**Московский государственный технический университет им. Н.Э. Баумана**

Факультет «Радиотехнический»

Кафедра ИУ5 «Системы обработки информации и управления»

Отчёт по лабораторной работе №4.

“ Шаблоны проектирования и модульное тестирование в Python”

|  |  |
| --- | --- |
| Выполнил: | Проверил: |
| студент группы РТ5-31Б | преподаватель каф. ИУ5 |
| Агеев Алексей | Гапанюк Ю.Е. |
| Подпись и дата: | Подпись и дата: |

Москва, 2021 г

# Описание задания

1. Необходимо для произвольной предметной области реализовать от одного до трех шаблонов проектирования: один порождающий, один структурный и один поведенческий. В качестве справочника шаблонов можно использовать [следующий каталог.](https://refactoring.guru/ru/design-patterns/catalog) Для сдачи лабораторной работы в минимальном варианте достаточно реализовать один паттерн.
2. Вместо реализации паттерна Вы можете написать тесты для своей программы решения биквадратного уравнения. В этом случае, возможно, Вам потребуется доработать программу решения биквадратного уравнения, чтобы она была пригодна для модульного тестирования.
3. В модульных тестах необходимо применить следующие технологии:
   * TDD - фреймворк.
   * BDD - фреймворк.
   * Создание Mock-объектов.

# Текст программы

from \_\_future\_\_ import annotations

from abc import ABC, abstractmethod

class AbstractCarFactory(ABC):

    @abstractmethod

    def ProduceCar(): pass

class CarFactory(AbstractCarFactory):

    def ProduceCar(type\_car) -> Car:

        if type\_car == "Racing\_Car":

            return Racing\_Car

        elif type\_car == "Trunc":

            return Trunc

class TunedCarFactory(AbstractCarFactory):

    def ProduceCar(type\_car) -> Car:

        if type\_car == "Racing\_Car":

            return Tuned\_Racing\_Car

        elif type\_car == "Trunc":

            return Tuned\_Trunc

class Car(ABC):

    @abstractmethod

    def Drive(): pass

class Racing\_Car(Car):

    def Drive():

        print("Едешь на спорткаре")

class Trunc(Car):

    def Drive():

        print("Едешь на грузовике")

class Tuned\_Racing\_Car(Car):

    def Drive():

        print("Едешь на тюнингованном спорткаре")

class Tuned\_Trunc(Car):

    def Drive():

        print("Едешь на тюнингованном грузовике")

class Factory(ABC):

    def GetFactory(tuning = False):

        if tuning:

            return TunedCarFactory

        else:

            return CarFactory

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

    factory1 = Factory.GetFactory()

    car11 = factory1.ProduceCar("Racing\_Car")

    car12 = factory1.ProduceCar("Trunc")

    car11.Drive()

    car12.Drive()

    print()

    factory2 = Factory.GetFactory(True)

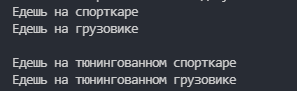
    car21 = factory2.ProduceCar("Racing\_Car")

    car22 = factory2.ProduceCar("Trunc")

    car21.Drive()

    car22.Drive()

**Пример работы программы.**

****