МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчёт по лабораторной работе №3

Специальность ПО11

Выполнил
Зайченко С.В.
студент группы ПО11

Проверил
Крощенко А. А.
ст. преп. кафедры ИИТ

Брест 2025

Цель работы: Закрепить навыки объектно-ориентированного программирования на языке Python.

Ход Работы

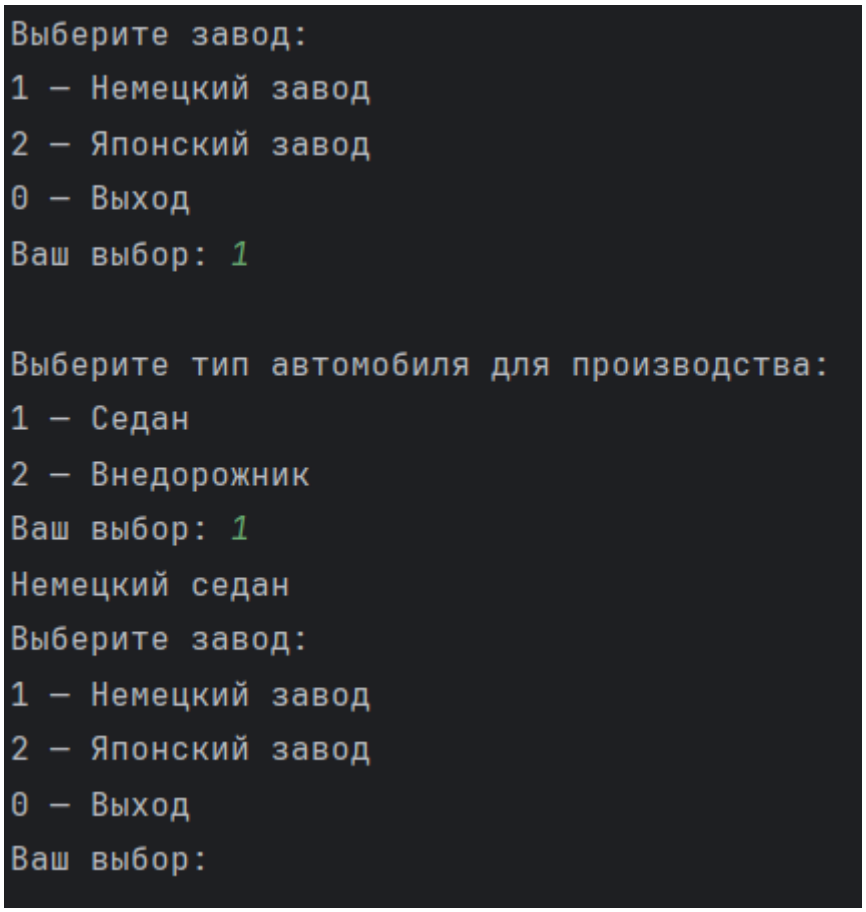
Требования к выполнению
Заводы по производству автомобилей. Реализовать возможность создавать автомобили различных типов на различных заводах.
Код программы:

```
from abc import ABC, abstractmethod

class Sedan(ABC):
  @abstractmethod
  def drive(self):
    pass

class SUV(ABC):
  @abstractmethod
  def drive(self):
    pass


class GermanSedan(Sedan):
  def drive(self):
    print("Немецкий седан")

class GermanSUV(SUV):
  def drive(self):
    print("Немецкий внедорожник")


class JapaneseSedan(Sedan):
  def drive(self):
    print("Японский седан")

class JapaneseSUV(SUV):
  def drive(self):
    print("Японский внедорожник")


class CarFactory(ABC):
  @abstractmethod
  def create_sedan(self) -> Sedan:
    pass
```

```
    @abstractmethod
    def create_suv(self) -> SUV:
        pass


class GermanCarFactory(CarFactory):
    def create_sedan(self) -> Sedan:
        return GermanSedan()

    def create_suv(self) -> SUV:
        return GermanSUV()


class JapaneseCarFactory(CarFactory):
    def create_sedan(self) -> Sedan:
        return JapaneseSedan()

    def create_suv(self) -> SUV:
        return JapaneseSUV()


def user_interface():
    while True:
        print("Выберите завод:")
        print("1 — Немецкий завод")
        print("2 — Японский завод")
        print("0 — Выход")

        factory_choice = input("Ваш выбор: ").strip()
        if factory_choice == "0":
            print("До свидания!")
            break

        if factory_choice == "1":
            factory = GermanCarFactory()
        elif factory_choice == "2":
            factory = JapaneseCarFactory()
        else:
            print("Неверный выбор завода. Попробуйте снова.\n")
            continue

        print("\nВыберите тип автомобиля для производства:")
```

```
    print("1 — Седан")
    print("2 — Внедорожник")
    car_choice = input("Ваш выбор: ").strip()

    if car_choice == "1":
        car = factory.create_sedan()
        car.drive()
    elif car_choice == "2":
        car = factory.create_suv()
        car.drive()
    else:
        print("Неверный тип автомобиля. Попробуйте снова.\n")



if __name__ == "__main__":
    user_interface()
```

