UTM
UNIVERSITI TEKNOLOGI MALAYSIA

Library Management System

Group 8

SECJ2154 - OBJECT-ORIENTED PROGRAMMING USING JAVA

MADAM LIZAWATI MI YUSUF

The Group Members

Anushka

Worked On:

- DeleteBook.java
- Exit.java
- Members.java
- Order.java
- User.java

Afiq

Worked On:

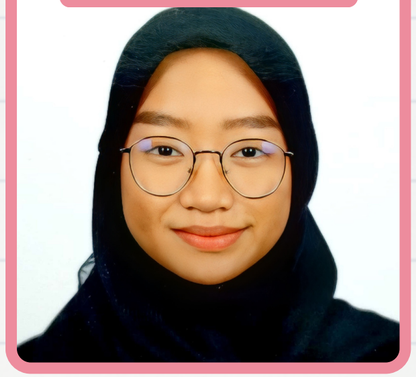
- AddBook.java
- Admin.java
- Book.java
- BorrowBook.java
- Borrowing.java

Danesh

Worked On:

- CalculateFine.java
- Database.java
- Main.java
- PlaceOrder.java
- ReturnBook.java

Aina



Worked On:

- DeleteAllData.java
- IOOperation.java
- Search.java
- ViewBook.java
- ViewOrders.java

Table of Contents

Project Description

The system we developed as our final project for Object Oriented Programming is Library Management System. This system ideally serves as a way to manage a library in a much more efficient way by having a digital database of all the books available as well as keeping track of borrowed books. This system can be used by running the java application provided in the GitHub repository. Our system's main features include, view books, add books, search for books, borrow books as well as calculate and impose fines on members who fail to return the book within the particular time.

Problem Statement

Although online libraries have been the new trend, many physical libraries still maintain their libraries the traditional way. However, as times change, many of these libraries have shown interest in embracing technology to help assist in everyday tasks such as maintaining the library database as well as keeping track of borrowed books instead of the traditional library cards. One such library is Otter's Library, a traditional library embracing technology.

Mr. Otter has been facing difficulties involving daily tasks of managing his library and overall, his library would be in a much better state with the help of a simple system such as a Library Management System.

Objective & Scope

Objective

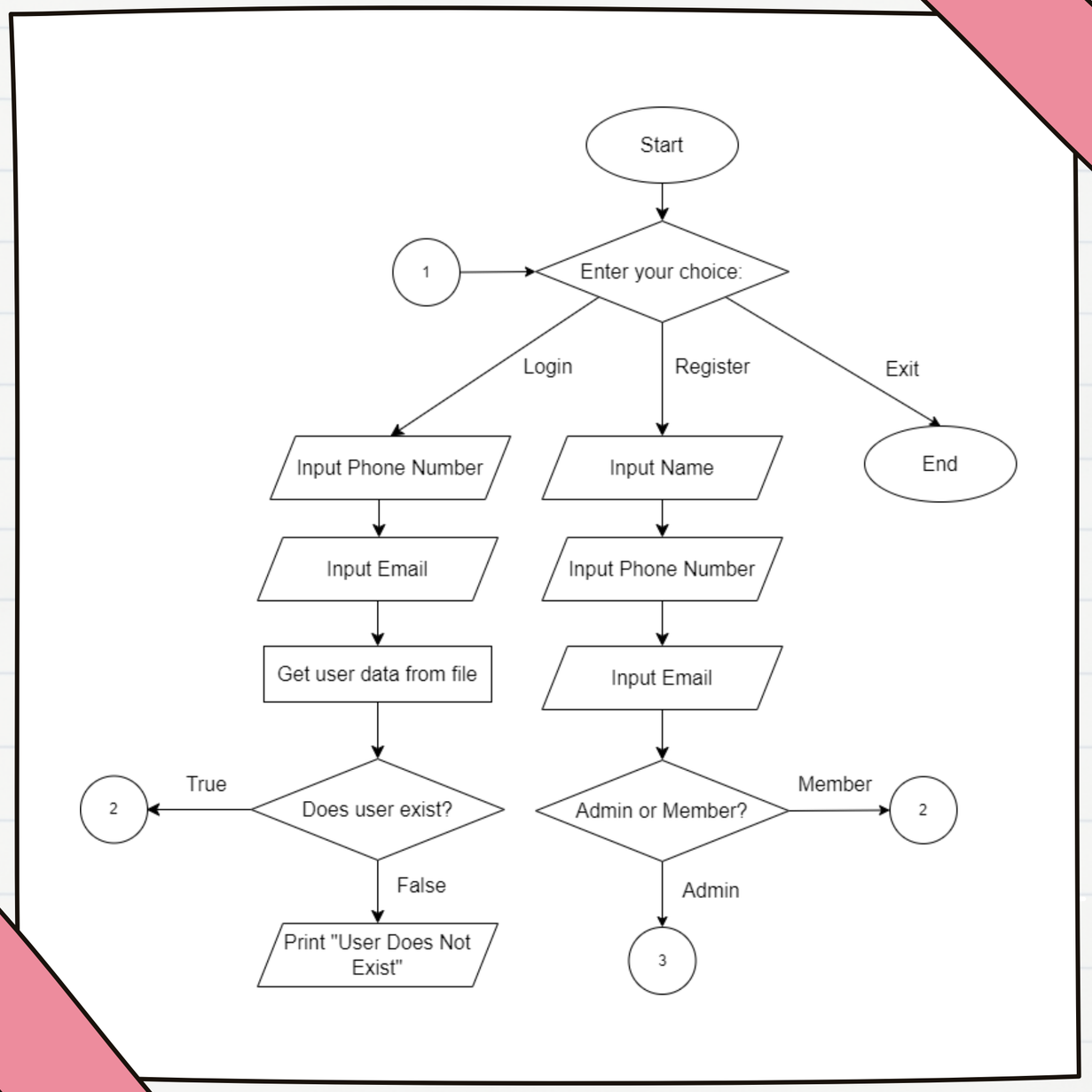
- To aid in the borrowing and keeping track of the status of books.
- To enable librarian and members to search for books and survey the number of copies available.
- To ensure the security as well as integrity of the online library database.

