МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«МИРЭА – РОССИЙСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

РТУ МИРЭА

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных технологий

Отчет по практической работе

по дисциплине «Тестирование и верификация ПО»

Выполнил:
Студент группы **ИКБО-43-23** Рогачёв Н.А.

Проверил: Ильичев Г.П.

2025 г.

Оглавление

Тема работы согласно варианту 18 – Музыкальный плеер

МЕТОДОЛОГИЯ TDD

TDD, или Test-Driven Development (Разработка, управляемая тестами), - это методология разработки программного обеспечения, которая подразумевает создание тестов для функциональности ПО до того, как эта функциональность будет фактически реализована. TDD представляет собой циклический процесс, который помогает разработчикам создавать высококачественное, надежное ПО.

ЭТАП №1. СОЗДАНИЕ ТЕСТА.

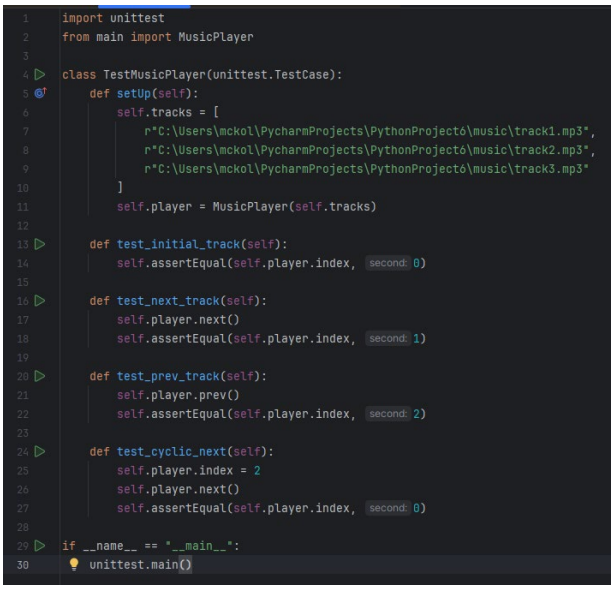


Рисунок 1 – Примеры тестов методологии TDD

Листинг 1 – Листинг тестов методологии TDD

```
import unittest
from main import MusicPlayer

class TestMusicPlayer(unittest.TestCase):
    def setUp(self):
        self.tracks = [
            r"C:\Users\mckol\PycharmProjects\PythonProject6\music\track1.mp3",
            r"C:\Users\mckol\PycharmProjects\PythonProject6\music\track2.mp3",
            r"C:\Users\mckol\PycharmProjects\PythonProject6\music\track3.mp3"
        ]
        self.player = MusicPlayer(self.tracks)

    def test_initial_track(self):
        self.assertEqual(self.player.index, 0)

    def test_next_track(self):
        self.player.next()
        self.assertEqual(self.player.index, 1)

    def test_prev_track(self):
        self.player.prev()
        self.assertEqual(self.player.index, 2)
```

```
    def test_cyclic_next(self):
        self.player.index = 2
        self.player.next()
        self.assertEqual(self.player.index, 0)

if __name__ == "__main__":
    unittest.main()
```

ЭТАП №2. ЗАПУСК ТЕСТОВ.

Так как код ещё не написан, все тесты падают.