Рисунки с результатами работы программы:



Задание 2

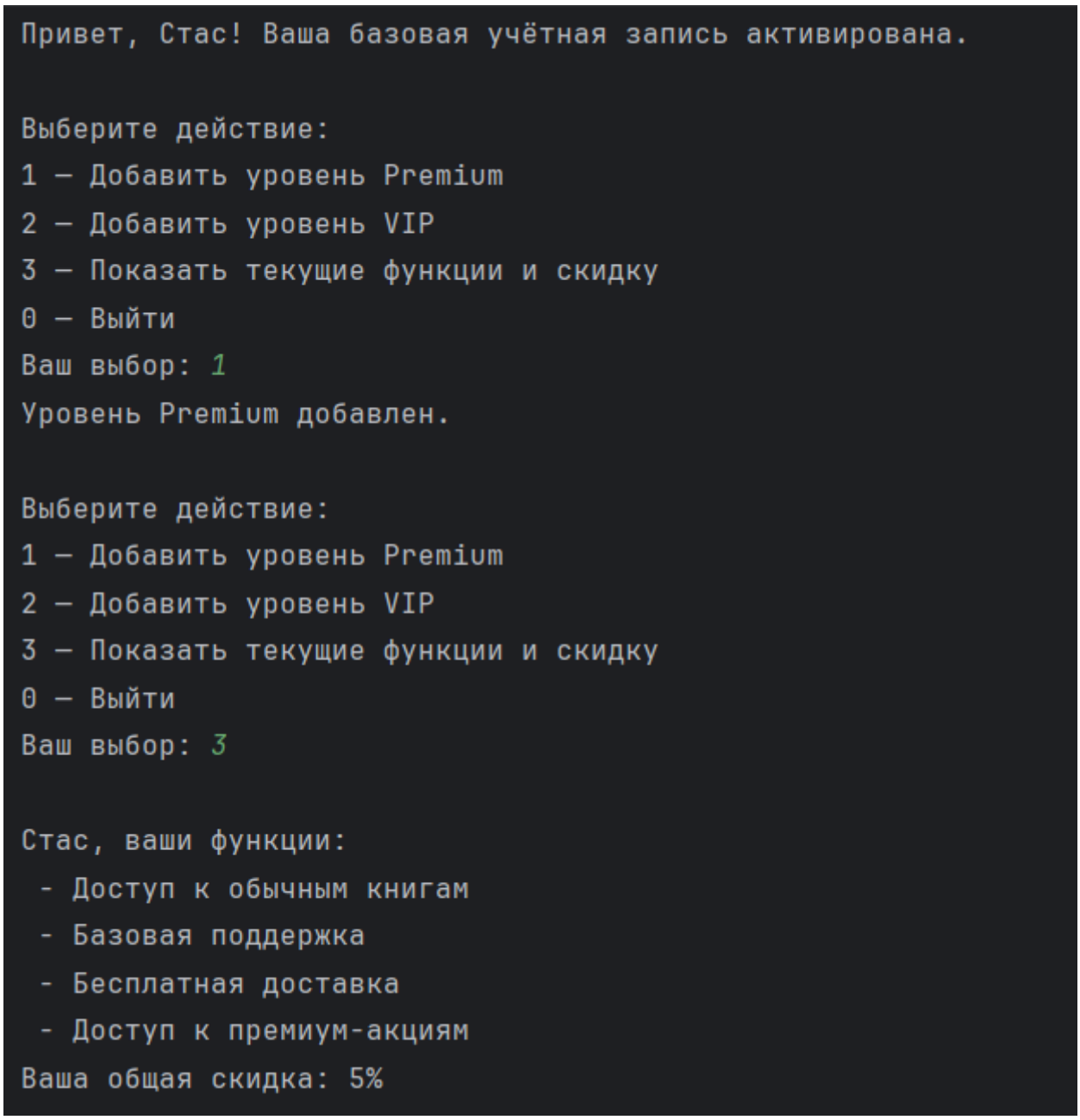
Учетная запись покупателя книжного интернет-магазина. Предусмотреть различные уровни учетки в зависимости от активности покупателя. Дополнительные уровни добавляют функциональные возможности и

