МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №2

Специальность ПО-13

Выполнил:

Д. А. Марковский

студент группы ПО-13

Проверил:

А. А. Крощенко

Доцент кафедры ИИТ

20.02.2026

Брест 2026

Цель работы:

Закрепить навыки объектно-ориентированного программирования на языке Python.

Задание 1.

• 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__

5) Множество целых чисел ограниченной мощности – Предусмотреть возможность объединения 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. Мощность множества задается при создании объекта. Реализацию множества осуществить на базе списка. Переопределить метод __eq__, выполняющий сравнение объектов данного типа.

Код программы:

```
class IntegerSet:
    def __init__(self, max_size, elements=None):
        if elements is None:
            elements = []
        self.max_size = max_size
        self.elements = list(set(elements))[:max_size]

    def __str__(self):
        return f"{self.elements}"

    def __repr__(self):
        return f"IntegerSet(max_size={self.max_size}, elements={self.elements})"

    def __eq__(self, other):
        return self.max_size == other.max_size and self.elements ==
other.elements

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

    def union(self, other):
        new_max = max(self.max_size, other.max_size)
        new_elements = list(set(self.elements + other.elements))[:new_max]
        return IntegerSet(new_max, new_elements)

    def add(self, item):
        try:
            num = int(item)
            if num not in self.elements and len(self.elements) < self.max_size:
                self.elements.append(num)
            elif num in self.elements:
                print(f"Элемент {num} уже существует в множестве")
            else:
                print(f"Нельзя добавить {num}: достигнута максимальная мощность
{self.max_size}")
        except ValueError:
            print(f"Ошибка: {item} не является целым числом")
```

```
    def remove(self, value):
        try:
            self.elements.remove(value)
        except ValueError:
            print(f"Элемент {value} не найден в множестве")


intSet1 = IntegerSet(max_size=5, elements=[6, 6, 2])
intSet2 = IntegerSet(max_size=4)
print(repr(intSet1))
print(repr(intSet2))
intSet2.add(1)
intSet2.add(2)
intSet2.add(3)
intSet2.add(4)
intSet2.add(5)
print("Множество 1:", repr(intSet1))
print("Множество 2:", repr(intSet2))
intSet3 = intSet1.union(intSet2)
print("Объединённое множество 3:", repr(intSet3))
print("Содержит ли множество 3 элемент 8:", intSet3.contains(8))
```

Спецификация ввода

Ввод не требуется – основная программа приводит тестовое исполнение функциональности.

Спецификация вывода

В консоль выводятся два инициализированных множества, сообщение о невозможности добавить элемент по достижению максимальной мощности, множества 1 и 2 и объединённое множество 3, проверка на наличие во множестве 3 элемента 8.

Рисунки с результатами работы программы

```
E:\kurs3\SPP\LAB1\.venv\Scripts\python.exe E:\kurs3\SPP\LAB1\spp_po13\reports\Markovsky\2\src\1.py
IntegerSet(max_size=5, elements=[2, 6])
IntegerSet(max_size=4, elements=[])
Нельзя добавить 5: достигнута максимальная мощность 4
Множество 1: IntegerSet(max_size=5, elements=[2, 6])
Множество 2: IntegerSet(max_size=4, elements=[1, 2, 3, 4])
Объединённое множество 3: IntegerSet(max_size=5, elements=[1, 2, 3, 4, 6])
Содержит ли множество 3 элемент 8: False
```

Задание 2.

Построить модель программной системы с применением отношений (обобщения, агрегации, ассоциации, реализации) между классами. Задать атрибуты и методы классов. Реализовать (если необходимо) дополнительные классы. Продемонстрировать работу разработанной системы.

5) Система Библиотека. Читатель оформляет Заказ на Книгу. Система осуществляет поиск в Каталоге. Библиотекарь выдает Читателю Книгу на абонемент или в читальный зал. При невозвращении Книги Читателем он может быть занесен Администратором в «черный список».