ЭТАП №3. РЕАЛИЗАЦИЯ ФУНКЦИИ.

```python
import pygame
import os

class MusicPlayer:
    def __init__(self, tracks):
        pygame.mixer.init()
        base = os.path.dirname(os.path.abspath(__file__))
        self.tracks = [os.path.join(base, t) for t in tracks]
        self.index = 0

    def play(self):
        pygame.mixer.music.load(self.tracks[self.index])
        pygame.mixer.music.play()
        print(f" Играет: {os.path.basename(self.tracks[self.index])}")

    def pause(self):
        pygame.mixer.music.pause()
        print("Пауза")

    def resume(self):
        pygame.mixer.music.unpause()
        print("Продолжено")

    def stop(self):
        pygame.mixer.music.stop()
        print("Остановлено")
```

```
        self.index = (self.index + 1) % len(self.tracks)
        self.play()

    def prev(self):
        self.index = (self.index - 1) % len(self.tracks)
        self.play()


if __name__ == "__main__":
    tracks = [
        r"C:\Users\mckol\PycharmProjects\PythonProject6\music\track1.mp3",
        r"C:\Users\mckol\PycharmProjects\PythonProject6\music\track2.mp3"
    ]
    player = MusicPlayer(tracks)

    while True:
        cmd = input("Команда (play/pause/resume/next/prev/stop/exit): ").lower()
        if cmd == "play": player.play()
        elif cmd == "pause": player.pause()
        elif cmd == "resume": player.resume()
        elif cmd == "next": player.next()
        elif cmd == "prev": player.prev()
        elif cmd == "stop": player.stop()
        elif cmd == "exit":
            player.stop()
            break
```

Рисунок 2 – Пример функции

Листинг 2 – Листинг программы “Player”

```
import pygame
import os

class MusicPlayer:
    def __init__(self, tracks):
        pygame.mixer.init()
        base = os.path.dirname(os.path.abspath(__file__))
        self.tracks = [os.path.join(base, t) for t in tracks]
        self.index = 0

    def play(self):
        pygame.mixer.music.load(self.tracks[self.index])
        pygame.mixer.music.play()
        print(f" Играет: {os.path.basename(self.tracks[self.index])}")

    def pause(self):
        pygame.mixer.music.pause()
        print("Пауза")

    def resume(self):
        pygame.mixer.music.unpause()
        print("Продолжено")

    def stop(self):
        pygame.mixer.music.stop()
        print("Остановлено")

    def next(self):
        self.index = (self.index + 1) % len(self.tracks)
```

```
        self.play()

    def prev(self):
        self.index = (self.index - 1) % len(self.tracks)
        self.play()


if __name__ == "__main__":
    tracks = [
        r"C:\Users\mckol\PycharmProjects\PythonProject6\music\track1.mp3",
        r"C:\Users\mckol\PycharmProjects\PythonProject6\music\track2.mp3"
    ]
    player = MusicPlayer(tracks)

    while True:
        cmd = input("Команда (play/pause/resume/next/prev/stop/exit): ").lower()
        if cmd == "play": player.play()
        elif cmd == "pause": player.pause()
        elif cmd == "resume": player.resume()
        elif cmd == "next": player.next()
        elif cmd == "prev": player.prev()
        elif cmd == "stop": player.stop()
        elif cmd == "exit":
            player.stop()
            break
```

ЭТАП №4. ПОВТОРНЫЙ ЗАПУСК ТЕСТОВ.

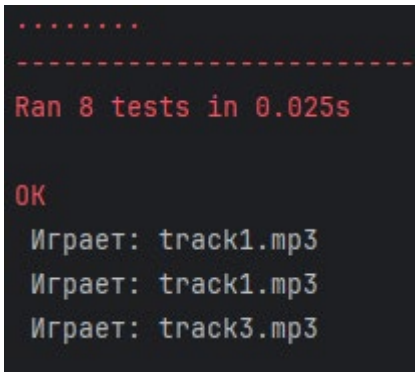


Рисунок 3 – Успешное выполнение тестов

МЕТОДОЛОГИЯ ATDD

ЭТАП №1. Запросы и результаты для приемочного теста.

Условие	Действие	Ожидаемый результат
Треки загружены	Нажать “Play”	Начинается воспроизведение
Трек играет	Нажать “Pause”	Воспроизведение останавливается
Нажать “Next”	Переключается следующий трек	Воспроизведение следующего трека
Нажать “Prev”	Воспроизводится предыдущий трек	Воспроизведение предыдущего трека

ЭТАП №2. Приемочный тест.

