МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ**

**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»**

**(БГТУ им. В. Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

**Лабораторная работа № 14**

по дисциплине: Объектно-ориентированное программирование

тема: «Тестирование. Знакомство с TDD. Тесты как способ формирования архитектуры»

Выполнил: ст. группы ПВ-223

Игнатьев Артур Олегович

Проверил:

асс. Черников Сергей Викторович

Белгород 2024г.

**Лабораторная работа №14**

**«Знакомство с библиотеками Python. PyQT»**

**Цель работы:** знакомство с понятием тестирования. Получение практических навыков для написания модельных тестов.

**Вариант 3**

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Тесты для начального условия:

import pytest

from LOOP\_14\_1 import Door, Room, main

import sys

def test\_initial\_door\_state():

    door = Door()

    assert not door.is\_open

def test\_open\_door():

    door = Door()

    door.open\_door()

    assert door.is\_open

def test\_close\_door():

    door = Door(is\_open=True)

    door.close\_door()

    assert not door.is\_open

Код для начальных условий задач:

class Door:

    def \_\_init\_\_(self, is\_open=False):

        self.is\_open = is\_open

    def open\_door(self):

        self.is\_open = True

        print("Дверь открыта")

    def close\_door(self):

        self.is\_open = False

        print("Дверь закрыта")

class Room:

    def \_\_init\_\_(self, number):

        self.number = number

def main():

    room1 = Room(1)

    room2 = Room(2)

    door = Door()

    choice = input("Выберите действие: 1 - войти в помещение 2, 2 - остаться в этом помещении: ")

    if choice == "1":

        if not door.is\_open:

            door.open\_door()

            print("Пользователь вошел в помещение 2")

        else:

            print("Дверь уже открыта")

    elif choice == "2":

        print("Пользователь остается в этом помещении. Дверь закрыта")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

Тесты для финальных условий:

import pytest

from LOOP\_14\_2 import Door, LockType

def test\_door\_init\_defaults():

    door = Door()

    assert not door.is\_open

    assert door.lock\_type == LockType.MECHANICAL

    assert door.password == "123"

def test\_open\_door():

    door = Door()

    door.open\_door()

    assert door.is\_open

def test\_close\_door():

    door = Door(is\_open=True)

    door.close\_door()

    assert not door.is\_open

def test\_set\_lock\_type\_digital():

    door = Door()

    door.set\_lock\_type(LockType.DIGITAL)

    assert door.lock\_type == LockType.DIGITAL

def test\_set\_lock\_type\_mechanical():

    door = Door(lock\_type=LockType.DIGITAL)

    door.set\_lock\_type(LockType.MECHANICAL)

    assert door.lock\_type == LockType.MECHANICAL

def test\_reset\_password\_digital():

    door = Door(lock\_type=LockType.DIGITAL)

    door.reset\_password('new\_pass')

    assert door.password == 'new\_pass'

def test\_reset\_password\_mechanical():

    door = Door(lock\_type=LockType.MECHANICAL)

    door.reset\_password('new\_pass')

    assert door.password == '123'

Код для финальных условий:

from enum import Enum

class LockType(Enum):

    DIGITAL = 'digital'

    MECHANICAL = 'mechanical'

class Door:

    def \_\_init\_\_(self, is\_open=False, lock\_type=LockType.MECHANICAL, password=None):

        self.is\_open = is\_open

        self.lock\_type = lock\_type

        self.password = password if password else "123"

    def open\_door(self):

        self.is\_open = True

        print("Дверь открыта")

    def close\_door(self):

        self.is\_open = False

        print("Дверь закрыта")

    def set\_lock\_type(self, lock\_type):

        if lock\_type == LockType.DIGITAL:

            print("Цифровой замок успешно установлен")

        elif lock\_type == LockType.MECHANICAL:

            print("Механический замок успешно установлен")

        self.lock\_type = lock\_type

    def check\_password(self, entered\_pass):

        if self.lock\_type == LockType.DIGITAL and self.password == entered\_pass:

            print("Пароль принят, проходите")

        else:

            print("Пароль не верный, дверь остается закрытой")

    def reset\_password(self, new\_password):

        if self.lock\_type == LockType.DIGITAL:

            self.password = new\_password

            print("Пароль успешно сброшен")

class Room:

    def \_\_init\_\_(self, number):

        self.number = number

def main():

    room1 = Room(1)

    room2 = Room(2)

    choice = input("Выберите действие: 1 - войти в соседнюю комнату, 2 - остаться в текущей комнате: ")

    door = None

    if choice == "1":

        door = Door()

        lock\_type\_input = input("Выберите тип замка на двери (digital/mechanical): ")

        lock\_type = LockType.DIGITAL if lock\_type\_input == 'digital' else LockType.MECHANICAL

        door.set\_lock\_type(lock\_type)

        if lock\_type == LockType.DIGITAL:

            password = input("Введите пароль от замка: ")

            door.check\_password(password)

            reset\_password = input("Желаете изменить пароль от цифрового замка? y/n:")

            if reset\_password.lower() == "y":

                new\_pass = input("Введите новый пароль: ")

                door.reset\_password(new\_pass)

        if not door.is\_open:

            door.open\_door()

            print("Вы вошли в соседнюю комнату")

    elif choice == "2":

        door = Door(is\_open=False) # Дверь закрыта, если пользователь остается в текущей комнате

        print("Дверь закрыта, вы остаетесь в текущей комнате")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

Результат работы программы:



Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**Вывод:** на этой лабораторной работе ознакомился с понятием тестирования. Получил практические навыки для написания модельных тестов