открывают доступ к уникальным предложениям.
Код программы:

```
from abc import ABC, abstractmethod


class Account(ABC):
  @abstractmethod
  def get_features(self):
    pass

  @abstractmethod
  def get_discount(self):
    pass


class BasicAccount(Account):
  def get_features(self):
    return ["Доступ к обычным книгам", "Базовая поддержка"]

  def get_discount(self):
    return 0


class AccountDecorator(Account):
  def __init__(self, wrapped_account: Account):
    self._wrapped_account = wrapped_account


class PremiumAccount(AccountDecorator):
  def get_features(self):
    return self._wrapped_account.get_features() + [
      "Бесплатная доставка", "Доступ к премиум-акциям"
    ]

  def get_discount(self):
    return self._wrapped_account.get_discount() + 5


class VIPAccount(AccountDecorator):
  def get_features(self):
    return self._wrapped_account.get_features() + [
      "Личный менеджер", "Эксклюзивные издания"
    ]
```

```
    def get_discount(self):
        return self._wrapped_account.get_discount() + 10


def user_interface():
    name = input("Введите ваше имя: ")
    account: Account = BasicAccount()

    print(f"\nПривет, {name}! Ваша базовая учётная запись активирована.")

    while True:
        print("\nВыберите действие:")
        print("1 — Добавить уровень Premium")
        print("2 — Добавить уровень VIP")
        print("3 — Показать текущие функции и скидку")
        print("0 — Выйти")

        choice = input("Ваш выбор: ").strip()

        if choice == "1":
            account = PremiumAccount(account)
            print("Уровень Premium добавлен.")
        elif choice == "2":
            account = VIPAccount(account)
            print("Уровень VIP добавлен.")
        elif choice == "3":
            print(f"\n{name}, ваши функции:")
            for feature in account.get_features():
                print(" -", feature)
            print(f"Ваша общая скидка: {account.get_discount()}%")
        elif choice == "0":
            print("До свидания!")
            break
        else:
            print("Неверный ввод. Попробуйте снова.")


if __name__ == "__main__":
    user_interface()
```

Рисунки с результатами работы программы:



Задание 3

Проект «Банкомат». Предусмотреть выполнение основных операций (ввод пин-кода, снятие суммы, завершение работы) и наличие различных режимов работы (ожидание, аутентификация, выполнение операции, блокировка – если нет денег). Атрибуты: общая сумма денег в банкомате, ID.

