МИНОБРНАУКИ РОССИИ
федеральное государственное автономное образовательное учреждение высшего образования
«Московский государственный технологический университет «СТАНКИН»
(ФГБОУ ВО «МГТУ «СТАНКИН»)

Институт информационных технологий

Кафедра управления и информатики в технических системах

ЛАБОРАТОРНАЯ РАБОТА №1

ПО ДИСЦИПЛИНЕ

«ПРИКЛАДНОЕ ПРОГРАММИРОВАНИЕ»

СТУДЕНТА 2 КУРСА БАКАЛАВРИАТА ГРУППЫ ИДБ-24-11

Башарова Тимура Руслановича

(ФИО)

ТЕМА РАБОТЫ

«ООП в Python. Обработка исключительных ситуаций. Форматы XML и JSON»

Направление: 09.03.03 Прикладная информатика

Отчет сдан «_____» _____2025 г.

Оценка _____

Преподаватель Кайкова Юлия Викторовна _____

(подпись)

МОСКВА 2025

Содержание

Описание работы

Цель работы:

- Разработать программу для предметной области интернет-магазина электроники

Вариант: 14

Технологии:

1. язык программирования Python,
2. диаграмма классов UML,
3. форматы данных XML и JSON.

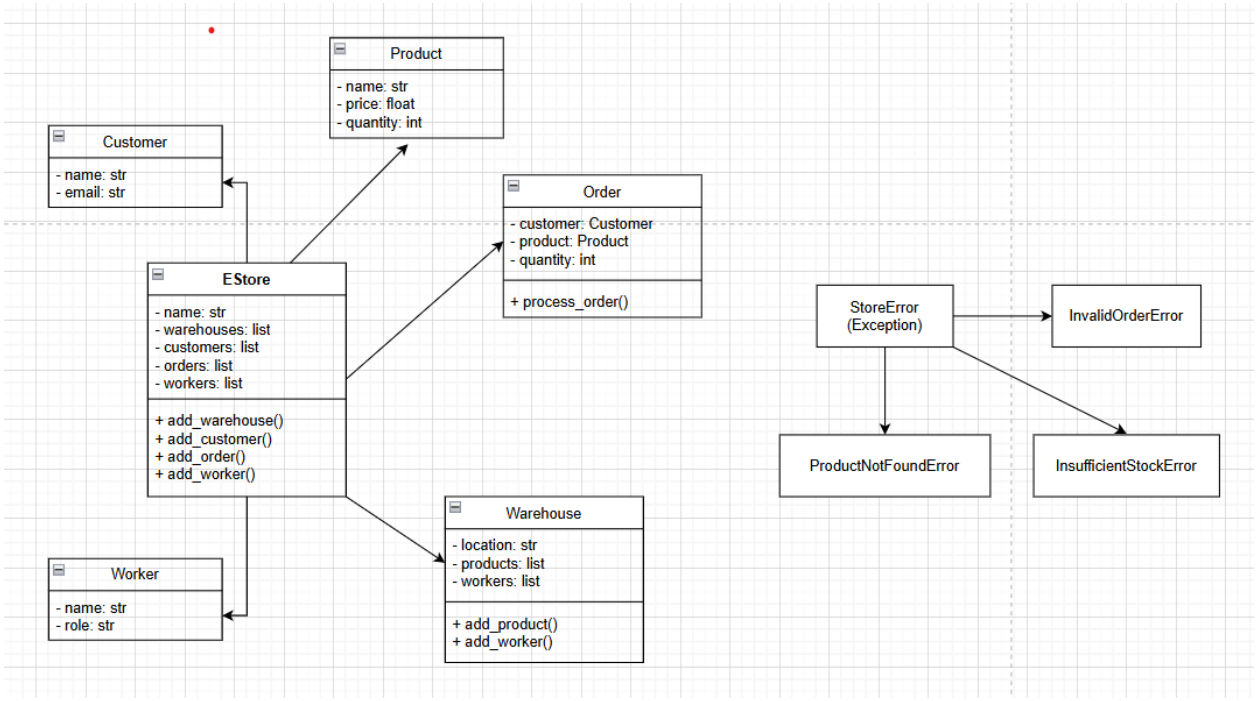
Инструменты:

1. интегрированная среда разработки для языка программирования Python (PyCharm или аналог)
2. сервис draw.io (https://app.diagrams.net/) или аналог,
3. текстовый редактор Notepad++, Sublime Text или аналог.

Задача:

1) спроектировать в сервисе draw.io диаграмму классов для выбранной предметной области (предложив свой вариант или выбрав один из представленных) и написать по ней код на Python в объектно-ориентированном стиле;
2) обработать встроенные и собственное исключения (чтобы при работе пользователя не допускать выхода из программы при возникновении ошибок), после чего актуализировать диаграмму классов;
3) спроектировать в текстовом редакторе структуру для хранения данных предметной области в форматах XML и JSON;
4) написать код на Python, реализующий считывание из файла и запись в файл по разработанной структуре в форматах XML и JSON;
5) подготовить отчет о выполненной лабораторной работе и подготовиться по вопросам к защите.

