VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

SOFTWARE TESTING (CO3015)

Semester 231

PROJ#1 - CODE REVIEW

Advisor: Bùi Hoài Thắng

Students:

- ✔ Mai Hữu Nghĩa - 2052612
- ✔ Lê Tuấn Hưng - 2052508
- ✔ Nguyễn Ngọc Hưng - 2053075
- ✔ Trần Trí Đạt - 2052443
- ✔ Lê Nguyễn Hoàng Nhân - 2052625

HO CHI MINH CITY, SEPTEMBER 2023

Table of Contents

1. INTRODUCTION

The major project aims to help students apply the knowledge and debugging methods they have learned in chapters 1 and 2, and to use checklists effectively. In addition, it helps students to understand Java and the potential errors that can occur when using this programming language.

2. CONTRIBUTION

Code reviewer	File review
Lê Tuấn Hưng	• Author.java • Book.java • BookDAO.java
Mai Hữu Nghĩa	• User.java • UserDAO.java • BookUpdate.java • Checkout.java
Trần Trí Đạt	• Publisher.java • Sale.java • SalesDAO.java • Login.java • Logout.java
Lê Nguyễn Hoàng Nhân	• Manager.java • Order.java • OrderDAO.java • Profile.java • Register.java
Nguyễn Ngọc Hưng	• BookCategory.java • Cart.java • ModelManager.java • Statistics.java • PasswordEncryptionService.java

3. CODE REVIEW FINDINGS

3.1. Beans

3.1.1. Author.java

Project code: book-me\src\main\java\beans\Author.java

- **Reviewer:** Lê Tuấn Hưng
- **Defects found:**
  - CRITICAL: Missing a constructor for class Author
  - Some variables and methods not following the naming convention.
- **Suggestions:**
  - Creating a constructor for class Author

```
public Author(int isbn, String name) {
    this.isbn = isbn;
    this.name = name;
}
```

  - Following name convention for naming variables and methods

3.1.2. Book.java

Project code: book-me\src\main\java\beans\Book.java

- **Reviewer:** Lê Tuấn Hưng
- **Defects found:**
  - Some variables and methods not following the naming convention.
  - Missing a constructor with parameters allowed, to reduce calling many setMethods()
- **Suggestions:**
  - Replace the defect names that accord with the naming convention.
  - Overload parameter to Book() constructor.

An example of overloading constructor for class Book

```
    public Book(double price, int threshold, int numberOfCopies, String
publicationYear, int numberOfSalesCopies) {
        this.price = price;
        this.threshold = threshold;
        this.numberOfCopies = numberOfCopies;
        this.publicationYear = publicationYear;
        this.numberOfSalesCopies = numberOfSalesCopies;
    }
```

We can add more overload constructor for each cases

3.1.3. BookCategory.java

Project code: book-me\src\main\java\beans\BookCategory.java

- **Reviewer:** Nguyễn Ngọc Hưng
- **Defects found:**
    - The constants inside the enum are not capitalized.
    - The enum BookCategory does not have any comment.
- **Suggestions:**
    - Replace the constant names with capitalized names to follow the naming convention.
    - Add comments for enum BookCategory.

3.1.4. Cart.java

Project code: book-me\src\main\java\beans\Cart.java

- **Reviewer:** Nguyễn Ngọc Hưng
- **Defects found:**

- Some variables and methods not following the naming convention.
- Using incorrect comparison operator.
- Lack of comments for methods and variables.

- **Suggestions:**
  - Replace the defect names with new names that accord with the naming convention.
  - Use the correct comparison operator.
  - Add more appropriate comments for the class, methods, and variables.

3.1.5. Manager.java

Project code: book-me\src\main\java\beans\Manager.java

- **Reviewer:** Lê Nguyễn Hoàng Nhân
- **Defects found:**
  - This file and class don't have any comment that describes the purpose of the file and how to use it.
- **Suggestions:**
  - Add additional comments

3.1.6. Order.java

Project code: book-me\src\main\java\beans\Order.java

- **Reviewer:** Lê Nguyễn Hoàng Nhân
- **Defects found:**
  - This file and class don't have any comment that describes the purpose of the file and how to use it.
  - The layout and format of the file is inconsistent.
  - Some variables do not follow a naming convention and some are not descriptive.

- Variables don’t get initialized properly, which lead to unexpected errors when trying to use uninitialized variables

- **Suggestions:**
  - Add additional comments
  - Trying to follow consistent style in a file and better, across different files. Developers can reference others' styling guides, such as https://google.github.io/styleguide/javaguide.html.
  - Because using uninitialized variables can cause crashes and undefined behavior, developers can use linter to catch these kinds of error and fix them.