Scope

The scope of this project will mainly focus on providing a platform for Otter's Library which aids everyday tasks of the library in a more efficient way. This system can provide information regarding the members, borrowed books, return dates as well as other books that are available in the library.

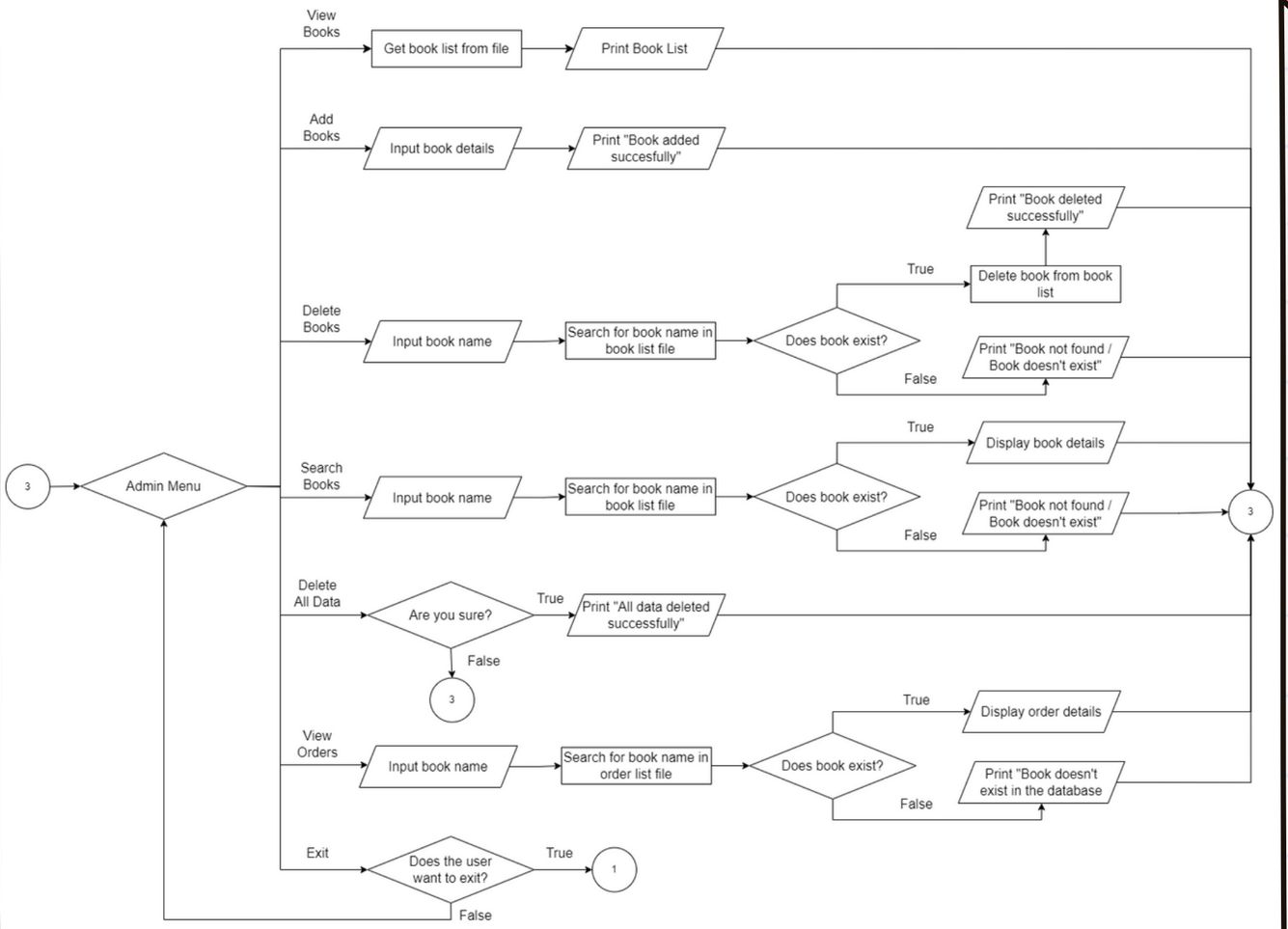
Analysis & Design

FLOW CHART (MAIN MENU)



