**Министерство науки и высшего образования Российской Федерации**
**Федеральное государственное бюджетное образовательное учреждение высшего образования**
**«Московский государственный технический университет имени Н.Э. Баумана**
**(национальный исследовательский университет)»**
**(МГТУ им. Н.Э. Баумана)**

**Факультет «ГУИМЦ»**

**Кафедра ИУ5 «Системы обработки информации и управления»**

Дисциплина «Базовые компоненты ИТ»

# ОТЧЕТ

Лабораторная работа №4

Студент: Федюнев А. Ю., группа ИУ5Ц-52Б

Преподаватель: Гапанюк Ю. Е.

2021г.

## 1. 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. 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.

## 2. Текст программы

**builder.py**

```
from abc import ABC, abstractmethod
from enum import Enum, auto

class Airplane(Enum):
    AirbusA220 = auto()
    AirbusA320 = auto()
    Boeing717 = auto()
    Boeing777X = auto()

class Where(Enum):
    Moscow = auto()
    Saratov = auto()
    Saint_Petersburg = auto()
    Samara = auto()
    Perm = auto()

class Wherefrom(Enum):
    Moscow = auto()
    Saratov = auto()
    Saint_Petersburg = auto()
    Samara = auto()
    Perm = auto()

class Company(Enum):
    Utair = auto()
    Aeroflot = auto()
    Pobeda = auto()
    Nordavia = auto()
    Komiavia = auto()

class Terminal(Enum):
    A = auto()
    B = auto()
```

```
    C = auto()
    D = auto()
    E = auto()


class Time(Enum):
    time12 = auto()
    time15 = auto()
    time18 = auto()
    time21 = auto()
    time00 = auto()

class Avia:
    def __init__(self, id):
        self.id = id
        self.airplane = None
        self.where = None
        self.wherefrom = None
        self.company = None
        self.terminal = None
        self.gate = None
        self.time = None
        self.cost = None

    def __str__(self):
        info: str = f"Avia ticket: {self.id} \n" \
                    f"{self.airplane} \n" \
                    f"{self.where} \n" \
                    f"{self.wherefrom} \n" \
                    f"{self.company} \n" \
                    f"{self.terminal} \n" \
                    f"{self.gate} \n" \
                    f"{self.time} \n" \
                    f"Cost: {self.cost} rub"
        return info

class Builder(ABC):

    @abstractmethod
    def add_airplane(self) -> None: pass

    @abstractmethod
    def add_where(self) -> None: pass

    @abstractmethod
    def add_wherefrom(self) -> None: pass

    @abstractmethod
    def add_company(self) -> None: pass

    @abstractmethod
    def add_terminal(self) -> None: pass

    @abstractmethod
    def add_time(self) -> None: pass

class MoscowPerm(Builder):
    def __init__(self):
        self.AviaTicket = Avia("MoscowPerm")
        self.AviaTicket.gate = 1
        self.AviaTicket.cost = 3566

    def add_airplane(self) -> None:
        self.AviaTicket.airplane = Airplane.AirbusA220
```

```
    def add_where(self) -> None:
        self.AviaTicket.where = Where.Perm

    def add_wherefrom(self) -> None:
        self.AviaTicket.wherefrom = Wherefrom.Moscow

    def add_company(self) -> None:
        self.AviaTicket.company = Company.Aeroflot

    def add_time(self) -> None:
        self.AviaTicket.time = Time.time18

    def add_terminal(self) -> None:
        self.AviaTicket.terminal = Terminal.C

    def get_lap(self) -> Avia:
        return self.AviaTicket

class PermSaratov(Builder):
    def __init__(self):
        self.AviaTicket = Avia("PermSaratov")
        self.AviaTicket.gate = 3
        self.AviaTicket.cost = 5000

    def add_airplane(self) -> None:
        self.AviaTicket.airplane = Airplane.Boeing777X

    def add_where(self) -> None:
        self.AviaTicket.where = Where.Saratov

    def add_wherefrom(self) -> None:
        self.AviaTicket.wherefrom = Wherefrom.Perm

    def add_company(self) -> None:
        self.AviaTicket.company = Company.Utair

    def add_time(self) -> None:
        self.AviaTicket.time = Time.time12

    def add_terminal(self) -> None:
        self.AviaTicket.terminal = Terminal.A

    def get_lap(self) -> Avia:
        return self.AviaTicket

class Director:
    def __init__(self):
        self.builder = None

    def set_builder(self, builder: Builder):
        self.builder = builder

    def make_lap(self):
        if not self.builder:
            raise ValueError("Builder didn't set")
        self.builder.add_airplane()
        self.builder.add_where()
        self.builder.add_wherefrom()
        self.builder.add_company()
        self.builder.add_terminal()
        self.builder.add_time()

def check_gate(id1):
```

```
    for al in (MoscowPerm, PermSaratov):
        director1 = Director()
        builder1 = al()
        director1.set_builder(builder1)
        director1.make_lap()
        AviaTicket1 = builder1.get_lap()
        if AviaTicket1.id == id1:
            return AviaTicket1.gate

def check_sum(x):
    for al in (MoscowPerm, PermSaratov):
        director1 = Director()
        builder1 = al()
        director1.set_builder(builder1)
        director1.make_lap()
        AviaTicket1 = builder1.get_lap()
        x += x + AviaTicket1.cost
    return x

if __name__ == "__main__":
    print("Объекты")
    director = Director()
    for el in (MoscowPerm, PermSaratov):
        builder = el()
        director.set_builder(builder)
        director.make_lap()
        AviaTicket = builder.get_lap()
        print(AviaTicket)
        print('------------------------------')
    id = "MoscowPerm"
    print(id, "gate: ", check_gate(id))
    x = 0
    print('sum = ', check_sum(x))
```

```
D:\python\python.exe D:/pystdy/pattern/builder.py
Объекты
Avia ticket: MoscowPerm
Airplane.AirbusA220
Where.Perm
Wherefrom.Moscow
Company.Aeroflot
Terminal.C
1
Time.time18
Cost: 3566 rub
------------------------------
Avia ticket: PermSaratov
Airplane.Boeing777X
Where.Saratov
Wherefrom.Perm
Company.Utair
Terminal.A
3
Time.time12
Cost: 5000 rub
------------------------------
MoscowPerm gate:  1
sum =  12132

Process finished with exit code 0
```

**mock.py**

```python
from ..builder import *
from unittest import TestCase
from unittest.mock import patch

class Testgate(TestCase):
    @patch('builder.check_sum', return_value=12132)
    def test_sum_cost(self, x):
        self.assertEqual(check_sum(0), 12132)
```

**tdd.py**

```python
import unittest
import sys, os

sys.path.append(os.getcwd())
from ..builder import *

class Testgate(unittest.TestCase):
```

```
    def test_gate(self):
        self.assertEqual(check_gate("MoscowPerm"), 1)

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

```
D:\python\python.exe "D:\PyCharm\PyCharm 2021.2.1\plugins\python\helpers\pycharm\_jb_unittest_runner.py" --path D:/pystdy/pattern/tests/tdd.py
Testing started at 3:50 ...


Ran 1 test in 0.001s

OK
Launching unittests with arguments python -m unittest D:/pystdy/pattern/tests/tdd.py in D:\pystdy
```