Код программы:

```
class Book:
    def __init__(self, title, author, year, isbn):
        self.title = title
```

```
        self.author = author
        self.year = year
        self.isbn = isbn
        self.is_available = True
        self.location = "каталог"

    def __str__(self):
        status = "в наличии" if self.is_available else "недоступна"
        return f"'{self.title}', {self.author} ({self.year}) - {status}"

    def __eq__(self, other):
        if isinstance(other, Book):
            return self.isbn == other.isbn
        return False


class Catalog:
    def __init__(self):
        self.books = []

    def add_book(self, book):
        if book not in self.books:
            self.books.append(book)
            print(f"Книга {book.title} добавлена в каталог")
        else:
            print("Такая книга уже есть в каталоге")

    def remove_book(self, book):
        if book in self.books:
            self.books.remove(book)
            print(f"Книга {book.title} удалена из каталога")

    def search_by_title(self, title):
        result = []
        for book in self.books:
            if title.lower() in book.title.lower():
                result.append(book)
        return result

    def search_by_author(self, author):
        result = []
        for book in self.books:
            if author.lower() in book.author.lower():
                result.append(book)
        return result

    def search_by_isbn(self, isbn):
        for book in self.books:
            if book.isbn == isbn:
                return book
        return None

    def show_all_books(self):
        print("Каталог: ")
        if not self.books:
            print("<пусто>")
        else:
            for i, book in enumerate(self.books, 1):
                print(f"{i}. {book}")


class Reader:
    def __init__(self, name, reader_id):
        self.name = name
        self.reader_id = reader_id
        self.orders = []
        self.borrowed_books = []
```

```
        self.is_blacklisted = False

    def __str__(self):
        status = " (В ЧЕРНОМ СПИСКЕ)" if self.is_blacklisted else ""
        return f"Читатель: {self.name} (ID: {self.reader_id}){status}"

    def create_order(self, book, order_type):
        if self.is_blacklisted:
            print(f"{self.name} в черном списке! Заказ невозможен")
            return None

        order = Order(self, book, order_type)
        self.orders.append(order)
        print(f"Заказ оформлен: {self.name} -> {book.title} ({order_type})")
        return order

    def return_book(self, book):
        if book in self.borrowed_books:
            self.borrowed_books.remove(book)
            book.is_available = True
            book.location = "каталог"
            print(f"{self.name} вернул книгу '{book.title}'")
            return True
        else:
            print(f"У {self.name} нет такой книги")
            return False

    def show_borrowed_books(self):
        if not self.borrowed_books:
            print(f"{self.name} не взял ни одной книги")
        else:
            print(f"Книги у {self.name}:")
            for book in self.borrowed_books:
                print(f"  - {book}")


class Order:
    def __init__(self, reader, book, order_type):
        self.reader = reader
        self.book = book
        self.order_type = order_type
        self.is_completed = False
        self.is_cancelled = False

    def __str__(self):
        status = "выполнен" if self.is_completed else "отменен" if
self.is_cancelled else "активен"
        return f"Заказ: {self.reader.name} -> {self.book.title}
({self.order_type}) - {status}"


class Librarian:
    def __init__(self, name, employee_id):
        self.name = name
        self.employee_id = employee_id
        self.issued_books = []

    def __str__(self):
        return f"Библиотекарь: {self.name} (ID: {self.employee_id})"

    def process_order(self, order):
        if order.is_completed or order.is_cancelled:
            print("Заказ уже обработан")
            return False

        book = order.book
```

```
        if not book.is_available:
            print(f"Книга '{book.title}' недоступна")
            order.is_cancelled = True
