Assignment

Mastering OOPS Basics in Java Youtube Series

Basic Assignment

1. Create a class called Student that stores student name, class, section and marks in 4 subjects. Write constructors, getter/setters, behaviour, and operations to accept and display the data. Also provide operations that return total and percentage. Test the class by creating an implementation function (main).
2. Create a class called Employee that stores its name, department, designation and basic salary. Write constructors, getter/setters, behaviour, and operations to accept and display the data. Also write methods that return his incentive. The incentives are HRA (20%), DA (10%), CA (10%).Test the above code by creating an implementation program.
3. Create a class called Rectangle derived from Point class. Apart from data of Point class, Rectangle should store its width and height. Write constructors, getter/setters, behaviour, and operations to accept and display the data. Also write methods that return its area and perimeter. Test the class by creating an implementation program.
4. Create a class called Employee that stores his code and name. Create two derive classes of Employee named TempEmp (temporary employee) and PerEmp (permanent employee). TempEmp should store wage grade and number of days worked whereas PerEmp should store department, designation and basic salary. Write constructors, getter/setters and appropriate operations.
5. Create a class that counts the number of its object created. If the objects counter it less than 5, it should display a message “Too Less”, when it is equal to 5 “Will do” should be displayed and if more than 5 objects are created it should display “Exceeding the Limits”. Write suitable constructors and methods. Also write a method that returns the number of objects created.

Assignment: Building a Simple Library Management System

This assignment will guide you through implementing a simple library management system using the concepts covered in the "**Mastering OOPS Basics in Java**" series. You will focus on creating and managing individual objects, constructors, inheritance, and using static, final, and abstract features in Java.

1. Classes and Objects

Objective:

Design the foundational classes representing the core entities of the library management system.

Tasks:

- Create the `Book` class:

- Fields:
  - `title` (String)
  - `author` (String)
  - `ISBN` (String)
  - `isBorrowed` (boolean, initially `false`)
- Methods:
  - `getTitle()`: Return the title of the book.
  - `getAuthor()`: Return the author of the book.
  - `getISBN()`: Return the ISBN of the book.
  - `borrowBook()`: Set `isBorrowed` to `true`.
  - `returnBook()`: Set `isBorrowed` to `false`.
  - `isAvailable()`: Return `true` if the book is not borrowed.

- Create the `Member` class:
  - Fields:
    - `name` (String)
    - `memberID` (String)
    - `borrowedBook1`, `borrowedBook2`, `borrowedBook3` (Book): Fields to store up to three borrowed books (you can add more if needed)
  - Methods:
    - `getName()`: Return the member's name.
    - `getMemberID()`: Return the member's ID.
    - `borrowBook(Book book)`: Check if any of the `borrowedBook` fields are `null`, then assign the book to that field and mark it as borrowed.
    - `returnBook(Book book)`: Compare the book with each `borrowedBook` field, if found, set the field to `null` and mark the book as returned.
    - `getBorrowedBooks()`: Display the titles of all books borrowed by the member.

- Create the `Library` class:
  - Fields:
    - `name` (String)
    - `book1`, `book2`, `book3` (Book): Fields to store up to three books in the library (you can add more if needed)
    - `member1`, `member2`, `member3` (Member): Fields to store up to three members in the library (you can add more if needed)
  - Methods:
    - `addBook(Book book)`: Add a book to the first available `book` field.
    - `addMember(Member member)`: Add a member to the first available `member` field.

- `borrowBook(String memberID, String ISBN)`: Allow a member to borrow a book by checking if the book is available and the member exists.
- `returnBook(String memberID, String ISBN)`: Allow a member to return a book.
- `listAvailableBooks()`: Display all books that are currently available.
- `listBorrowedBooks(String memberID)`: Display all books borrowed by a specific member.

2. Constructors

Objective:
Use constructors to initialize objects with specific values.

Tasks:
- `Book` class:
  - Create a constructor that initializes `title`, `author`, and `ISBN`.
  - Ensure `isBorrowed` is set to `false` by default.

- `Member` class:
  - Create a constructor that initializes `name` and `memberID`.
  - Ensure `borrowedBook1`, `borrowedBook2`, and `borrowedBook3` are initialized to `null`.

- `Library` class:
  - Create a constructor that initializes the `name` of the library and sets `book1`, `book2`, `book3`, `member1`, `member2`, and `member3` to `null`.

3. Inheritance

Objective:
Use inheritance to create a parent class and extend it in other classes.

Tasks:
- Create an abstract `Person` class:
  - Fields:
    - `name` (String)
  - Constructor:
    - Initialize `name`.
  - Methods:
    - `getName()`: Return the name of the person.
  - Abstract Method:

- `getDetails()`: An abstract method to be implemented by subclasses to provide details of the person.

- Extend `Person` in the `Member` class:
  - Implement the `getDetails()` method in `Member` to return the member's name and ID.

4. Static Members

Objective:
Use static fields and methods to manage information that is shared across all instances of a class.

Tasks:
- Add static fields to the `Library` class:
  - `totalBooks` (int): Keep track of the total number of books in the library.
  - `totalMembers` (int): Keep track of the total number of registered members.

- Implement static methods in the `Library` class:
  - `incrementBookCount()`: Increase `totalBooks` each time a new book is added.
  - `incrementMemberCount()`: Increase `totalMembers` each time a new member is registered.

- Static Block:
  - Use a static block in the `Library` class to print a welcome message when the class is loaded.

5. Final Members

Objective:
Use the `final` keyword to prevent modification of certain fields and methods.

Tasks:
- Make the following fields `final`:
  - `ISBN` in the `Book` class (since ISBN is unique and should not be changed).
  - `memberID` in the `Member` class (since the ID should not change once assigned).

- Create a `final` method in the `Person` class:
  - `showIdentity()`: A method that prints "I am a person" (this method should not be overridden by subclasses).

6. Abstract Classes and Methods

Objective:
Use abstract classes and methods to define templates that can be extended by other classes.

Tasks:
- Abstract `Person` class (already created in the inheritance section):
  - Ensure that `Person` is an abstract class and contains at least one abstract method (`getDetails()`).

- Implement the abstract method in the `Member` class:
  - Provide specific details about the member, such as their name and ID.

7. Application Implementation

Objective:
Put it all together in a main class to demonstrate the functionality of your library management system.

Tasks:
- Create the `LibraryApp` class (main class):
  - Create an instance of `Library`.
  - Add some books and members to the library.
  - Demonstrate borrowing and returning books.
  - List all available books in the library.
  - List all borrowed books for a member.
  - Display the total number of books and members using static methods.