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)

Курс «Разработка интернет-приложений»

Отчет по лабораторной работе №4

Выполнил:
студент группы РТ5-51Б

Коваль И.А.

Преподаватель:

Гапанюк Ю.Е

2020

Описание задания:

1. Необходимо для произвольной предметной области реализовать три шаблона проектирования: один порождающий, один структурный и один поведенческий. В качестве справочника шаблонов можно использовать следующий каталог.
2. Для каждой реализации шаблона необходимо написать модульный тест. В модульных тестах необходимо применить следующие технологии:
   - TDD - фреймворк.
   - BDD - фреймворк.
   - Создание Mock-объектов.

Текст программы:

Main.py

```
from patterns.fabric pattern.MilkFabric import CheeseFabric, SourCreameFabric
from patterns.adapter_pattern.Smartphone import Iphone
from patterns.adapter pattern.LightningWire import LightningWire
from patterns.adapter pattern.AdapterUsb import AdapterUsb from
patterns.adapter_pattern.UsbWire import UsbWire
from patterns.method_pattern.GameAI import ElfBaseAI, OrcBaseAI
```

```
if   name   == '  main  ':
 cheeseFabric = CheeseFabric()
 print(cheeseFabric.deliver(2))
 sourcreameFabric = SourCreameFabric()
 print(sourcreameFabric.deliver(3))
```

```
 iphone = Iphone()
 lightningwire = LightningWire()
 usbwire = UsbWire()
 adapterusb = AdapterUsb(usbwire)
 print(iphone.charge(lightningwire))
 print(iphone.charge(adapterusb))
 print(iphone.charge(usbwire))
```

```
 ElfBase = ElfBaseAI(2000)
 Orcbase = OrcBaseAI(2000)
 ElfBase.turn(Orcbase)
 Orcbase.turn(ElfBase)
```

Adapter_pattern

```
from patterns.adapter pattern.LightningWire import LightningWire
from patterns.adapter pattern.UsbWire import UsbWire


class AdapterUsb(LightningWire):
 def   init   (self, usbwire: UsbWire):
 self.usbwire = usbwire
 def get port(self) -> str:
 if self.usbwire.get port() == "usb": # если разъемы переходника и кабеля
совпадают, то мы соединяем переходник и получаем другой разъем на выходе
 return "lightning"
 else:
 return "incompatible ports"


class LightningWire:

 def   init   (self):
 self.  port = "lightning"

 def get port(self) -> str:
 return self.  port

from patterns.adapter pattern.LightningWire import LightningWire
import time


class Iphone:
 def   init   (self):
 self.  port = "lightning"

 def charge(self, wire: LightningWire):
 if self.  port == wire.get port():
 print("Charging...")
 time.sleep(1)
 print("Your iphone is fully charged")
 return True
 else:
 print("Incompatible ports")
 return False


class UsbWire:

 def   init   (self):
 self.  port = "usb"

 def get port(self):
 return self.  port
```

Fabric_pattern

```
from   future   import annotations
from abc import ABC, abstractmethod
from patterns.fabric pattern.Product import MilkProduct, Cheese, SourCreame
```

```
class MilkFabric(ABC):

 @abstractmethod
 def create_milk_product(self) -> MilkProduct:
 pass

 def deliver(self, amount: int) -> list[MilkProduct]:
 products = []
 for i in range(amount):
 products.append(self.create_milk_product())
 print("Products with code name {} were successfully
delivered".format(products[0]))
 return products
class CheeseFabric(MilkFabric):

 def create_milk_product(self) -> MilkProduct:
return Cheese()


class SourCreameFabric(MilkFabric):

 def create_milk_product(self) -> MilkProduct:
return SourCreame()

from __future__ import annotations
from abc import ABC, abstractmethod


class MilkProduct(ABC):

 @abstractmethod
 def __repr__(self) -> str:
 pass


class Cheese(MilkProduct):

 def __repr__(self) -> str:
 return "Cheese"


class SourCreame(MilkProduct):

 def __repr__(self) -> str:
 return "SourCreame"
```