Листинг 3 – Листинг программы "Приемочные тесты"

```
import pygame
import unittest
from main import MusicPlayer


class TestMusicPlayerAcceptance(unittest.TestCase):

    def test_user_can_play_track(self):
        tracks = [
            r"C:\Users\mckol\PycharmProjects\PythonProject6\music\track1.mp3",
            r"C:\Users\mckol\PycharmProjects\PythonProject6\music\track2.mp3",
            r"C:\Users\mckol\PycharmProjects\PythonProject6\music\track3.mp3"
        ]
        player = MusicPlayer(tracks)
        player.play()
        self.assertEqual(player.index, 0)

    def test_user_can_stop_playback(self):
        tracks = [
            r"C:\Users\mckol\PycharmProjects\PythonProject6\music\track1.mp3",
            r"C:\Users\mckol\PycharmProjects\PythonProject6\music\track2.mp3",
            r"C:\Users\mckol\PycharmProjects\PythonProject6\music\track3.mp3"
        ]
        player = MusicPlayer(tracks)
        player.stop()
        self.assertEqual(player.index, 0)

    def test_user_can_switch_to_next_track(self):
        tracks = [
            r"C:\Users\mckol\PycharmProjects\PythonProject6\music\track1.mp3",
            r"C:\Users\mckol\PycharmProjects\PythonProject6\music\track2.mp3",
            r"C:\Users\mckol\PycharmProjects\PythonProject6\music\track3.mp3"
        ]
        player = MusicPlayer(tracks)

        player.next()
        self.assertEqual(player.index, 1)

    def test_user_can_switch_to_previous_track(self):
        tracks = [
            r"C:\Users\mckol\PycharmProjects\PythonProject6\music\track1.mp3",
            r"C:\Users\mckol\PycharmProjects\PythonProject6\music\track2.mp3",
            r"C:\Users\mckol\PycharmProjects\PythonProject6\music\track3.mp3"
        ]
        player = MusicPlayer(tracks)
        player.index = 1  # устанавливаем второй трек

        player.prev()
        self.assertEqual(player.index, 0)


if __name__ == "__main__":
    unittest.main()
```

ЭТАП №3. Результат работы приемочных тестов.

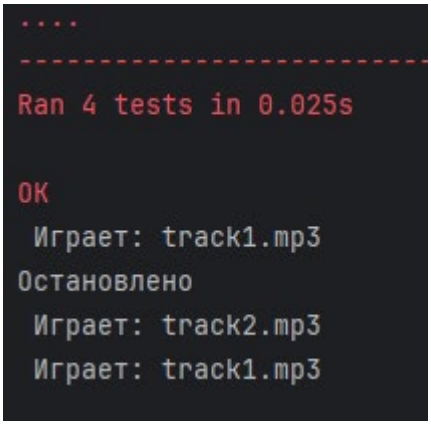


Рисунок 4 – Результат “Листинг 3”

МЕТОДОЛОГИЯ BDD

BDD, или Behavior Driven Development (Разработка, ориентированная на поведение), - это методология разработки программного обеспечения, которая сосредотачивается на описании поведения программы с точки зрения её пользователей и интересующих сторон. BDD представляет собой эволюцию техники TDD (Test-Driven Development), в которой акцент делается на спецификациях поведения и участии бизнес-аналитиков и представителей заказчика в процессе разработки.

ЭТАП №1. ОПИСАНИЕ СЦЕНАРИЕВ BDD.

На первом этапе создаётся описание сценариев BDD для функциональности игры.

