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

Факультет «Радиотехнический»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий» Отчёт по Лабораторной работе №4

Вариант №15

Выполнил: Проверил:

студент группы РТ5-31Б преподаватель каф. ИУ5

Мицкевич Владислав 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

* fabric.py

import math

from abc import ABC, abstractmethod import unittest

import sys import os

conf\_path = os.getcwd() print(conf\_path) sys.path.append(conf\_path) from Tests.tdd import \*

class Creator(ABC): @abstractmethod

def factory\_method(self): pass

class CarCreator(Creator): def factory\_method(self):

return Car()

class BikeCreator(Creator): def factory\_method(self):

return Bike()

class Transport(ABC): @abstractmethod

def deliver(self, \*args):

pass

@abstractmethod

def take\_item(self, \*args): pass

@abstractmethod def count\_time(self):

pass

class Car(Transport): # Грузовик speed = 10 # скорость

dist = None # расстояние item = None # Товар

def deliver(self, dist): self.dist = dist

def take\_item(self, item): self.item = item

def count\_time(self):

return math.ceil(self.dist / self.speed) # округление в большую сторону

class Bike(Transport): # Корабль speed = 5 # скорость

dist = None # расстояние item = None # Товар

def deliver(self, dist): self.dist = dist

def take\_item(self, item): self.item = item

def count\_time(self):

return math.ceil(self.dist / self.speed) # округление в большую сторону

if name == ' main ': unittest.main()

* tdd.py

import unittest

from Lab4.fabric import \*

class MyTestCase(unittest.TestCase): car = None

bike = None item1 = "Table" item2 = "Book" dist1 = 100

dist2 = 77 @classmethod

def setUp(self):

self.car = CarCreator().factory\_method() self.car.take\_item(self.item1) self.car.deliver(self.dist1)

self.bike = BikeCreator().factory\_method() self.bike.take\_item(self.item2) self.bike.deliver(self.dist2)

def test\_not\_none(self): self.assertIsNotNone(self.bike) self.assertIsNotNone(self.car)

def test\_item(self): self.assertEqual(self.car.item, self.item1) self.assertEqual(self.bike.item, self.item2)

def test\_upper(self): self.assertTrue(self.car.count\_time() \* self.car.speed

>= self.car.dist) self.assertTrue(self.bike.count\_time() \* self.bike.speed

>= self.bike.dist)

def test\_instance(self): self.assertIsInstance(self.car, Transport) self.assertIsInstance(self.bike, Transport)

@classmethod

def tearDownClass(self): del self.bike

del self.car

# Вывод программы

