МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”
КАФЕДРА ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

ОТЧЁТ

по лабораторной работе №3

Выполнил
студент 3 курса
группы ПО-9
Ничингер Кирилл Александрович

Проверил:
Крощенко А. А.

Брест 2024

Цель работы: научиться создавать и использовать классы в программах на языке программирования Java.

Вариант 4

Задание 1: реализовать класс прямоугольника, заданного длинами двух сторон. Предусмотреть возможность определения площади и периметра, а так же логические методы, определяющие, является ли прямоугольник квадратом и существует ли такой прямоугольник. Конструктор должен позволить создавать объекты с начальной инициализацией. Реализовать метод equals, выполняющий сравнение объектов данного типа.

Код программы:

```
package Task1;
public class Rectangle {
    private double length;
    private double width;

    public Rectangle(){
        this.length = 1;
        this.width = 1;
    }

    public Rectangle(double length, double width) throws Exception {
        if (length <= 0 || width <= 0)
            throw new Exception("Length and width must be positive");
        this.length = length;
        this.width = width;
    }

    public double getLength() {
        return length;
    }

    public double getWidth() {
        return width;
    }

    public void setWidth(double width) {
        if (width >= 0)
            this.width = width;
    }

    public void setLength(double length) {
        if (length >= 0)
            this.length = length;
    }

    public double getArea() {
        return length * width;
    }
```

```
    public double getPerimeter() {
        return 2 * (length + width);
    }


    public boolean isSquare() {
        return length == width;
    }

    @Override
    public boolean equals(Object obj) {
        if (obj == null || getClass() != obj.getClass()) {
            return false;
        }
        Rectangle rect = (Rectangle) obj;
        return Double.compare(rect.length, length) == 0 &&
Double.compare(rect.width, width) == 0;
    }

    @Override
    public String toString() {
        return "rectangle(" +
                "length = " + length +
                ", width = " + width + ")";
    }
}
```

Входные данные:

```
package Task1;
public class RectangleTest {
    public static void main(String[] args) throws Exception {
        Rectangle rect1 = new Rectangle();
        Rectangle rect2 = new Rectangle(1, 1);

        if (rect1.equals(rect2))
            System.out.println("Rectangles are equal");
        else
            System.out.println("Rectangles are not equal");

        if (rect1.isSquare())
            System.out.print(rect1 + " is square;" +
                    " its area = " + rect1.getArea() +
                    "; its perimeter = " + rect1.getPerimeter());
    }
}
```

Результат работы программы:

```
D:\SDK\JDK\bin\java.exe "-javaagent:D:\JetBrains\IntelliJ IDEA Community Edition 20
Rectangles are equal
rectangle(length = 1.0, width = 1.0) is square; its area = 1.0; its perimeter = 4.0
Process finished with exit code 0
```

Задание 2: реализовать автоматизированную систему в библиотеке. Составить программу, которая содержит текущую информацию о книгах в библиотеке. Сведения о книгах (Book) содержат:

• Номер УДК;
• Фамилию и инициалы автора;
• Название;
• Год издания;
• Количество экземпляров в библиотеке;
• Количество страниц;
• Количество томов;
• ФИО читателя, взявшего книгу (при наличии);
• Срок сдачи книги (если была взята).

Программа должна обеспечивать:

• Формирование общего списка книг;
• Формирование списка книг, старше n лет;
• Формирование списка книг, взятых на чтение;
• Формирование списка книг, взятых на чтение с выводом личной информации о читателях;
• Формирование списка книг, которые задержаны читателем дольше указанного срока.

