**Министерство образования и науки Российской Федерации**

Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет

«Высшая школа экономики»

Кафедра Компьютерной безопасности

**ОТЧЕТ**

**К ЛАБОРАТОРНОЙ РАБОТЕ №5**

**по дисциплине**

**«Методы программирования»**

# Постановка задачи

**Вариант 7.** Используя паттерн Abstract Factory, реализовать представление графического интерфейса приложения, состоящего из нескольких частей (например, текст, баннеры, руководство пользователя) для нескольких языков (например, русского, английского и т.д.).

# Код программы

Main.cpp – Base\_Interface класс “абстрактной фабрики”, в которой определены абстрактные товары. В Interface1,2,3 Определены уже конкретные “товары” для определенных фабрик: русского, английского и немецкого языка (фабрик).

## main.cpp

from \_\_future\_\_ import annotations  
from abc import ABC, abstractmethod  
  
  
class Base\_Interface(ABC):  
 @abstractmethod  
 def get\_text(self):  
 pass  
  
 @abstractmethod  
 def get\_banner(self):  
 pass  
  
 @abstractmethod  
 def get\_manual(self):  
 pass  
  
  
class Interface1(Base\_Interface):  
 def get\_text(self):  
 return Rus\_Text()  
  
 def get\_banner(self):  
 return Rus\_Banner()  
  
 def get\_manual(self):  
 return Rus\_Manual()  
  
  
class Interface2(Base\_Interface):  
 def get\_text(self):  
 return Eng\_Text()  
  
 def get\_banner(self):  
 return Eng\_Banner()  
  
 def get\_manual(self):  
 return Eng\_Manual()  
  
  
class Interface3(Base\_Interface):  
 def get\_text(self):  
 return Du\_Text()  
  
 def get\_banner(self):  
 return Du\_Banner()  
  
 def get\_manual(self):  
 return Du\_Manual()  
  
  
class Text(ABC):  
 @abstractmethod  
 def Make\_Text(self) -> str:  
 pass  
  
class Rus\_Text(Text):  
 def Make\_Text(self) -> str:  
 return **"Этот текст показан на русском"**class Eng\_Text(Text):  
 def Make\_Text(self) -> str:  
 return **"This text is English"**class Du\_Text(Text):  
 def Make\_Text(self) -> str:  
 return **"Text auf Deutsch"**class Banner(ABC):  
 @abstractmethod  
 def Make\_Banner(self) -> str:  
 pass  
  
  
class Rus\_Banner(Banner):  
 def Make\_Banner(self) -> str:  
 return **"||БАННЕР НА РУССКОМ||"**class Eng\_Banner(Banner):  
 def Make\_Banner(self) -> str:  
 return **"||ENGLISH BANNER||"**class Du\_Banner(Banner):  
 def Make\_Banner(self) -> str:  
 return **"||BANNER AUF DEUTSCH||"**class Manual(ABC):  
 @abstractmethod  
 def Make\_Manual(self) -> str:  
 pass  
  
class Rus\_Manual(Manual):  
 def Make\_Manual(self) -> str:  
 return **"Мануал на русском: программа выполнена корректно!"**class Eng\_Manual(Manual):  
 def Make\_Manual(self) -> str:  
 return **"English manual: program is done!"**class Du\_Manual(Manual):  
 def Make\_Manual(self) -> str:  
 return **"Handbuch auf Deutsch: Das Programm wurde korrekt ausgeführt!"**def client\_code(lang) -> None:  
 if lang == **"Rus"**:  
 Interface = Interface1()  
 if lang == **"Eng"**:  
 Interface = Interface2()  
 if lang == **"Du"**:  
 Interface = Interface3()  
  
 print(Interface.get\_text().Make\_Text())  
 print(Interface.get\_banner().Make\_Banner())  
 print(Interface.get\_manual().Make\_Manual())  
  
if \_\_name\_\_ == **"\_\_main\_\_"**:  
 print(**"Choose language: Rus/Eng/Du (Russian/English/German)"**)  
 lang = str(input())  
 client\_code(lang)