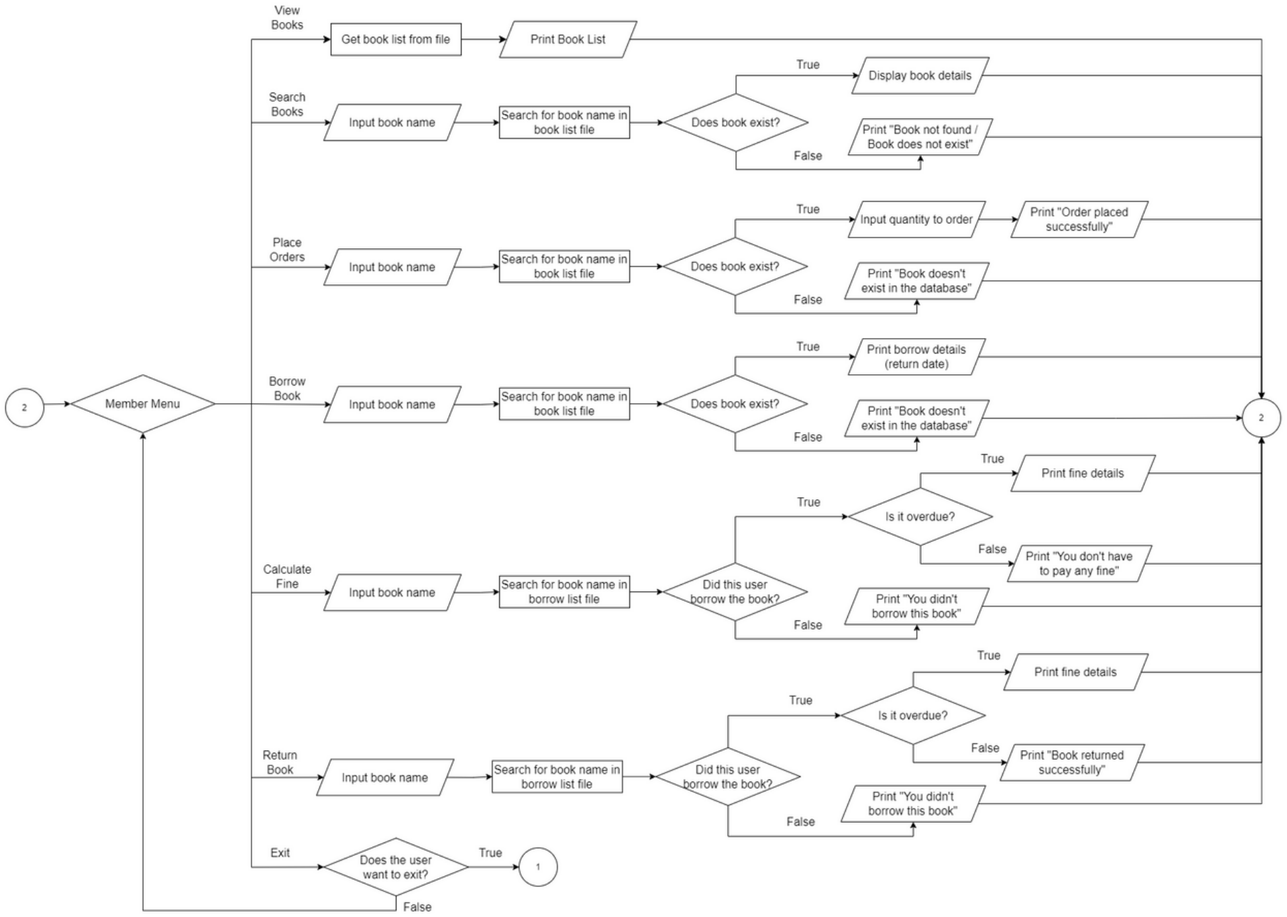
Analysis & Design

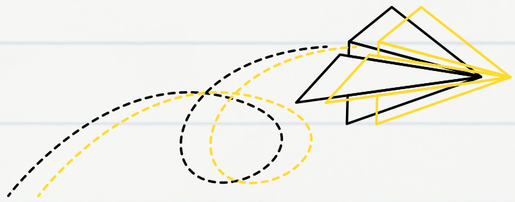
FLOW CHART (ADMIN)