Код программы:

```
package Task2;

import java.util.Date;
import java.util.HashMap;

class Book {
    private String udcCode;
    private String author;
    private String title;
    private int publicationYear;
    private int pagesCount;
    private int volumesCount;
    private int copiesCount;
    private HashMap<String, Date> borrowedBooks;
    public Book(String udkNumber, String author, String title, int publicationYear,
                int pageCount, int volumesCount, int copiesCount) {
        this.udcCode = udkNumber;
        this.author = author;
        this.title = title;
        this.publicationYear = publicationYear;
        this.pagesCount = pageCount;
        this.volumesCount = volumesCount;
        this.copiesCount = copiesCount;
        this.borrowedBooks = new HashMap<>();
    }
```

```
public String getUdcCode() {
    return udcCode;
}

public void setUdcCode(String udcCode) {
    if (udcCode != null && !udcCode.isEmpty()){
        this.udcCode = udcCode;
    }
}

public String getAuthor() {
    return author;
}

public void setAuthor(String author) {
    if (author != null && !author.isEmpty()){
        this.author = author;
    }
}

public String getTitle() {
    return title;
}

public void setTitle(String title) {
    if (title != null && !title.isEmpty()){
        this.title = title;
    }
}

public int getPublicationYear() {
    return publicationYear;
}

public void setPublicationYear(int publicationYear) {
    if (publicationYear > 0){
        this.publicationYear = publicationYear;
    }
}

public int getPagesCount() {
    return pagesCount;
}

public void setPagesCount(int pagesCount) {
    if (pagesCount > 0){
        this.pagesCount = pagesCount;
    }
}

public int getVolumesCount() {
    return volumesCount;
}

public void setVolumesCount(int volumesCount) {
    if (volumesCount > 0){
        this.volumesCount = volumesCount;
    }
}
```

```
    public int getCopiesCount() {
        return copiesCount;
    }

    public void setCopiesCount(int copiesCount) {
        if (copiesCount > 0){
            this.copiesCount = copiesCount;
        }
    }

    @Override
    public String toString() {
        StringBuilder readers = new StringBuilder();
        if (!borrowedBooks.isEmpty()){
            readers.append(", readers( ");
            borrowedBooks.forEach((readerName, deadline) -> {
                readers.append(readerName).append(",
").append(deadline).append(";");
            });
            readers.append(")");
        }

        return "book(" +
                "udkNumber = " + udcCode +
                ", author = " + author +
                ", title = " + title +
                ", publicationYear = " + publicationYear +
                ", pageCount = " + pagesCount +
                ", volumesCount = " + volumesCount +
                ", copiesCount = " + copiesCount +
                readers + ")";
    }

    public Book borrowBook(String reader, Date deadline){
        if (reader != null && deadline != null && copiesCount >= 1){
            copiesCount--;
            borrowedBooks.put(reader, deadline);
            return this;
        }
        else return null;
    }

    public void returnBook(String reader){
        borrowedBooks.remove(reader);
    }

    public HashMap<String, Date> getBorrowedBooks(){
        return borrowedBooks;
    }
}
```

```java
package Task2;

import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.Map;

public class Library {
    private ArrayList<Book> books;

    public Library() {
        books = new ArrayList<>();
    }

    public void addBook(Book book){
        if (!books.contains(book))
            books.add(book);
    }

    public void removeBook(Book book){
        books.remove(book);
    }

    public Book borrowBook(Book book, String reader){
        if (books.contains(book)){
            int bookBorrowingDays = 30;
            Calendar calendar = Calendar.getInstance();
            calendar.setTime(new Date());
            calendar.add(Calendar.DAY_OF_YEAR, bookBorrowingDays);
            return book.borrowBook(reader, calendar.getTime());
        }
        else return null;
    }

    public void returnBookToLibrary(Book book, String reader){
        if (books.contains(book)){
            book.returnBook(reader);
        }
    }

    public ArrayList<Book> getAllBooks() {
        return books;
    }

    public ArrayList<Book> getBooksOlderThan(int year) {
        ArrayList<Book> result = new ArrayList<>();
        for (Book book : books)
            if (book.getPublicationYear() <
Calendar.getInstance().get(Calendar.YEAR) - year)
                result.add(book);
        return result;
    }
```

```
    public ArrayList<Book> getBooksOnLoan() {
        ArrayList<Book> result = new ArrayList<>();
        for (Book book : books) {
            if (!book.getBorrowedBooks().isEmpty()) {
                result.add(book);
            }
        }
        return result;
    }

    public ArrayList<Book> getOverdueBooks() {
        ArrayList<Book> result = new ArrayList<>();
        Date date = new Date();
        for (Book book : books)
            for (Map.Entry<String, Date> entry :
book.getBorrowedBooks().entrySet())
                if (entry.getValue().after(date)){
                    result.add(book);
                    break;
                }
        return result;
    }

    public Book getFirstBookByTitle(String title){
        for (Book book : books)
            if (book.getTitle().equals(title))
                return book;
        return null;
    }
}
```

Входные данные:

```
package Task2;

public class LibraryTest {
    public static void main(String[] args) throws Exception {
        Library library = new Library();
        Book book1 = new Book("123", "Kirlil",
                "Книга1",
                2010, 666, 1, 1);
        Book book2 = new Book("321", "Kirill",
                "Книга2",
                2015, 666, 3, 2);
        Book book3 = new Book("213", "Kilirl",
                "Книга3",
                2020, 666, 6, 1);
        library.addBook(book1);
        library.addBook(book2);
        library.addBook(book3);
        System.out.println("All books:");
        for (Book bk : library.getAllBooks()){
            System.out.println(bk);
        }

        System.out.println("\nBooks on loan:");
        library.borrowBook(book1, "Kir");
        for (Book bk : library.getBooksOnLoan()){
            System.out.println(bk);
        }

        System.out.println("\nBooks older then 5 years:");
        for (Book bk : library.getBooksOlderThan(5)){
            System.out.println(bk);
        }
    }
}
```

Результат работы программы:

```
All books:
book(udkNumber = 123, author = Kirlil, title = Книга1, publicationYear = 2010, pageCount = 666,
 volumesCount = 1, copiesCount = 1)
book(udkNumber = 321, author = Kirill, title = Книга2, publicationYear = 2015, pageCount = 666,
 volumesCount = 3, copiesCount = 2)
book(udkNumber = 213, author = Kilirl, title = Книга3, publicationYear = 2020, pageCount = 666,
 volumesCount = 6, copiesCount = 1)

Books on loan:
book(udkNumber = 123, author = Kirlil, title = Книга1, publicationYear = 2010, pageCount = 666,
 volumesCount = 1, copiesCount = 0, readers( Kir, Wed Mar 06 23:26:44 MSK 2024;))

Books older then 5 years:
book(udkNumber = 123, author = Kirlil, title = Книга1, publicationYear = 2010, pageCount = 666,
 volumesCount = 1, copiesCount = 0, readers( Kir, Wed Mar 06 23:26:44 MSK 2024;))
book(udkNumber = 321, author = Kirill, title = Книга2, publicationYear = 2015, pageCount = 666,
 volumesCount = 3, copiesCount = 2)
```

Вывод: я научился создавать и использовать классы в программах на языке программирования Java.