Код программы:

```
from abc import ABC, abstractmethod


class ATM:
  def __init__(self, atm_id, total_money):
    self.atm_id = atm_id
```

```
        self.total_money = total_money
        self.correct_pin = "1234"
        self.state: ATMState = WaitingState(self)

    def set_state(self, state):
        self.state = state

    def insert_card(self):
        self.state.insert_card()

    def enter_pin(self, pin):
        self.state.enter_pin(pin)

    def withdraw(self, amount):
        self.state.withdraw(amount)

    def finish(self):
        self.state.finish()


class ATMState(ABC):
    def __init__(self, atm: ATM):
        self.atm = atm

    @abstractmethod
    def insert_card(self): pass

    @abstractmethod
    def enter_pin(self, pin): pass

    @abstractmethod
    def withdraw(self, amount): pass

    @abstractmethod
    def finish(self): pass


class WaitingState(ATMState):
    def insert_card(self):
        print("Карта вставлена. Введите PIN.")
        self.atm.set_state(AuthenticationState(self.atm))

    def enter_pin(self, pin):
```

```
        print("Вставьте карту сначала.")

    def withdraw(self, amount):
        print("Невозможно снять деньги. Нет карты.")

    def finish(self):
        print("Нет активной сессии.")

class AuthenticationState(ATMState):
    def insert_card(self):
        print("Карта уже вставлена.")

    def enter_pin(self, pin):
        if pin == self.atm.correct_pin:
            print("ПИН-код верный.")
            self.atm.set_state(OperationState(self.atm))
        else:
            print("Неверный ПИН-код.")
            self.atm.set_state(WaitingState(self.atm))

    def withdraw(self, amount):
        print("Сначала введите ПИН.")

    def finish(self):
        print("Сначала введите ПИН.")

class OperationState(ATMState):
    def insert_card(self):
        print("Операция уже выполняется.")

    def enter_pin(self, pin):
        print("ПИН уже введён.")

    def withdraw(self, amount):
        if amount <= self.atm.total_money:
            self.atm.total_money -= amount
            print(f"Выдано: {amount}. Остаток: {self.atm.total_money}")
            if self.atm.total_money == 0:
                print("Банкомат пуст. Блокировка.")
                self.atm.set_state(BlockedState(self.atm))
        else:
            print("Недостаточно средств в банкомате.")
```

```
    def finish(self):
        print("Сессия завершена.")
        self.atm.set_state(WaitingState(self.atm))

class BlockedState(ATMState):
    def insert_card(self):
        print("Банкомат заблокирован. Недостаточно средств.")

    def enter_pin(self, pin):
        print("Банкомат заблокирован.")

    def withdraw(self, amount):
        print("Невозможно выполнить операцию. Банкомат пуст.")

    def finish(self):
        print("Банкомат заблокирован.")


if __name__ == "__main__":
    atm = ATM(atm_id="1", total_money=500)

    while True:
        print("1. Вставить карту")
        print("2. Ввести ПИН-код")
        print("3. Снять деньги")
        print("4. Завершить сессию")
        print("0. Выйти")

        choice = input("Выберите действие: ").strip()

        if choice == "1":
            atm.insert_card()
        elif choice == "2":
            pin = input("Введите ПИН-код: ")
            atm.enter_pin(pin)
        elif choice == "3":
            amount = int(input("Введите сумму для снятия: "))
            atm.withdraw(amount)
        elif choice == "4":
            atm.finish()
        elif choice == "0":
            print("До свидания!")
            break
```

```
    else:
        print("Неверный ввод. Попробуйте снова.")
```

Рисунки с результатами работы программы:

```
Карта вставлена. Введите PIN.
1. Вставить карту
2. Ввести ПИН-код
3. Снять деньги
4. Завершить сессию
0. Выйти
Выберите действие: 2
Введите ПИН-код: 1234
ПИН-код верный.
1. Вставить карту
2. Ввести ПИН-код
3. Снять деньги
4. Завершить сессию
0. Выйти
Выберите действие: 3
Введите сумму для снятия: 32
Выдано: 32. Остаток: 468
```

Вывод: Закрепил навыки объектно-ориентированного программирования на языке Python