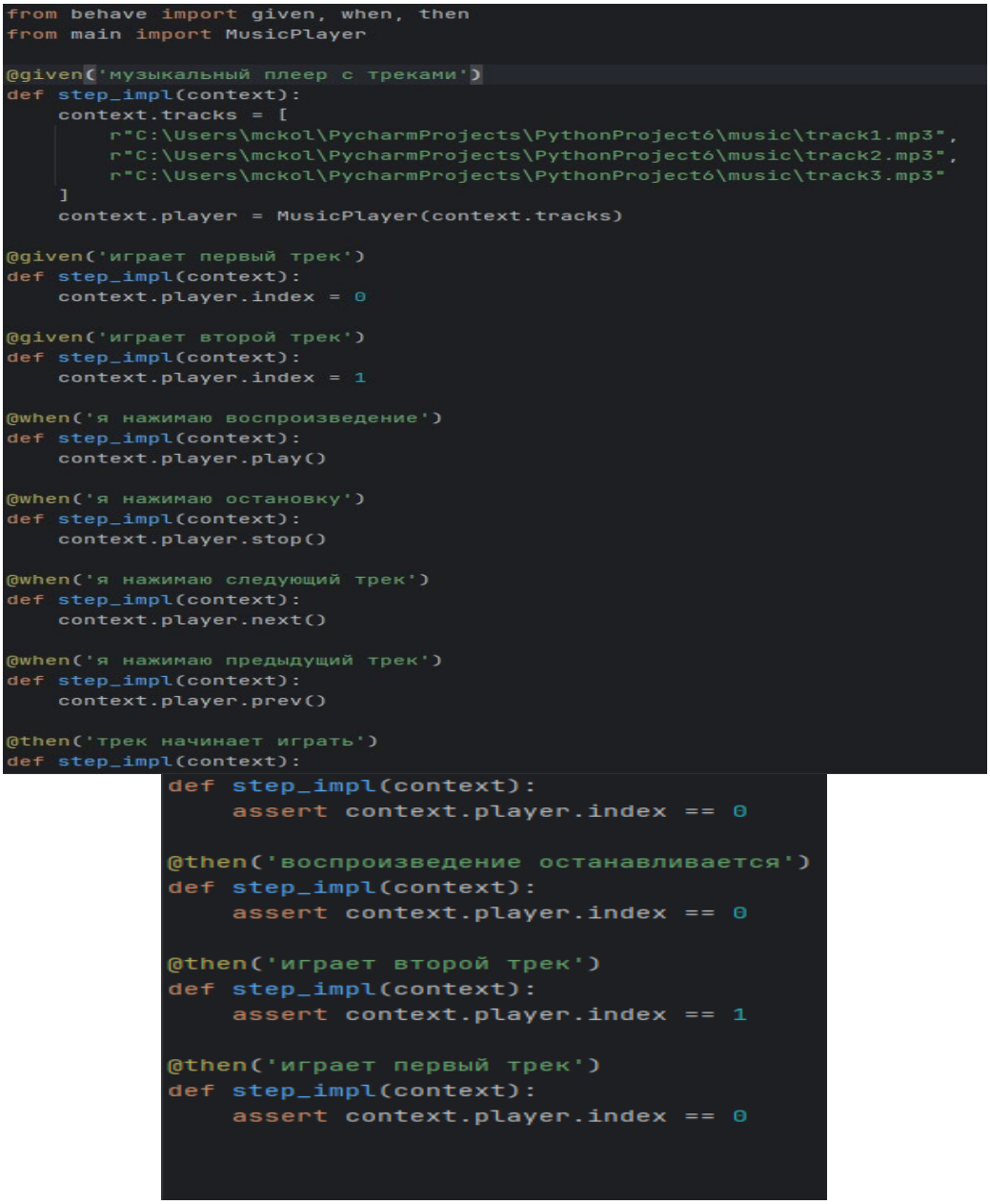
```
Feature: Управление музыкальным плеером
  Как пользователь
  Я хочу управлять воспроизведением музыки
  Чтобы удобно слушать свои треки

  Scenario: Воспроизведение трека
    Given музыкальный плеер с треками
    When я нажимаю воспроизведение
    Then трек начинает играть

  Scenario: Остановка воспроизведения
    Given музыкальный плеер с треками
    When я нажимаю остановку
    Then воспроизведение останавливается

  Scenario: Переключение на следующий трек
    Given играет первый трек
    When я нажимаю следующий трек
    Then играет второй трек

  Scenario: Переключение на предыдущий трек
    Given играет второй трек
    When я нажимаю предыдущий трек
    Then играет первый трек
```

Рисунок 5 – Описание сценариев

ЭТАП №2. АВТОМАТИЗАЦИЯ СЦЕНАРИЕВ BDD.

