1. **To develop a system that handles book-related operations within a library context.**

Requirements:
Book Class:
Attributes:
bookId: Integer (Unique identifier for the book)
title: String
author: String
isAvailable: Boolean (availability status of the book)
Methods:
Constructor to initialize all the attributes
Getters and setters for all attributes
toString(): A method to return the book's information in a string format
Static Features:
A static variable in the Book class to keep track of the total number of books.
A static method in the Book class that returns the count of total books.
Main Class:
A Main class for running the operations.
Implement a simple command-line user interface that allows users to perform the aforementioned operations.
Instructions:
Book Class Implementation:
Create the Book class with the specified fields, constructors, and methods.
Static Features:
Use a static counter in the Book class to keep track of how many books have been added to the library.

Interaction Through Main Class:
Create 3 instances of book and show details, do the R&D of getters to get specific details of a book and setter method to update specific details of book, and static methods defined in a class

```
public class Main {
    public static void main(String[] args) {
        // Create three book instances
        Book book1 = new Book(1, "1984", "George Orwell", true);
        Book book2 = new Book(2, "To Kill a Mockingbird", "Harper Lee", true);
        Book book3 = new Book(3, "The Great Gatsby", "F. Scott Fitzgerald", false);

        // Display initial details of all books
        book1.displayInfo();
        book2.displayInfo();
        book3.displayInfo();

        // Use getters to get specific details
        System.out.println("Title of Book 1: " + book1.getTitle());
        System.out.println("Author of Book 2: " + book2.getAuthor());
        System.out.println("Availability of Book 3: " + (book3.isAvailable() ? "Available" : "Not 
Available"));

        // Use setters to update specific details
        book3.setTitle("The Great Gatsby (Updated Edition)");
        book3.setAvailable(true);

        // Display updated details of Book 3
        System.out.println("Updated details of Book 3:");
        book3.displayInfo();

        // Display total number of books created
        System.out.println("Total number of books: " + Book.getBookCount());
    }
}
```

Deliverables:
Complete Java code implementation for the Book details management within Library Management System.
Code should be well-commented to explain significant areas and decisions.
A concise report summarizing the structure and functionality of your system, highlighting how OOP concepts are applied.
This assignment aims to deepen your understanding of OOP in Java through a practical application, illustrating how objects and classes can be used to manage complex data and operations effectively.

2. Simple Bank Account Management System

Objective:
Develop a simple system to manage bank accounts for a small bank, allowing the bank to add accounts, deposit money, withdraw money, and display account details.

Requirements:
BankAccount Class:
Attributes:
accountId: Integer (Unique identifier for the account)
accountHolder: String (Name of the account owner)
balance: Double (Current balance of the account)
Methods:
Constructor to initialize all the attributes.
Getters and setters for all attributes.
deposit(double amount): Method to deposit money into the account.
withdraw(double amount): Method to withdraw money from the account, if sufficient funds are available.
displayAccountDetails(): A method to display the account's details.
Bank Operations:
A static array of BankAccount objects to manage multiple accounts.
A method to add a new account to the bank.
A method to deposit money to a specific account by account ID.
A method to withdraw money from a specific account by account ID.
A method to display details of all bank accounts.
Static Features:
A static counter in the BankAccount class to keep track of the total number of accounts created.
A static method in the BankAccount class that returns the count of total accounts.
Main Class:
Implement a simple text-based user interface using the command line for the bank's operations.
The Main class should utilize the static array and provide functionalities as described in the bank operations section.
Instructions:
BankAccount Class Implementation:
Design and implement the BankAccount class with necessary fields, methods, and a constructor.
Static Features:
Use a static integer to maintain the number of bank accounts.
Implement a static method to get the total number of accounts.
Bank Operations Using Array:
Since no advanced collections are to be used, manage bank accounts using a static array. Ensure to handle the array's size limitations manually.
User Interaction Through Main Class:
Use the Scanner class for input.
Implement a menu system to add accounts, make deposits, withdrawals, and display account information.
Example Usage:

```
public static void main(String[] args)
{
        Scanner scanner = new Scanner(System.in);
        BankAccount[] accounts = new BankAccount[10];  // Array to store up to 10 bank accounts

        boolean exit = false;
        while (!exit) {
            System.out.println("Choose an option:");
            System.out.println("1. Add Account");
            System.out.println("2. Deposit Money");
            System.out.println("3. Withdraw Money");
            System.out.println("4. Display All Accounts");
            System.out.println("5. Exit");
            int choice = scanner.nextInt();
            switch (choice) {
                case 1:
                    // Implementation to add an account
                    break;
                case 2:
                    // Implementation to deposit money
                    break;
                case 3:
                    // Implementation to withdraw money
                    break;
                case 4:
                    // Implementation to display all accounts
                    break;
                case 5:
                    exit = true;
                    break;
                default:
                    System.out.println("Invalid option. Please try again.");
            }
        }
        scanner.close();
    }
```

Deliverables:
Complete Java code implementation for the Bank Account Management System.
Your code should be well-documented, explaining significant sections and decisions.
Write a brief report that outlines how your implementation works and how it uses the core OOP concepts mentioned.