

**CMPE 202 - SW SEC-48 SYSTEMS ENGR**

**Submitted To** - Prof.Vinodh Gopinath

**Teacher Assistant** - Madhurima Dani

**Submitted By** - Rishikesh Andhare

Assignment - Code Refactoring

A.

The first code contains a class named "BookRental" that consists of private variables such as "bookTitle" and "author," among others. The second code features a class named "MovieRental" that also contains private variables like "movieTitle" and "classification." However, there is some repetition between these two classes, which contradicts the "Don't Repeat Yourself" (DRY) principle. This repetition can be eliminated by creating a common superclass or interface that would allow the "isOverdue()" method to be reused. This superclass or interface would include private variables such as "rentDate," "dueDate," and "rentalFee."

B.

To eliminate the code smell caused by duplicated code between the BookRental and MovieRental classes, we can extract their shared behavior into a common interface or abstract class. Both classes can then implement or extend this common interface or abstract class, enabling them to use the isOverdue() method without duplicating code.

C.

public abstract class Rental {

//bookTitle and movieTitle; i.e both the classes had same variable ‘title’

private String title;

//same goes with dates. Dates due and rent are present in both class.

private Date dueDate;

private Date rentDate;

private double rentalFee;

//the method isOverDue is present in both the classes.

public boolean isOverdue() {

Date now = new Date();

return now.after(dueDate);

}

//The class BookRental and MovieRental are the subclasses of Rental and have unique attributes so we use inheritance.

public class BookRental extends Rental {

private String author;

}

public class MovieRental extends Rental {

private String classification;

}

}