Analysis & Design

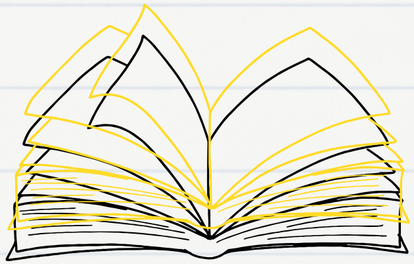
FLOW CHART (MEMBER)

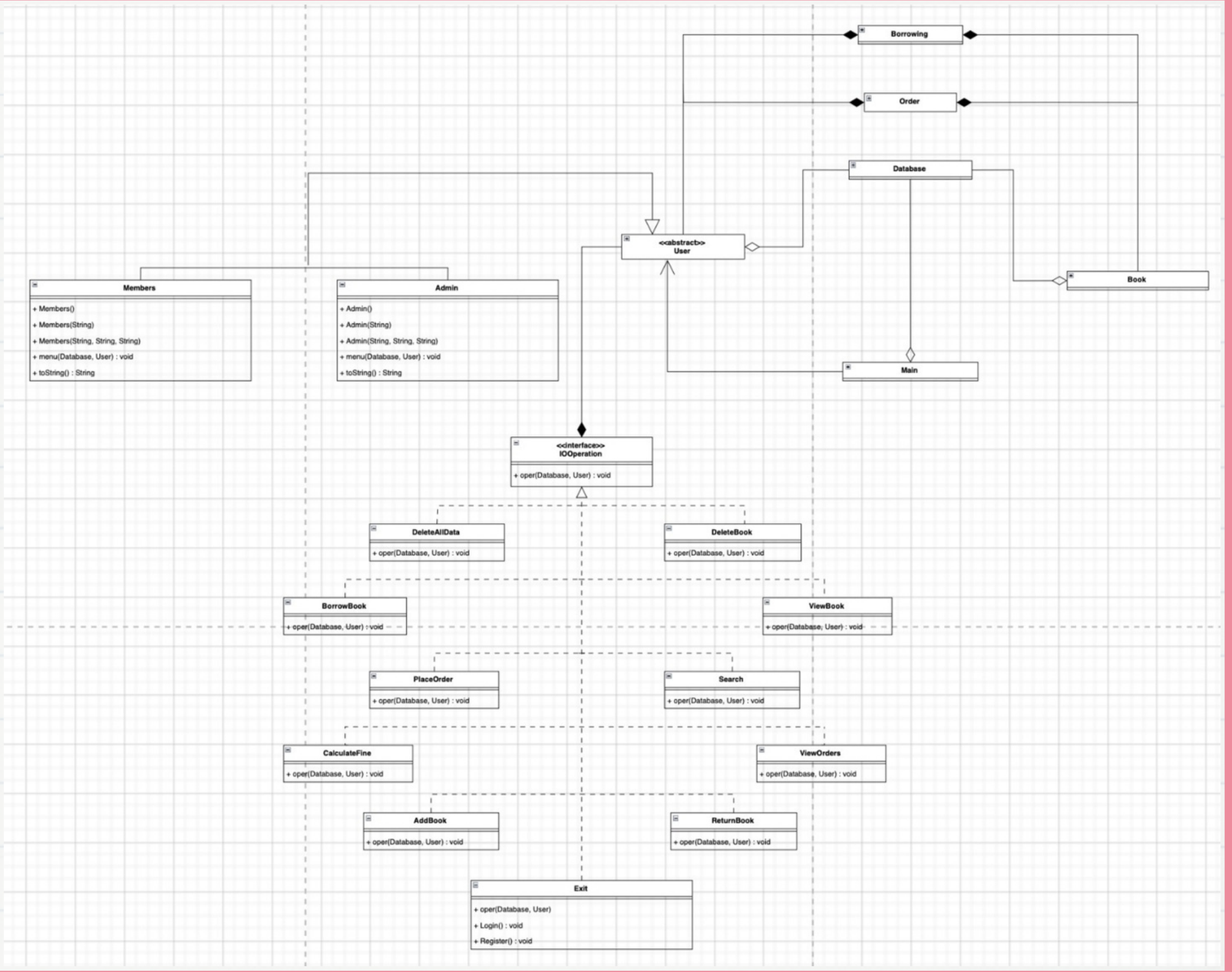




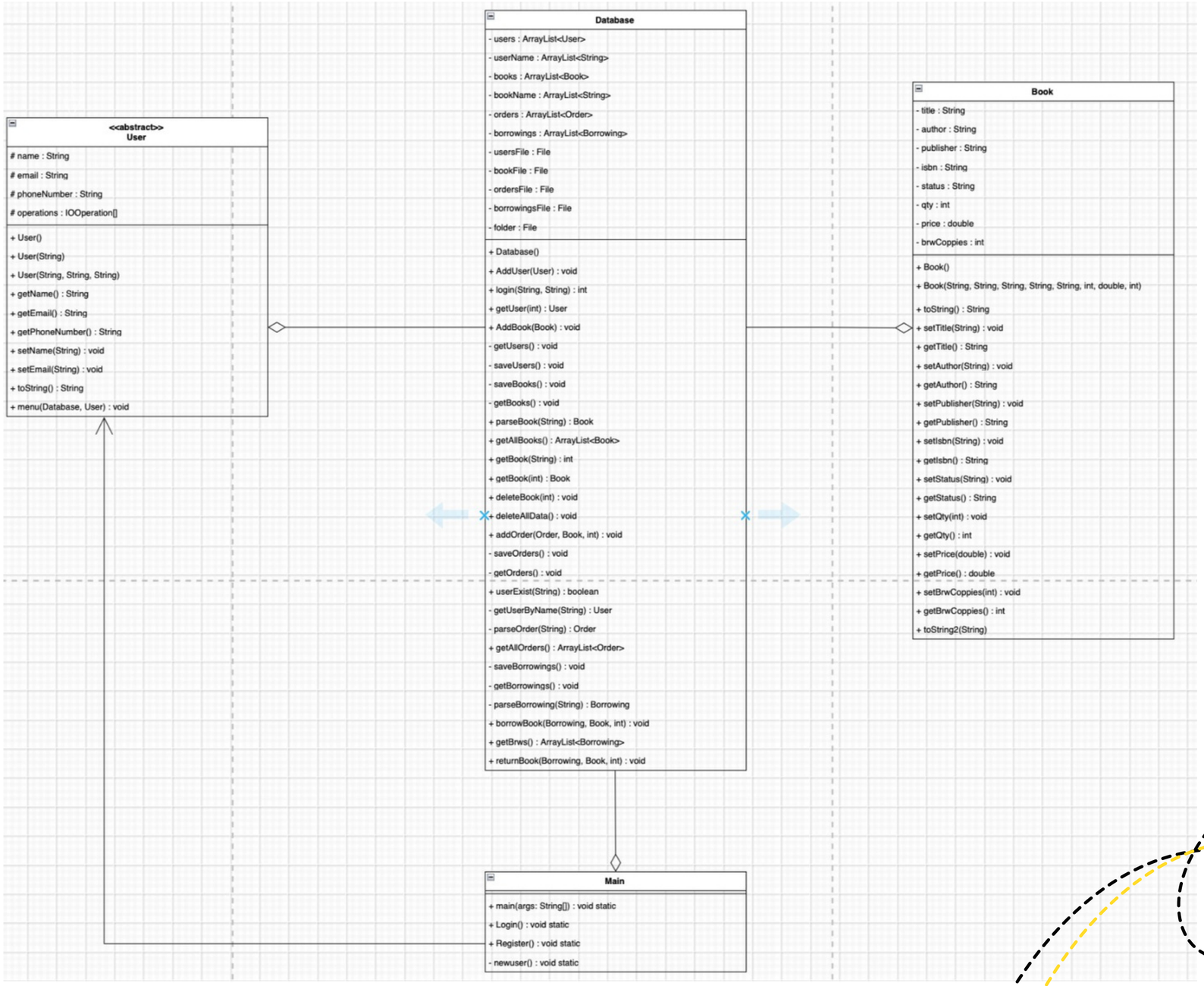
The

Class Diagram

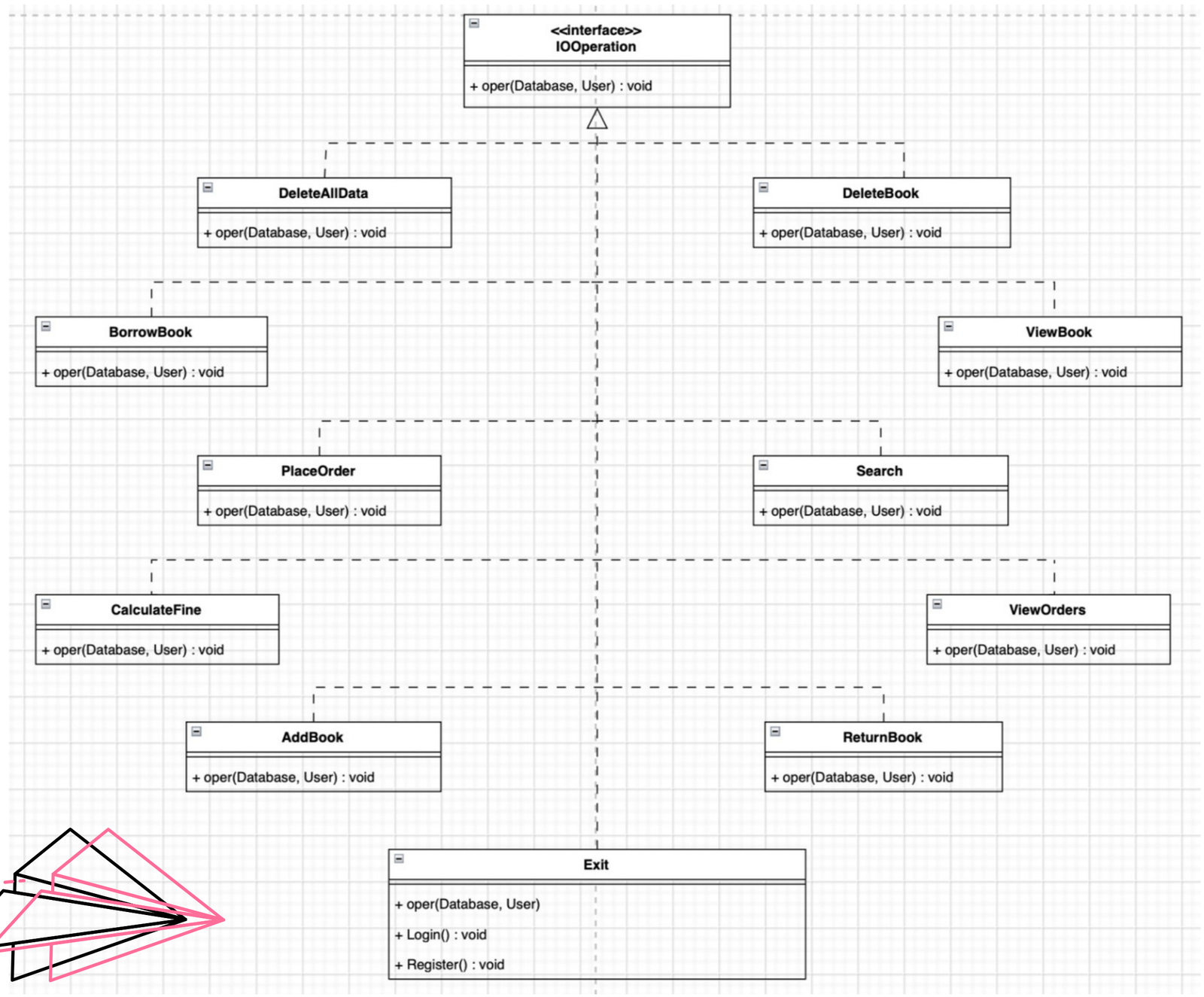




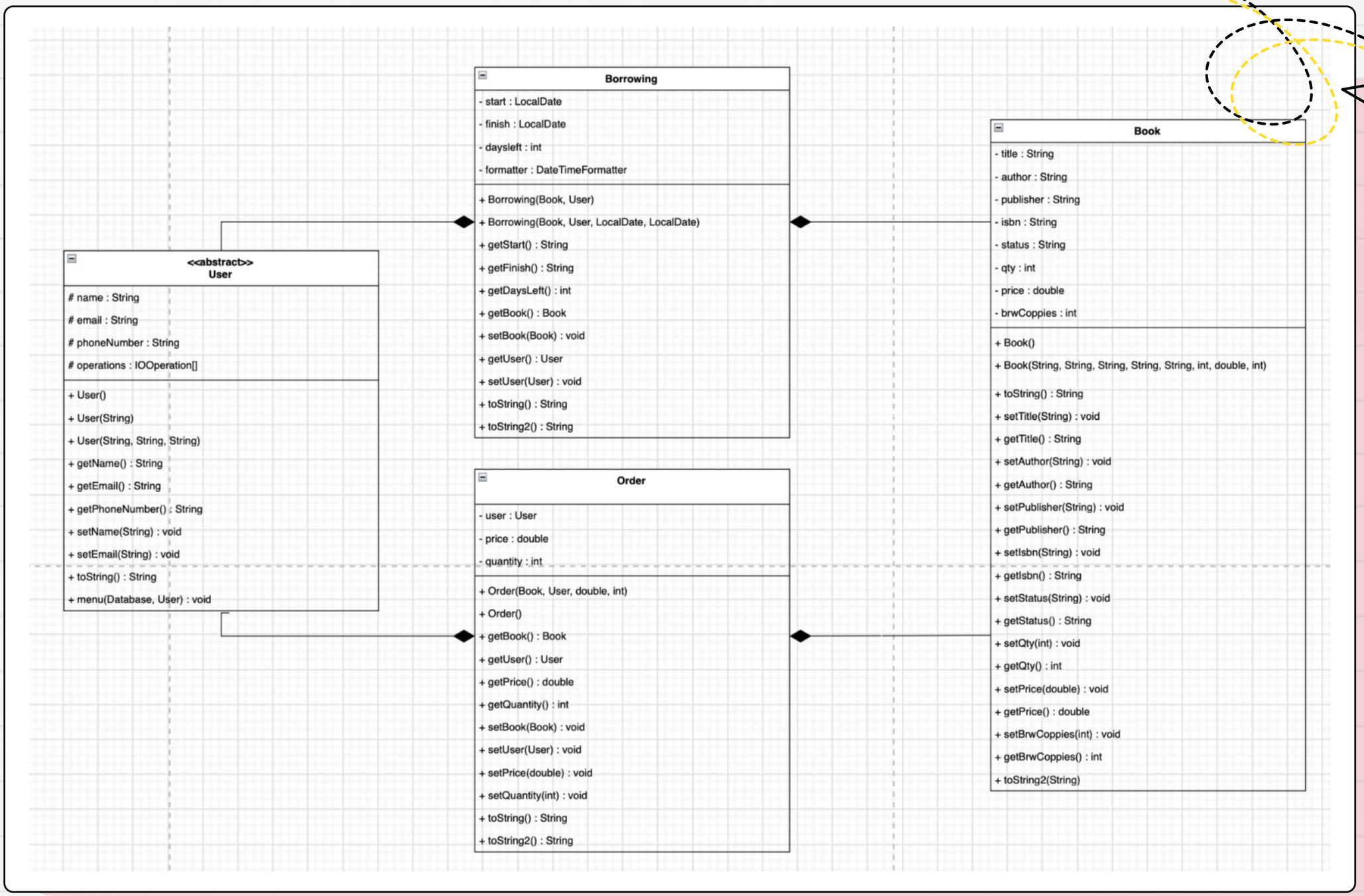
Main/Primary Classes



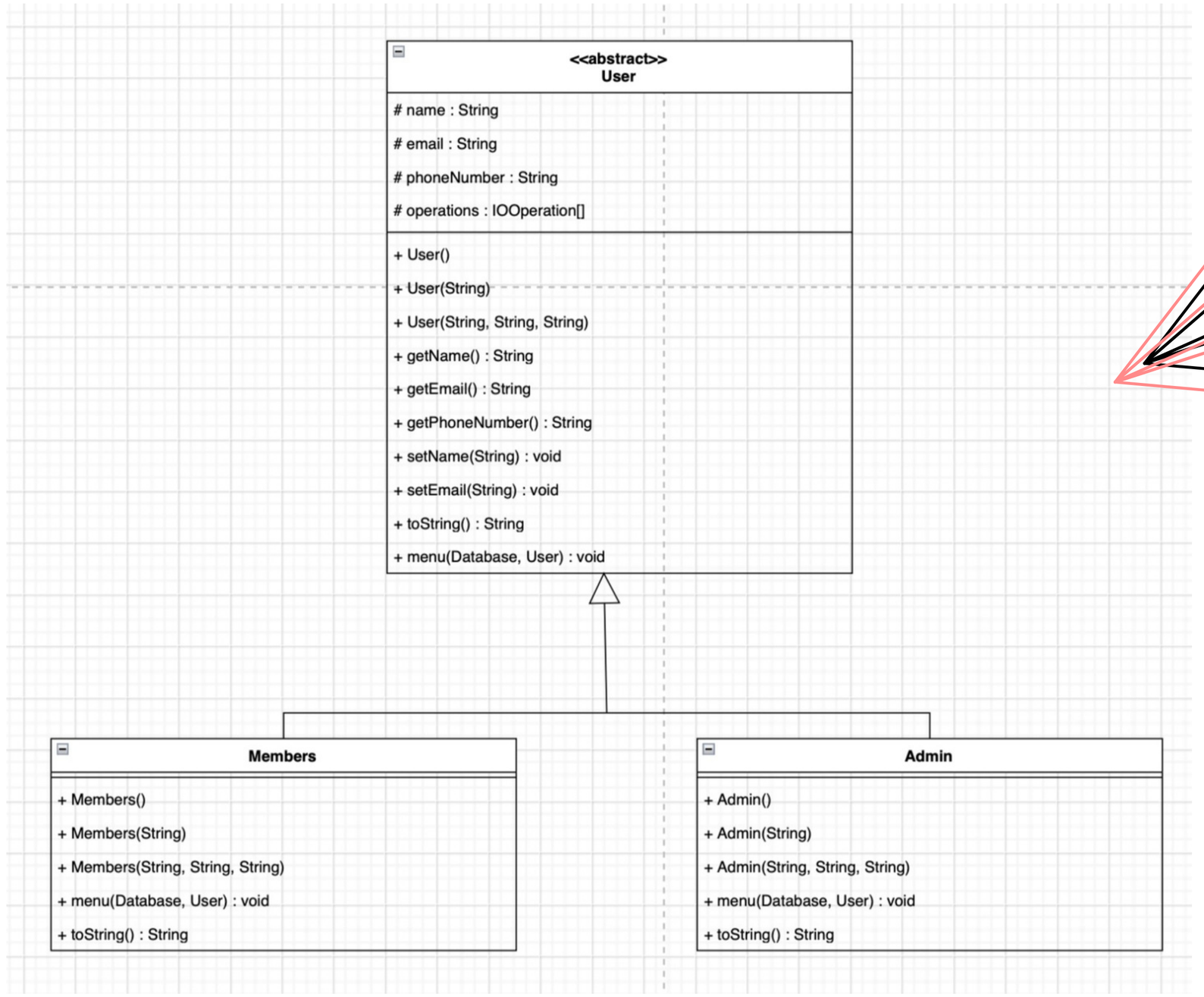
Interface Classes



Composition Classes



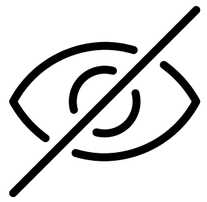
Inheritance Classes



Concept Implemented

Data Hiding

...is the ability to hide data of an object from another within a program. It also protects attributes from accidental corruption or “breaking” of a code.



```java
package Library;
public class Book {

    private String title;
    private String author;
    private String publisher;
    private String isbn;
    private String status;
    private int qty;
    private double price;
    private int brwCoppies;
```

```java
public class Database {