            return False

        if order.reader.is_blacklisted:
            print(f"Читатель {order.reader.name} в черном списке!")
            order.is_cancelled = True
            return False

        book.is_available = False
        book.location = order.order_type
        order.reader.borrowed_books.append(book)
        order.is_completed = True
        self.issued_books.append(book)

        print(f"{self.name} выдал книгу '{book.title}' {order.reader.name}
({order.order_type})")
        return True

    @staticmethod
    def check_return(reader, book):
        return reader.return_book(book)


class Administrator(Librarian):
    def __init__(self, name, employee_id):
        super().__init__(name, employee_id)
        self.blacklist = []

    def __str__(self):
        return f"Администратор: {self.name} (ID: {self.employee_id})"

    def add_to_blacklist(self, reader, reason=""):
        if reader not in self.blacklist:
            reader.is_blacklisted = True
            self.blacklist.append(reader)
            reason_text = f"по причине: {reason}" if reason else "<не указано>"
            print(f"{self.name} занес {reader.name} в черный список
{reason_text}")
        else:
            print(f"{reader.name} уже в черном списке")

    def remove_from_blacklist(self, reader):
        if reader in self.blacklist:
            reader.is_blacklisted = False
            self.blacklist.remove(reader)
            print(f"{self.name} удалил {reader.name} из черного списка")
        else:
            print(f"{reader.name} не найден в черном списке")

    def show_blacklist(self):
        print("ЧЕРНЫЙ СПИСОК:")
        if not self.blacklist:
            print("<пусто>")
        else:
            for reader in self.blacklist:
                print(f"  - {reader.name} (ID: {reader.reader_id})")


print("СИСТЕМА БИБЛИОТЕКА")
catalog = Catalog()

book1 = Book("Война и мир", "Лев Толстой", 1869, "978-5-699-38289-1")
book2 = Book("Преступление и наказание", "Федор Достоевский", 1866, "978-5-17-
123456-7")
book3 = Book("Анна Каренина", "Лев Толстой", 1877, "978-5-699-38290-7")
```

```
book4 = Book("Мастер и Маргарита", "Михаил Булгаков", 1967, "978-5-389-12345-6")

print("\nДобавление книг в каталог")
catalog.add_book(book1)
catalog.add_book(book2)
catalog.add_book(book3)
catalog.add_book(book4)

catalog.show_all_books()

print("\nСоздание сотрудников")
librarian = Librarian("Анна Петровна", "LIB001")
admin = Administrator("Иван Иванович", "ADM001")

print(librarian)
print(admin)

print("\nСоздание читателей ")
reader1 = Reader("Петр Сидоров", "R001")
reader2 = Reader("Мария Иванова", "R002")
reader3 = Reader("Алексей Смирнов", "R003")

print(reader1)
print(reader2)
print(reader3)

print("\nПоиск книг по автору 'Толстой' ")
tolstoy_books = catalog.search_by_author("Толстой")
for book in tolstoy_books:
    print(f"Найдено: {book}")

print("\nОформление заказов ")
order1 = reader1.create_order(book1, "абонемент")
order2 = reader2.create_order(book2, "читальный зал")
order3 = reader3.create_order(book4, "абонемент")

print("\nОбработка заказов библиотекарем ")
librarian.process_order(order1)
librarian.process_order(order2)

print("\nПопытка заказать уже выданную книгу ")
order_again = reader3.create_order(book1, "абонемент")
librarian.process_order(order_again)

print("\nВозврат книги ")
reader1.show_borrowed_books()
librarian.check_return(reader1, book1)
reader1.show_borrowed_books()

print("\nЗаказ вернувшейся книги ")
order_new = reader3.create_order(book1, "абонемент")
librarian.process_order(order_new)

print("\nРабота с черным списком ")
print(reader2)
admin.add_to_blacklist(reader2, "не вернул книгу вовремя")
admin.show_blacklist()
print(reader2)

print("\nПопытка заказа книгу читателем из ЧС ")
bad_order = reader2.create_order(book3, "читальный зал")
if bad_order:
    librarian.process_order(bad_order)

print("\nУдаление из черного списка ")
admin.remove_from_blacklist(reader2)
admin.show_blacklist()
```

```
print(reader2)

print("\nИтоговое состояние ")
catalog.show_all_books()
reader1.show_borrowed_books()
reader2.show_borrowed_books()
reader3.show_borrowed_books()
```