Method_pattern

```
from __future__ import annotations
from abc import ABC, abstractmethod
from patterns.method_pattern.Unit import Elf, Orc


class BaseAI(ABC):
 """Base class"""

 @abstractmethod
 def build_structures(self):
 pass
 @abstractmethod
 def gather_army(self):
 pass

 def attack(self, target: BaseAI):
"""default method"""
 return "Attacking {}".format(target)

 def turn(self, target: BaseAI):
 print(self.build structures())
print(self.gather army())
 print(self.attack(target))


class ElfBaseAI(BaseAI):
 def   init   (self, money):
 self.  money = money
 self.  unit = Elf()
 self.  building cost = 500
 self.built structures = 0
 self.army = []
 self.  unit cost = 200

 def build structures(self):
 amount = int((self.  money/2) / self.  building cost)
self.built structures = amount
 return "{} structures were built".format(self.built_structures)

 def gather army(self):
 amount = int((self.  money/2)/self.  unit cost)  for i
in range(amount):
 self.army.append(Elf())
 return "{} elves were recruited".format(len(self.army))

 def   repr   (self):
 return "ElfBase"


class OrcBaseAI(BaseAI):


 def   init   (self, money):
 self.  money = money
 self.  unit = Orc()
 self.  building cost = 300
 self.built structures = 0
 self.army = []
 self.  unit cost = 100

 def build structures(self):
 amount = int((self.  money / 3) / self.  building cost)
self.built structures = amount
 return "{} structures were built".format(self.built structures)

 def gather army(self):
 amount = int((self.  money * 2 / 3) / self.  unit cost)  for i
in range(amount):
 self.army.append(Elf())
 return "{} orcs were recruited".format(len(self.army))
 def   repr  (self):
 return "OrcBase"
from   future   import annotations
from abc import ABC, abstractmethod


class Unit(ABC):

 @abstractmethod
 def __repr__(self):
 pass


class Elf(Unit):
 def   init  (self):
 self.  unit = "elf"
 def   repr  (self):
 return self.  unit


class Orc(Unit):
 def   init  (self):
 self.  unit = "orc"

 def   repr  (self):
 return self.  unit
```

Tests

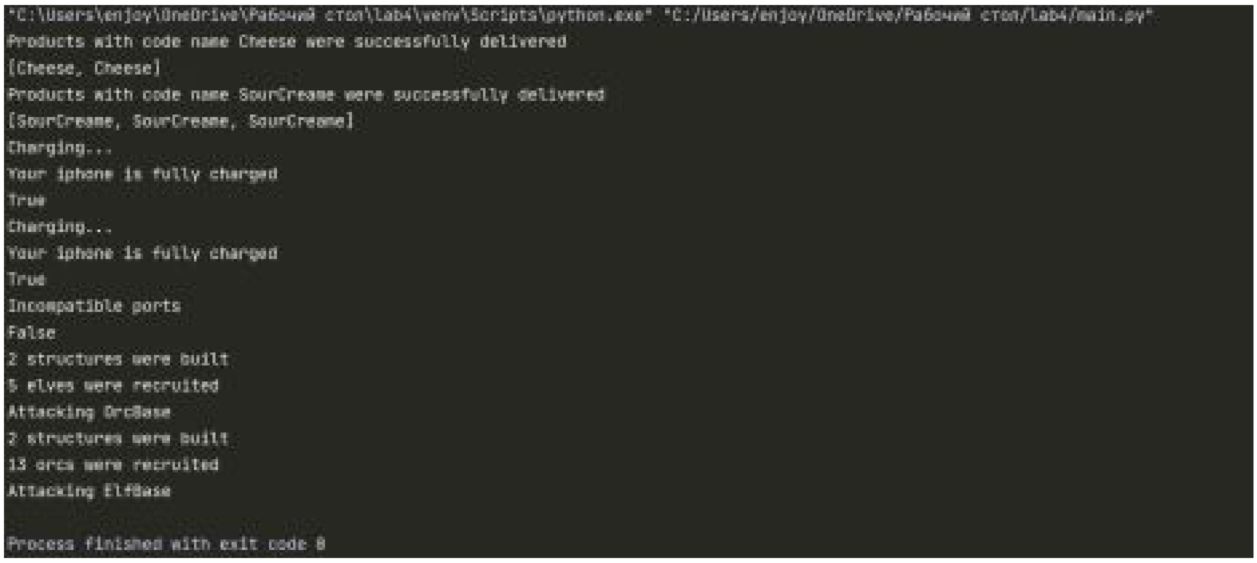
```
from patterns.adapter pattern.Smartphone import Iphone
from patterns.adapter pattern.LightningWire import LightningWire
from patterns.adapter pattern.AdapterUsb import AdapterUsb from
patterns.adapter_pattern.UsbWire import UsbWire


def test charging():
  iphone = Iphone()
 lightningwire = LightningWire()
 usbwire = UsbWire()
 adapterusb = AdapterUsb(usbwire)

 assert iphone.charge(lightningwire)
 assert not iphone.charge(usbwire)
 assert iphone.charge(adapterusb)


from patterns.fabric pattern.MilkFabric import CheeseFabric, SourCreameFabric
from patterns.fabric pattern.Product import Cheese, SourCreame


def get cheese list():
 cheese list = [Cheese(), Cheese(), Cheese()]
 return cheese list


def test fabric(monkeypatch):
 cheesefabric = CheeseFabric()
 monkeypatch.setattr(cheesefabric, "deliver", get cheese list)
```

```
assert len(cheesefabric.deliver()) == 3
 sourcreamefabric = SourCreameFabric()
 assert type(sourcreamefabric.deliver(2)) == list
 assert len(sourcreamefabric.deliver(2)) == 2
 assert type(sourcreamefabric.deliver(2)[0]) == SourCreame
```

```
from patterns.method pattern.GameAI import ElfBaseAI, OrcBaseAI
```