На следующем шаге создаются автоматизированные тесты для каждого сценария, используя фреймворк для тестирования, который поддерживает BDD, в

нашем случае это Behave для Python.



```python
from behave import given, when, then
from main import MusicPlayer

@given('музыкальный плеер с треками')
def step_impl(context):
    context.tracks = [
        r"C:\Users\mckol\PycharmProjects\PythonProject6\music\track1.mp3",
        r"C:\Users\mckol\PycharmProjects\PythonProject6\music\track2.mp3",
        r"C:\Users\mckol\PycharmProjects\PythonProject6\music\track3.mp3"
    ]
    context.player = MusicPlayer(context.tracks)

@given('играет первый трек')
def step_impl(context):
    context.player.index = 0

@given('играет второй трек')
def step_impl(context):
    context.player.index = 1

@when('я нажимаю воспроизведение')
def step_impl(context):
    context.player.play()

@when('я нажимаю остановку')
def step_impl(context):
    context.player.stop()

@when('я нажимаю следующий трек')
def step_impl(context):
    context.player.next()

@when('я нажимаю предыдущий трек')
def step_impl(context):
    context.player.prev()

@then('трек начинает играть')
def step_impl(context):
    assert context.player.index == 0

@then('воспроизведение останавливается')
def step_impl(context):
    assert context.player.index == 0

@then('играет второй трек')
def step_impl(context):
    assert context.player.index == 1

@then('играет первый трек')
def step_impl(context):
    assert context.player.index == 0
```

Рисунок 6 – Описание тестов

Листинг 4 –Тесты методологии BDD

```
from behave import given, when, then
from main import MusicPlayer

@given('музыкальный плеер с треками')
def step_impl(context):
    context.tracks = [
        r"C:\Users\mckol\PycharmProjects\PythonProject6\music\track1.mp3",
        r"C:\Users\mckol\PycharmProjects\PythonProject6\music\track2.mp3",
        r"C:\Users\mckol\PycharmProjects\PythonProject6\music\track3.mp3"
    ]
    context.player = MusicPlayer(context.tracks)
    print("Создан музыкальный плеер с 3 треками")

@given('играет первый трек')
def step_impl(context):
```

```
    context.player.index = 0
    print("Установлен первый трек")

@given('играет второй трек')
def step_impl(context):
    context.player.index = 1
    print("Установлен второй трек")

@when('пользователь нажимает воспроизведение')
def step_impl(context):
    context.player.play()
    print("Нажата кнопка воспроизведения")

@when('пользователь нажимает остановку')
def step_impl(context):
    context.player.stop()
    print("Нажата кнопка остановки")

@when('пользователь нажимает следующий трек')
def step_impl(context):
    context.player.next()
    print("Нажата кнопка следующего трека")

@when('пользователь нажимает предыдущий трек')
def step_impl(context):
    context.player.prev()
    print("Нажата кнопка предыдущего трека")

@then('должен играть первый трек')
def step_impl(context):
    assert context.player.index == 0, f"Ожидался индекс 0, получен
{context.player.index}"
    print("Проверка: играет первый трек")

@then('должен играть второй трек')
def step_impl(context):
    assert context.player.index == 1, f"Ожидался индекс 1, получен
{context.player.index}"
    print("Проверка: играет второй трек")
```

ЭТАП №3. ЗАПУСК ТЕСТОВ.