Спецификация ввода

Ввод не требуется – основная программа приводит тестовое исполнение функциональности.

Спецификация вывода

В консоль выводятся результаты добавления книг в каталог, создания сотрудников, создания читателей, поиска книг по автору, оформления заказов, обработки заказов библиотекарем, попытки заказать существующую книгу, возврат книги, работы с чёрным списком.

Рисунки с результатами работы программы

```
E:\kurs3\SPP\LAB1\.venv\Scripts\python.exe E:\kurs3\SPP\LAB1\spp_po13\reports\Markovsky\2\src\2.py
СИСТЕМА БИБЛИОТЕКА

Добавление книг в каталог
Книга Война и мир добавлена в каталог
Книга Преступление и наказание добавлена в каталог
Книга Анна Каренина добавлена в каталог
Книга Мастер и Маргарита добавлена в каталог
Каталог:
1. 'Война и мир', Лев Толстой (1869) - в наличии
2. 'Преступление и наказание', Федор Достоевский (1866) - в наличии
3. 'Анна Каренина', Лев Толстой (1877) - в наличии
4. 'Мастер и Маргарита', Михаил Булгаков (1967) - в наличии

Создание сотрудников
Библиотекарь: Анна Петровна (ID: LIB001)
Администратор: Иван Иванович (ID: ADM001)

Создание читателей
Читатель: Петр Сидоров (ID: R001)
Читатель: Мария Иванова (ID: R002)
Читатель: Алексей Смирнов (ID: R003)

Поиск книг по автору 'Толстой'
Найдено: 'Война и мир', Лев Толстой (1869) - в наличии
Найдено: 'Анна Каренина', Лев Толстой (1877) - в наличии

Оформление заказов
Заказ оформлен: Петр Сидоров -> Война и мир (абонемент)
Заказ оформлен: Мария Иванова -> Преступление и наказание (читальный зал)
```

```
Оформление заказов
Заказ оформлен: Петр Сидоров -> Война и мир (абонемент)
Заказ оформлен: Мария Иванова -> Преступление и наказание (читальный зал)
Заказ оформлен: Алексей Смирнов -> Мастер и Маргарита (абонемент)

Обработка заказов библиотекарем
Анна Петровна выдал книгу 'Война и мир' Петр Сидоров (абонемент)
Анна Петровна выдал книгу 'Преступление и наказание' Мария Иванова (читальный зал)

Попытка заказать уже выданную книгу
Заказ оформлен: Алексей Смирнов -> Война и мир (абонемент)
Книга 'Война и мир' недоступна

Возврат книги
Книги у Петр Сидоров:
  - 'Война и мир', Лев Толстой (1869) - недоступна
Петр Сидоров вернул книгу 'Война и мир'
Петр Сидоров не взял ни одной книги

Заказ вернувшейся книги
Заказ оформлен: Алексей Смирнов -> Война и мир (абонемент)
Анна Петровна выдал книгу 'Война и мир' Алексей Смирнов (абонемент)

Работа с черным списком
Читатель: Мария Иванова (ID: R002)
Иван Иванович занес Мария Иванова в черный список по причине: не вернул книгу вовремя
ЧЕРНЫЙ СПИСОК:
  - Мария Иванова (ID: R002)
Читатель: Мария Иванова (ID: R002) (В ЧЕРНОМ СПИСКЕ)
```

```
Попытка заказа книгу читателем из ЧС
Мария Иванова в черном списке! Заказ невозможен

Удаление из черного списка
Иван Иванович удалил Мария Иванова из черного списка
ЧЕРНЫЙ СПИСОК:
<пусто>
Читатель: Мария Иванова (ID: R002)

Итоговое состояние
Каталог:
1. 'Война и мир', Лев Толстой (1869) - недоступна
2. 'Преступление и наказание', Федор Достоевский (1866) - недоступна
3. 'Анна Каренина', Лев Толстой (1877) - в наличии
4. 'Мастер и Маргарита', Михаил Булгаков (1967) - в наличии
Петр Сидоров не взял ни одной книги
Книги у Мария Иванова:
  - 'Преступление и наказание', Федор Достоевский (1866) - недоступна
Книги у Алексей Смирнов:
  - 'Война и мир', Лев Толстой (1869) - недоступна

Process finished with exit code 0
```

Вывод:

Я закрепил навыки объектно-ориентированного программирования на языке Python.