    private ArrayList<User> users = new ArrayList<User>();
    private ArrayList<String> userName = new ArrayList<String>();
    private ArrayList<Book> books = new ArrayList<Book>();
    private ArrayList<String> bookName = new ArrayList<String>();
    private ArrayList<Order> orders = new ArrayList<Order>();
    private ArrayList<Borrowing> borrowings = new ArrayList<Borrowing>();
```

Concept Implemented

Encapsulation

...combines data and behavior in one package while also hiding the implementation of the data from the user.

```
public String toString(){
    return name + "<N/>" + email + "<N/>" + phoneNumber + "<N/>" + "Admin";
}
```

```
public Book getBook() {
    return this.book;
}

public User getUser() {
    return this.user;
}

public double getPrice() {
    return this.price;
}

public int getQuantity() {
    return this.quantity;
}
```

Concept Implemented

Composition

...is a form of Aggregation. It has a very strong ownership or binding and its part belongs to one whole.

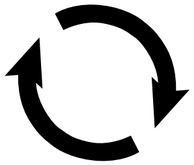
```java
public Borrowing(Book book, User user) {
    start  = LocalDate.now();
    finish = start.plusDays(daysToAdd:14);
    daysleft = Period.between(start, finish).getDays();
    this.book = book;
    this.user = user;
}
```

```java
public void oper(Database database,User user){

    System.out.print(s:"Enter book name to delete : ");
    String bookName = s.nextLine();


    int i = database.getBook(bookName);
    if (i>-1){
        database.deleteBook(i);
        System.out.println(x:"Book deleted successfully\n");
    }else{
        System.out.println(x:"Book not found / Book Doesn't exist\n");
    }
```

Concept Implemented

Inheritance

...consists of a 'Superclass' which is a generalized class and a 'Subclass' which is a specialized class that inherits methods and instance variables from the Superclass.



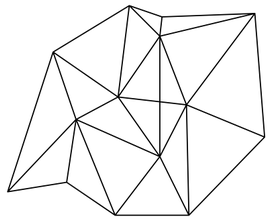
