Министерство Образования Республики Беларусь
УО Брестский Государственный Технический Университет
Факультет электронно-информационных систем
Кафедра ИИТ

Отчет по лабораторной работе № 2

Специальность ПО13

Выполнил:
А.С.Потапчук,
Студент группы ПО13
Проверил:
А.А.Крощенко
Доцент кафедры ИИТ
«____» _____ 2026 г.

Брест 2026

Цель работы: закрепить навыки объектно-ориентированного программирования на языке Python.
Вариант 8

Задание 1

Реализовать простой класс

Требования к выполнению

• Реализовать пользовательский класс по варианту.

Для каждого класса

• Создать атрибуты (поля) классов

• Создать методы классов

• Добавить необходимые свойства и сеттеры (по необходимости)

• Переопределить магические методы __str__ и __eq__

8)Множество целых чисел переменной мощности – Предусмотреть возможность пересечения двух множеств, вывода на консоль элементов множества, а также метод, определяющий, принадлежит ли указанное значение множеству. Класс должен содержать методы, позволяющие добавлять и удалять элемент в/из множества. Конструктор должен позволять создавать объекты с начальной инициализацией. Реализацию множества осуществить на базе списка. Переопределить метод __eq__, выполняющий сравнение объектов данного типа.

Выполнение:

Код программы:

```
class IntegerSet:
    def __init__(self, initial_elements=None):
        if initial_elements is None:
            self.elements = []
        else:
            # Убираем дубликаты, сохраняем порядок первого появления
            seen = set()
            self.elements = [
                x for x in initial_elements if not (x in seen or seen.add(x))
            ]

    def add(self, element):
        if element not in self.elements:
            self.elements.append(element)

    def remove(self, element):
        if element in self.elements:
            self.elements.remove(element)

    def contains(self, element):
        return element in self.elements

    def intersection(self, other):
        if not isinstance(other, IntegerSet):
            raise ValueError("Can only intersect with another IntegerSet")
        common = [x for x in self.elements if x in other.elements]
        return IntegerSet(common)

    def __str__(self):
        if not self.elements:
            return "{}"
        return "{" + ", ".join(map(str, sorted(self.elements))) + "}"

    def __eq__(self, other):
        if not isinstance(other, IntegerSet):
            return False
        return sorted(self.elements) == sorted(other.elements)


def main():
    try:
```

```
        # Ввод первого множества
        n1 = int(input("Enter number of elements in first set: "))
        line1 = input("Enter elements of first set: ").strip()
        elems1 = list(map(int, line1.split()))[:n1]
        set1 = IntegerSet(elems1)

        # Ввод второго множества
        n2 = int(input("Enter number of elements in second set: "))
        line2 = input("Enter elements of second set: ").strip()
        elems2 = list(map(int, line2.split()))[:n2]
        set2 = IntegerSet(elems2)

        # Вывод результатов
        print("First set:      ", set1)
        print("Second set:     ", set2)
        print("Intersection:   ", set1.intersection(set2))

        value = int(input("Enter value to check in first set: "))
        print(f"{value} in first set?   {set1.contains(value)}")

        add_val = int(input("Enter value to add to first set: "))
        set1.add(add_val)
        print("After adding:   ", set1)

        rem_val = int(input("Enter value to remove from first set: "))
        set1.remove(rem_val)
        print("After removing: ", set1)

        print("Sets are equal: ", set1 == set2)

    except ValueError as e:
        print("Error: invalid input (expected integer(s))")
    except Exception as e:
        print(f"Unexpected error: {e}")


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

Результат работы программы:

```
Enter number of elements in first set: 4
Enter elements of first set: 1 3 5 6
Enter number of elements in second set: 3
Enter elements of second set: 3 5 1
First set:       {1, 3, 5, 6}
Second set:      {1, 3, 5}
Intersection:    {1, 3, 5}
Enter value to check in first set: 3
3 in first set?   True
Enter value to add to first set: 1
After adding:    {1, 3, 5, 6}
Enter value to remove from first set: 5
After removing:  {1, 3, 6}
Sets are equal:  False
Для продолжения нажмите любую клавишу . . .
```

Задание 2

8) Система Интернет-магазин. Администратор добавляет информацию о Товаре. Клиент делает и оплачивает Заказ на Товары. Администратор регистрирует Продажу и может занести неплательщиков в «черный список».

Выполнение:

Код программы :

```
class Товар:
    def __init__(self, название, цена):
        self.название = название
        self.цена = цена

    def __str__(self):
        return f"{self.название} ({self.цена} руб.)"


