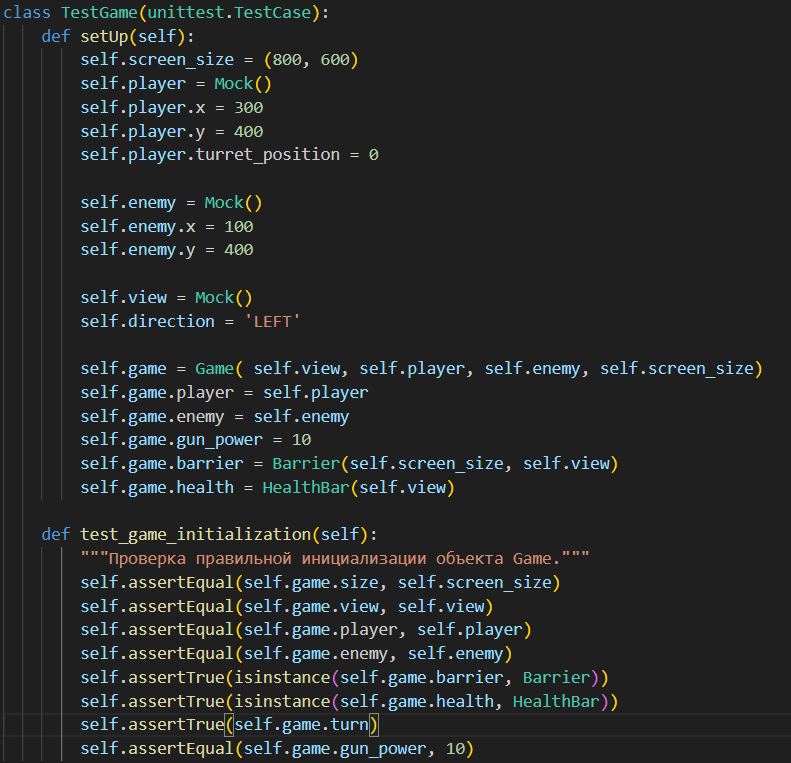
Проект представляет собой игру с использованием библиотеки Pygame для создания графических и звуковых эффектов. Вот основные особенности архитектуры проекта:

1. *Модульность:* Проект разделен на несколько модулей, каждый из которых отвечает за определенную функциональность. Например, есть модули для управления танками, обработки взрывов и управления игровым процессом.
2. *Инкапсуляция:* Классы и функции инкапсулируют логику, связанную с конкретными аспектами игры. Это облегчает поддержку и расширение кода.
3. *Использование Pygame:* Pygame используется для создания графических и звуковых эффектов. Это позволяет легко реализовать визуальные и аудиальные аспекты игры.
4. *Основные классы:* Проект включает несколько ключевых классов, таких как Tank, Shell, Explosion и Game. Каждый класс отвечает за свою часть игрового процесса.
5. *Геймплей:* Игра включает в себя управление танками, стрельбу снарядами и обработку столкновений. Взрывы и звуковые эффекты добавляют динамику и реализм к игровому процессу.
6. *Интерфейс пользователя:* Проект предоставляет простой и интуитивно понятный интерфейс пользователя, который позволяет легко управлять танками и взаимодействовать с игровым миром.

## Unit-тесты

Для проверки работоспособности некоторых функций были созданы unit – тесты.

### TestGame



В данном тестовом классе TestGame, осуществляется проверка корректной инициализации объекта Game. В методе setUp, который вызывается перед каждым тестом, создаются необходимые экземпляры объектов, которые будут использоваться в тестах. Определяются размеры экрана, позиция игрока и врага, а также создаются макеты для различных объектов, таких как view и enemy. После этого инициализируется экземпляр класса Game, которому передаются все ранее созданные объекты и параметры, такие как мощность оружия, барьер и индикатор здоровья.

Тестовый метод test\_game\_initialization проверяет, что после создания экземпляра Game все атрибуты инициализированы корректно. Это включает проверку соответствия размера экрана, ссылки на представление, положение игрока и врага, а также удостоверение, что объекты Barrier и HealthBar созданы и правильно назначены. Кроме того, тест подтверждает, что переменная turn установлена в значение True и что мощность оружия соответствует ожидаемому значению 10.