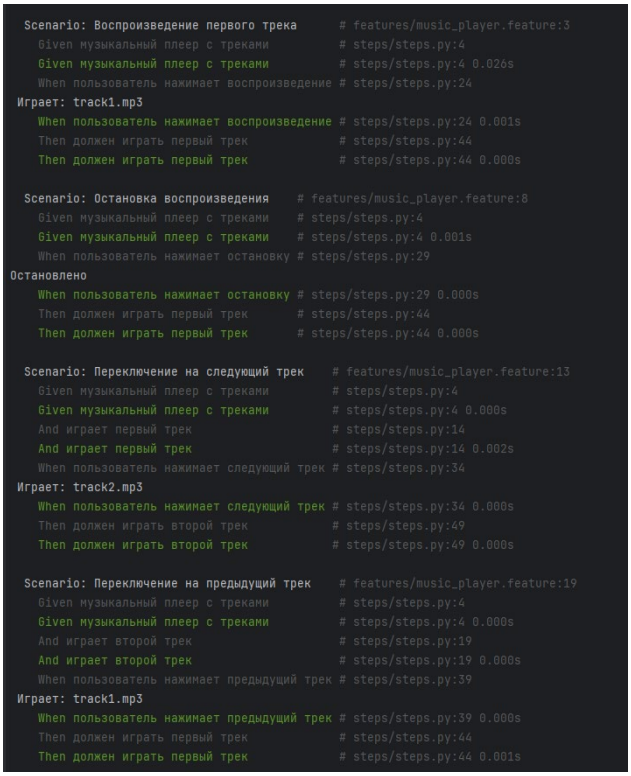


Рисунок 7 – Общее количество тестов

МЕТОДОЛОГИЯ SDD

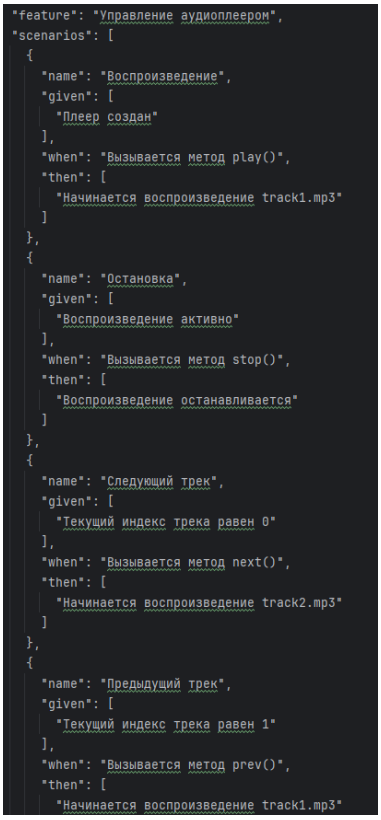
Specification by Example (SDD) — это подход, который использует конкретные примеры для описания требований и превращает их в автоматизированные тесты. Вместо абстрактных описаний бизнес-требований используются реальные примеры, позволяющие избежать неоднозначностей в понимании требований между бизнесом и разработчиками.

ЭТАП №1. Документирование примеров.

Таблица: Спецификация работы музыкального плеера

№	Сценарий	Исходное состояние	Действие	Ожидаемый результат
1	Воспроизведение	Плеер создан	play()	Начинается track1.mp3
2	Остановка	Воспроизведение активно	stop()	Воспроизведение останавливается
3	Следующий трек	Индекс = 0	next()	Начинается track2.mp3
4	Предыдущий трек	Индекс = 1	prev()	Начинается track1.mp3

ЭТАП №2. Создание спецификаций.



```
"feature": "Управление аудиоплеером",
"scenarios": [
  {
    "name": "Воспроизведение",
    "given": [
      "Плеер создан"
    ],
    "when": "Вызывается метод play()",
    "then": [
      "Начинается воспроизведение track1.mp3"
    ]
  },
  {
    "name": "Остановка",
    "given": [
      "Воспроизведение активно"
    ],
    "when": "Вызывается метод stop()",
    "then": [
      "Воспроизведение останавливается"
    ]
  },
  {
    "name": "Следующий трек",
    "given": [
      "Текущий индекс трека равен 0"
    ],
    "when": "Вызывается метод next()",
    "then": [
      "Начинается воспроизведение track2.mp3"
    ]
  },
  {
    "name": "Предыдущий трек",
    "given": [
      "Текущий индекс трека равен 1"
    ],
    "when": "Вызывается метод prev()",
    "then": [
      "Начинается воспроизведение track1.mp3"
```