class Пользователь:
```

```
    def __init__(self, имя):
        self.имя = имя


class Клиент(Пользователь):
    def __init__(self, имя, баланс=0):
        super().__init__(имя)
        self.баланс = баланс
        self.в_черном_списке = False

    def оплатить(self, сумма):
        if self.баланс >= сумма:
            self.баланс -= сумма
            return True
        return False


class Администратор(Пользователь):
    def добавить_товар(self, каталог, товар):
        каталог.append(товар)
        print(f"→ Админ {self.имя}: добавлен товар '{товар.название}'")

    def заблокировать(self, клиент):
        клиент.в_черном_списке = True
        print(f"!!! Админ {self.имя}: клиент {клиент.имя} занесён в ЧЁРНЫЙ СПИСОК!")


class Заказ:
    def __init__(self, клиент):
        self.клиент = клиент
        self.товары = []

    def добавить_товар(self, товар):
        self.товары.append(товар)

    def общая_стоимость(self):
        return sum(товар.цена for товар in self.товары)


class Магазин:
    def __init__(self, админ):
        self.админ = админ
        self.каталог = []
        self.продажи = []

    def зарегистрировать_продажу(self, заказ):
        if заказ.клиент.в_черном_списке:
            print(f"Отказ: клиент {заказ.клиент.имя} в чёрном списке.")
            return

        стоимость = заказ.общая_стоимость()
        print(f"Попытка оплаты заказа на {стоимость} руб.")

        if заказ.клиент.оплатить(стоимость):
            self.продажи.append(заказ)
            print(f"Успех: продажа оформлена для {заказ.клиент.имя}.")
        else:
            print(f"Ошибка: недостаточно средств у {заказ.клиент.имя}.")
            self.админ.заблокировать(заказ.клиент)


def main():
    try:
        имя_админа = input("Имя администратора магазина: ").strip()
        админ = Администратор(имя_админа)
        магазин = Магазин(админ)

        print("\n--- Добавление товаров (напишите 'стоп' для завершения) ---")
        while True:
            название = input("Название товара: ").strip()
            if название.lower() == "стоп":
                break
            try:
                цена = int(input(f"Цена '{название}': "))
                if цена < 0:
                    print("Цена не может быть отрицательной")
                    continue
                админ.добавить_товар(магазин.каталог, Товар(название, цена))
            except ValueError:
                print("Ошибка: введите целое число для цены")

        if not магазин.каталог:
```

```
            print("Каталог пуст → завершение программы")
            return

        print("\n--- Оформление заказа ---")
        имя_клиента = input("Имя клиента: ").strip()
        try:
            деньги = int(input(f"Сколько денег у {имя_клиента}? "))
            if деньги < 0:
                print("Баланс не может быть отрицательным")
                return
        except ValueError:
            print("Ошибка: введите целое число")
            return

        клиент = Клиент(имя_клиента, деньги)
        заказ = Заказ(клиент)

        print("\nДоступные товары:")
        for i, товар in enumerate(магазин.каталог):
            print(f"  {i:2d}. {товар}")

        ввод = input("\nНомера товаров через пробел (например: 0 2 1): ").strip()
        if not ввод:
            print("Заказ пуст")
            return

        for часть in ввод.split():
            try:
                индекс = int(часть)
                if 0 <= индекс < len(магазин.каталог):
                    заказ.добавить_товар(магазин.каталог[индекс])
                else:
                    print(f"Неверный номер: {часть} (игнорируется)")
            except ValueError:
                print(f"Некорректный ввод: {часть} (игнорируется)")

        if not заказ.товары:
            print("В заказ ничего не добавлено")
            return

        print("\n--- Результат ---")
        магазин.зарегистрировать_продажу(заказ)

        print(f"\nОстаток у клиента {клиент.имя}: {клиент.баланс} руб.")

    except Exception as e:
        print(f"Произошла ошибка: {e}")


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

Результат работы программы:

```
Имя администратора магазина: anita

--- Добавление товаров (напишите 'стоп' для завершения) ---
Название товара: мука
Цена 'мука': 12
→ Админ anita: добавлен товар 'мука'
Название товара: соль
Цена 'соль': 1
→ Админ anita: добавлен товар 'соль'
Название товара:
Цена '':
Ошибка: введите целое число для цены
Название товара: перец
Цена 'перец': 23
→ Админ anita: добавлен товар 'перец'
Название товара: стоп

--- Оформление заказа ---
Имя клиента: оля
Сколько денег у оля? 24

Доступные товары:
   0. мука (12 руб.)
   1. соль (1 руб.)
   2. перец (23 руб.)

Номера товаров через пробел (например: 0 2 1): 0

--- Результат ---
Попытка оплаты заказа на 12 руб.
Успех: продажа оформлена для оля.

Остаток у клиента оля: 12 руб.
Press any key to continue . . .
```

Вывод: закрепила навыки объектно-ориентированного программирования на языке Python.