### TestPlayerTank

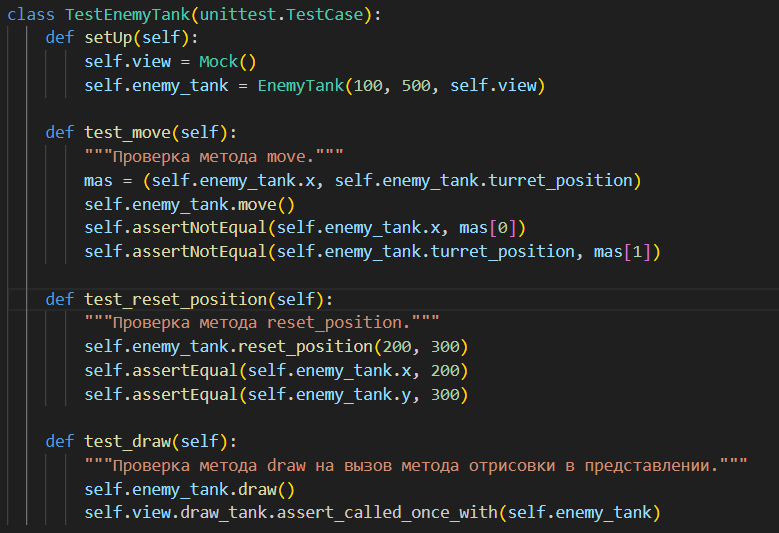


Тест test\_move проверяет работу метода move, который отвечает за изменение положения танка. В этом тесте передается вектор перемещения, и затем проверяется, что координата x танка изменяется на 5 единиц, а позиция турели обновляется до 8. Этот тест подтверждает, что метод move корректно обрабатывает изменения положения.

Тест test\_reset\_position оценивает функциональность метода reset\_position, который должен устанавливать положение танка на заданные координаты. В тесте проверяется, что после вызова метода reset\_position танк правильно перемещается в новые координаты (800, 600).

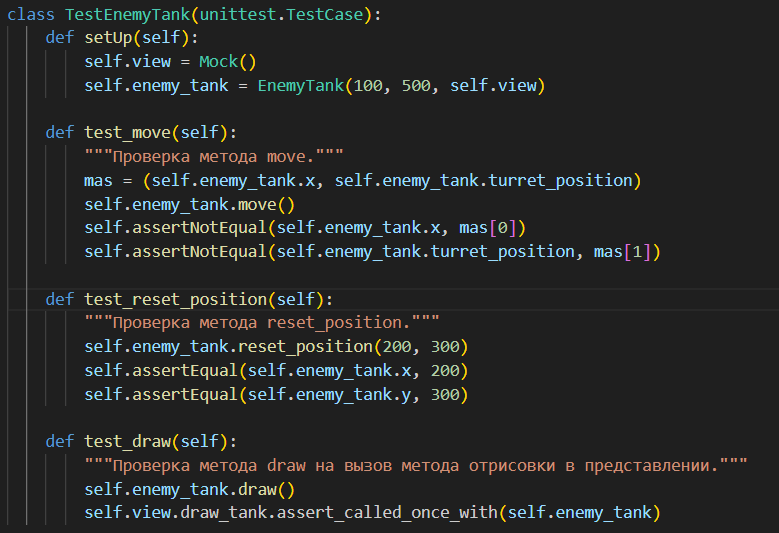
Тест test\_draw проверяет, что метод draw вызывает метод отрисовки танка в объекте view. При вызове draw у объекта view проверяется, что метод draw\_tank был вызван один раз с текущим экземпляром танка в качестве аргумента. Этот тест гарантирует, что визуализация танка осуществляется корректно.

### TestEnemyTank



Аналогично тестам TestPlayerTank.

### TestShell



В классе TestShell, проверяется функциональность класса Shell, особенно метод fire. В тесте используется декоратор @patch для создания макета интерфейса IView, что позволяет контролировать поведение метода отрисовки и проверять его вызовы.