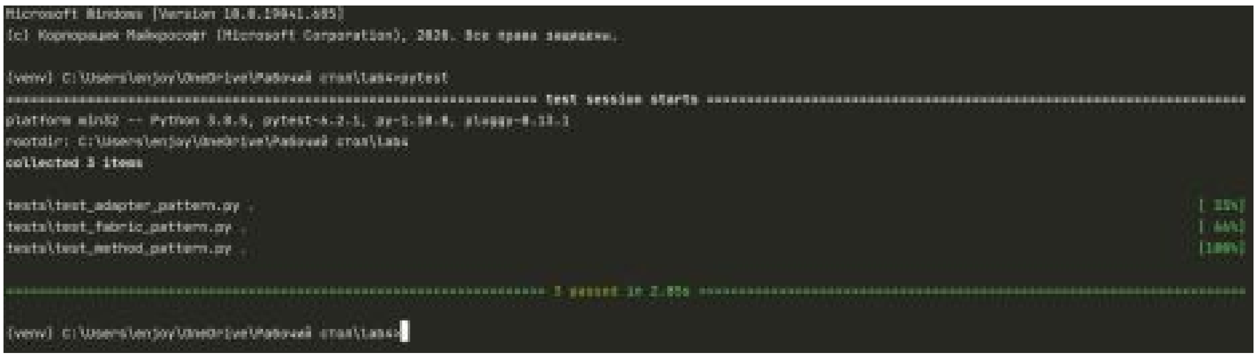
```
def test bases():
 elf base = ElfBaseAI(2000)
 assert elf base.gather army() == "5 elves were recruited"  assert
elf base.build structures() == "2 structures were built"  orc base =
OrcBaseAI(3000)
 assert orc base.gather army() == "20 orcs were recruited"  assert
orc base.build structures() == "3 structures were built"
```

Экранные формы с примерами выполнения программы



```
"C:\Users\enjoy\OneDrive\Рабочий стол\lab4\venv\Scripts\python.exe" "C:/Users/enjoy/OneDrive/Рабочий стол/lab4/main.py"
Products with code name Cheese were successfully delivered
[Cheese, Cheese]
Products with code name SourCreame were successfully delivered
[SourCreame, SourCreame, SourCreame]
Charging...
Your iphone is fully charged
True
Charging...
Your iphone is fully charged
True
Incompatible ports
False
2 structures were built
5 elves were recruited
Attacking OrcBase
2 structures were built
13 orcs were recruited
Attacking ElfBase

Process finished with exit code 0
```

Результат выполнения тестирования



```
Microsoft Windows [Version 10.0.19041.685]
(c) Корпорация Майкрософт (Microsoft Corporation), 2020. Все права защищены.

(venv) C:\Users\enjoy\OneDrive\Рабочий стол\lab4>pytest
============================================ test session starts ============================================
platform win32 -- Python 3.8.5, pytest-6.2.1, py-1.10.0, pluggy-0.13.1
rootdir: C:\Users\enjoy\OneDrive\Рабочий стол\lab4
collected 3 items

tests\test_adapter_pattern.py .                                                                        [ 33%]
tests\test_fabric_pattern.py .                                                                         [ 66%]
tests\test_method_pattern.py .                                                                         [100%]

============================================= 3 passed in 2.05s =============================================

(venv) C:\Users\enjoy\OneDrive\Рабочий стол\lab4>
```