```java
public class Members extends User {

    public Members(){};

    public Members(String name){
        super(name);
        this.operations = new IOOperation[]{
            new ViewBook(),
            new Search(),
            new PlaceOrder(),
            new BorrowBook(),
            new CalculateFine(),
            new ReturnBook(),
            new Exit()

        };
    }
```

```java
package Library;
import java.util.*;

public class Admin extends User{

```

Concept Implemented

Polymorphism

...is when the same method is able to execute different behaviours. It changes the way it behaves depending on the way it is invoked.



```java
public class BorrowBook implements IOOperation{
    Scanner s = new Scanner(System.in);
    public void oper(Database database,User user){

        System.out.println(x:"Enter Book You Wanna borrow : ");
        String bookName = s.nextLine();
```

```java
public String toString(){
    return name + "<N/>" + email + "<N/>" + phoneNumber + "<N/>" + "Admin";
}
```

Concept Implemented

Exception Handling

...handles the process of responding to instances that are usually bad or unforeseen. It is used to ensure a system crash does not occur.

```
public Database(){
    if (!folder.exists()){
        folder.mkdir();
    }
    if (!usersfile.exists()){
        try{
            usersfile.createNewFile();
        }catch (Exception e){}
    }
    if (!bookfile.exists()){
        try{
            bookfile.createNewFile();
        }catch (Exception e){}
    }
    if (!ordersfile.exists()){
        try{
            ordersfile.createNewFile();
        }catch (Exception e){}
    }
    if (!borrowingsfile.exists()){
        try{
            borrowingsfile.createNewFile();
        }catch (Exception e){}
    }
    getUsers();
    getBooks();
    getOrders();
    getBorrowings();
}
```

System Demonstration