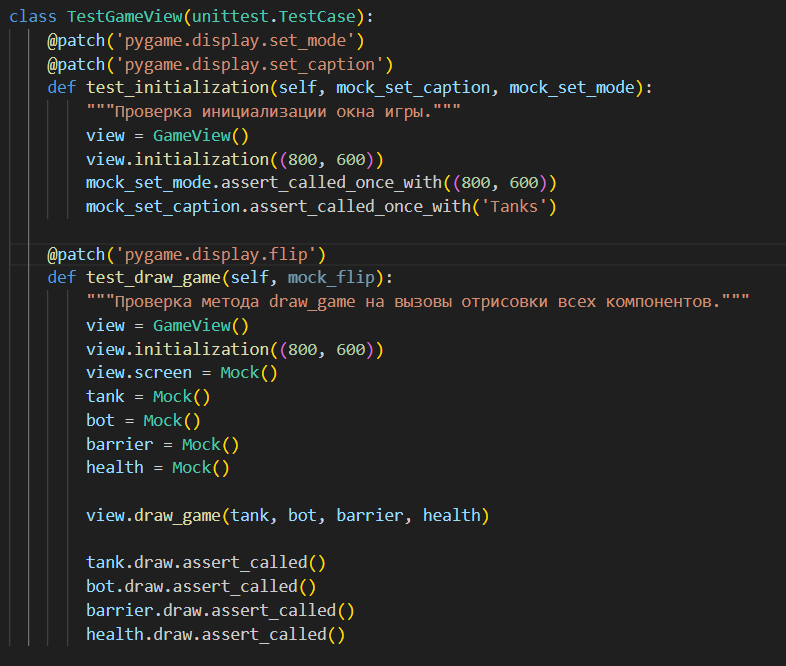
В методе test\_shell\_fire сначала создается экземпляр макета MockView, который будет использоваться для проверки взаимодействия с визуализацией. Затем создается макет барьера с определенными атрибутами, такими как высота земли и ширина барьера, что позволяет настроить условия для тестирования.

Далее создается экземпляр Shell, которому передаются необходимые параметры, включая размеры экрана, начальные координаты, позицию, мощность и ссылку на барьер, а также объект представления. После этого вызывается метод fire, который должен запускать поведение снаряда.

Наконец, тест проверяет, что метод draw\_shell на объекте view был вызван, что указывает на корректную интеграцию с системой визуализации. Это подтверждает, что метод fire выполняет свои функции правильно и инициирует соответствующий визуальный вывод.

Таким образом, данный тест обеспечивает уверенность в том, что метод fire в классе Shell корректно работает и взаимодействует с компонентами визуализации, вызывая нужные методы для отображения снаряда.

### TestGameView



Метод test\_initialization проверяет процесс инициализации окна игры. В этом тесте создается экземпляр GameView, а затем вызывается метод initialization с заданными размерами окна (800x600). После этого тест проверяет, что функции set\_mode и set\_caption были вызваны именно с этими параметрами, что подтверждает правильную настройку игрового окна.

Метод test\_draw\_game оценивает функциональность метода draw\_game, который отвечает за отрисовку всех игровых компонентов на экране. Сначала происходит инициализация игрового окна, после чего создаются макеты для различных игровых объектов: танка, бота, барьера и индикатора здоровья. Затем вызывается метод draw\_game с этими объектами в качестве аргументов.

Тест проверяет, что методы отрисовки для каждого из переданных объектов (draw) были вызваны, что указывает на корректное поведение метода draw\_game в отношении визуализации всех элементов игры.

Таким образом, оба теста в классе TestGameView обеспечивают уверенность в том, что класс GameView правильно инициализирует окно игры и корректно выполняет отрисовку всех игровых компонентов.

# Выводы

В ходе лабораторной работы была проанализирована процедурная программа, выявлены её недостатки (отсутствие модульности, документации, сложность кода), которые затрудняли понимание и сопровождение.

Была разработана новая архитектура программы на основе объектно-ориентированного подхода с использованием UML-диаграмм классов, что улучшило визуализацию структуры и связей между компонентами.

Реализация новой архитектуры на Python с использованием принципов инкапсуляции, наследования повысила читаемость и сопровождение кода.

Для проверки функциональности были проведены unit-тесты, охватывающие ключевые аспекты программы (инициализация, движение танков, стрельба).