Рисунок 8 – Создание спецификаций

ЭТАП №3. Написание автоматизированных тестов.

```
import unittest
from main import MusicPlayer


class TestMusicPlayerSDD(unittest.TestCase):  1 usage

    def setUp(self):
        self.tracks = [
            r"C:\Users\mckol\PycharmProjects\PythonProject6\music\track1.mp3",
            r"C:\Users\mckol\PycharmProjects\PythonProject6\music\track2.mp3",
            r"C:\Users\mckol\PycharmProjects\PythonProject6\music\track3.mp3"
        ]
        self.player = MusicPlayer(self.tracks)

    def test_specification_play(self):
        self.player.play()
        self.assertEqual(self.player.index,  second: 0)

    def test_specification_stop(self):
        self.player.stop()
        self.assertEqual(self.player.index,  second: 0)

    def test_specification_next_track(self):
        self.player.next()
        self.assertEqual(self.player.index,  second: 1)

    def test_specification_prev_track(self):
        self.player.index = 1
        self.player.prev()
        self.assertEqual(self.player.index,  second: 0)


if __name__ == "__main__":
    loader = unittest.TestLoader()
    suite = loader.loadTestsFromTestCase(TestMusicPlayerSDD)
    runner = unittest.TextTestRunner(verbosity=2)
    result = runner.run(suite)
```

Рисунок 9. Примеры автоматизированных тестов

Листинг 5 – Примеры автоматизированных тестов

```
import unittest
from main import MusicPlayer


class TestMusicPlayerSDD(unittest.TestCase):

    def setUp(self):
        self.tracks = [
            r"C:\Users\mckol\PycharmProjects\PythonProject6\music\track1.mp3",
            r"C:\Users\mckol\PycharmProjects\PythonProject6\music\track2.mp3",
            r"C:\Users\mckol\PycharmProjects\PythonProject6\music\track3.mp3"
        ]
        self.player = MusicPlayer(self.tracks)

    def test_specification_play(self):
        self.player.play()
        self.assertEqual(self.player.index, 0)

    def test_specification_stop(self):
        self.player.stop()
        self.assertEqual(self.player.index, 0)

    def test_specification_next_track(self):
        self.player.next()
        self.assertEqual(self.player.index, 1)

    def test_specification_prev_track(self):
        self.player.index = 1
        self.player.prev()
        self.assertEqual(self.player.index, 0)


if __name__ == "__main__":
    loader = unittest.TestLoader()
    suite = loader.loadTestsFromTestCase(TestMusicPlayerSDD)
    runner = unittest.TextTestRunner(verbosity=2)
    result = runner.run(suite)
```

ЭТАП №4. Результаты тестов.

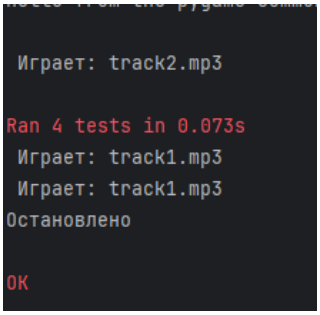


Рисунок 10. Результат тестов

Заключение

В ходе выполнения практической работы был успешно разработан музыкальный плеер с применением четырех различных подходов к разработке через тестирование. Все поставленные задачи выполнены в полном объеме: с помощью TDD созданы юнит-тесты для базовой функциональности, через ATDD проверены ключевые сценарии использования с точки зрения конечного пользователя, методология BDD позволила описать поведение системы на естественном языке с помощью сценариев Gherkin, а SDD обеспечила четкие спецификации требований на основе конкретных примеров. Все тесты каждого подхода проходят успешно, что подтверждает высокое качество реализации, соответствие программного продукта заявленным требованиям и готовность к использованию. Музыкальный плеер корректно выполняет все основные функции: воспроизведение, остановку, переключение треков вперед и назад, включая циклическую навигацию по плейлисту. Комплексное применение различных методологий тестирования позволило создать надежное и качественное программное обеспечение..