Программная реализация



```python
import json
import xml.etree.ElementTree as ET


# главный класс магазина
class EStore:
    """Главный класс, представляющий интернет-магазин электроники."""

    def __init__(self, name: str):
        self.name = name
        self.warehouses: list[Warehouse] = []
        self.customers: list[Customer] = []
        self.orders: list[Order] = []
        self.workers: list[Worker] = []

    def add_warehouse(self, warehouse: "Warehouse") -> None:
        self.warehouses.append(warehouse)

    def add_customer(self, customer: "Customer") -> None:
        self.customers.append(customer)

    def add_order(self, order: "Order") -> None:
        self.orders.append(order)

    def add_worker(self, worker: "Worker") -> None:
        self.workers.append(worker)

    #работа с JSON файлами
    def save_to_json(self, filename: str) -> None:
        """Сохранение информации о магазине в JSON."""
        data = {
            "name": self.name,
```

```python
#работа с JSON файлами
def save_to_json(self, filename: str) -> None:
    """Сохранение информации о магазине в JSON."""
    data = {
        "name": self.name,
        "warehouses": [
            {
                "location": w.location,
                "products": [{"name": p.name, "price": p.price, "quantity": p.quantity} for p in w.products],
                "workers": [{"name": wr.name, "role": wr.role} for wr in w.workers]
            } for w in self.warehouses
        ],
        "customers": [{"name": c.name, "email": c.email} for c in self.customers],
        "orders": [
            {"customer": o.customer.name, "product": o.product.name, "quantity": o.quantity}
            for o in self.orders
        ],
    }
    with open(filename, "w", encoding="utf-8") as f:
        json.dump(data, f, ensure_ascii=False, indent=4)
```

```python
    # работа с XML файлами
    def save_to_xml(self, filename: str) -> None:
        """Сохранение информации о магазине в XML."""
        root = ET.Element( tag: "estore", name=self.name)

        for w in self.warehouses:
            w_el = ET.SubElement(root,  tag: "warehouse", location=w.location)

            for p in w.products:
                ET.SubElement(
                    w_el,  tag: "product",
                    name=p.name,
                    price=str(p.price),
                    quantity=str(p.quantity)
                )

        tree = ET.ElementTree(root)
        tree.write(filename, encoding="utf-8", xml_declaration=True)


class Product:
    """Класс, представляющий товар магазина."""

    def __init__(self, name: str, price: float, quantity: int = 0):
        self.name = name
        self.price = price
        self.quantity = quantity
```

```python
class Worker:

    def __init__(self, name: str, role: str):
        self.name = name
        self.role = role


class Customer:
    """Класс, представляющий покупателя."""

    def __init__(self, name: str, email: str):
        self.name = name
        self.email = email


class Warehouse:
    """Класс, представляющий склад магазина."""

    def __init__(self, location: str):
        self.location = location
        self.products: list[Product] = []
        self.workers: list[Worker] = []

    def add_product(self, product: Product) -> None:
        """Добавить товар на склад."""
        self.products.append(product)

    def get_products(self) -> list[Product]:
        """Получить список товаров."""
        return self.products
```

```python
def get_products(self) -> list[Product]:
    """Получить список товаров."""
    return self.products

def update_product(self, name: str, new_price: float) -> None:
    """Изменить цену товара."""
    for p in self.products:
        if p.name == name:
            p.price = new_price
            return
    raise ProductNotFoundError(f"Товар '{name}' не найден")

def remove_product(self, name: str) -> None:
    """Удалить товар по названию."""
    for p in self.products:
        if p.name == name:
            self.products.remove(p)
            return
    raise ProductNotFoundError(f"Товар '{name}' не найден")

def add_worker(self, worker: Worker) -> None:
    """Добавить сотрудника на склад."""
    self.workers.append(worker)
```

```python
#классы исключения
class StoreError(Exception):
    """Базовый класс ошибок"""
    pass


class ProductNotFoundError(StoreError):
    """Ошибка: товар не найден"""
    pass


class InsufficientStockError(StoreError):
    """Ошибка: недостаточно товара на складе"""
    pass


class InvalidOrderError(StoreError):
    """Ошибка: некорректный заказ"""
    pass
```

Вывод

В ходе выполнения лабораторной работы была разработана программа для моделирования работы интернет-магазина с использованием объектно-ориентированного подхода. В программе реализованы основные классы предметной области, обработка исключений, а также функции для сохранения данных в формате JSON и XML.

Были закреплены навыки работы с классами, аннотациями типов, сериализацией данных и принципами структурирования кода по стандарту PEP 8.

Результатом работы стала корректно функционирующая программа, демонстрирующая создание объектов, выполнение операций и обработку ошибок.

https://github.com/TimurBasharov/PPlabNo1