3.1.7. Publisher.java

Project code: book-me\src\main\java\beans\Publisher.java

- **Reviewer:** Trần Trí Đạt
- **Defects found:**
  - Class attributes are not properly initialized, hence can have null values.
  - Class has no constructor.
  - Parameters are not checked for non-null before being assigned to attributes.
  - Missing comments for class, attributes, and methods.
  - Methods’ behavior are not expressed in plain language.
- **Suggestions:**
  - Add constructor to initialize attributes

```
    /**
     * Constructor for Publisher and initializes name, addresses and
phones.
     * @param name
     */
    public Publisher(String name) {
```

```
        this.name = name;
        this.addresses = new ArrayList<>();
        this.phones = new ArrayList<>();
    }
```

- Check parameters “addresses” and “phones” before assign to attributes

```
    public void setAddresses(List<String> addresses) {
        if (addresses == null) {
            throw new IllegalArgumentException("Addresses cannot be
null");
        }
        this.addresses = addresses;
    }
    public void setPhones(List<String> phones) {
        if (phones == null) {
            throw new IllegalArgumentException("Phones cannot be null");
        }
        this.phones = phones;
    }
```

- Add comments for class, attributes, and methods.
- Add methods’ behavior details in methods’ comments.

3.1.8. Sale.java

Project code: book-me\src\main\java\beans\Sale.java

- **Reviewer:** Trần Trí Đạt
- **Defects found:**
  - Some attribute and method names do not accord with naming convention.
  - Class attributes are not properly initialized, hence can have null values.

- Parameters are not checked for non-null before being assigned to attributes.
- Parameter "copies" is not checked for non-negative before used
- Class and some methods have no comment.

- **Suggestions:**
  - Fix attribute names to accord with naming convention, such as “ISBN” to “isbn”, “sale_name” to “saleName”, “user_first_name” to “userFirstName”
  - Modify constructors to initialize attributes.

```
    public Sale() {
        this.ISBN = 0;
        this.copies = 0;
        this.price = 0.0;
        this.sale_name = "";
        this.user_first_name = "";
        this.user_last_name = "";
    }
```

  - Check parameter “copies” for non-negative

```
        if (copies < 0) {
            throw new IllegalArgumentException("Number of copy cannot
negative");
        }
```

  - Add comments for class and methods.

3.1.9. User.java

Project code: book-me\src\main\java\beans\User.java

- **Reviewer:** Mai Hữu Nghĩa
- **Defects found:**
  - Some variables and methods not following the naming convention.
  - Some method do not have any comment for it

- In the setPhoneNumber method don't have exception handling for each case of the phone number
- There are 6 methods and 1 constructor that don't have any logic implementation and it are not be called by any other method
- There are too many repeated codes, the creator creates 1 setMethod() for each of the variables in the class.

- **Suggestions:**
  - Replace the defect names with new names that accord with the naming convention.
  - Add some comment for the missing method (in the comment will mention what this method do, how many param it receive and what is this method return)
  - My suggestion is to add 3 more conditions to check if the input phone number is valid or not.

```
if (phoneNumber == null) {
  throw new IllegalArgumentException(""Phone number cannot be null."");
}
if (phoneNumber.length() < MIN_PHONE_NUMBER_LENGTH) {
    throw new IllegalArgumentException(""Phone number is too short."");
}
if (phoneNumber.length() > MAX_PHONE_NUMBER_LENGTH) {
    throw new IllegalArgumentException(""Phone number is too long."");
}
```

  - Remove the unused method.
  - We can improve this by updating the constructor to set all the variable values when calling that class. No need for multi method for multi variable

3.2. Model

3.2.1. BookDAO.java

Project code: book-me\src\main\java\model\BookDAO.java

- **Reviewer:** Lê Tuấn Hưng
- **Defects found:**
    - Some variables should follow the name convention, and be named with the meaningful name
    - Call too many setMethods() for creating a new book.
    - Catch exceptions but sometimes return false, sometimes not.
    - Many findBy'Methods'() have only different at "query" field
- **Suggestions:**
    - Following the name convention, and name the variable in meaningful
    - Adding return false when catch Exception, or comment that Exception may not affect the code flow if not return false
    - Making the request to the "Book.java" file's author to create a constructor with parameters.
    - Condensing many findBy'Methods' into a findByQuery, and name the query added with the meaningful to help to know that is which kind of finding.

```
public static ArrayList<Book> findByQuery(String query,@NotNull String
findData) {

     // Use query string as parameter to find

}
```

3.2.2. ModelManager.java

Project code: book-me\src\main\java\model\ModelManager.java

- **Reviewer:** Nguyễn Ngọc Hưng

- **Defects found:**
    - Some variables should be constants as they are not meant to be changed in the code.
    - Does not have any comments for methods, variables, constants, and attributes.
- **Suggestions:**
    - Change the declarations of those variables to constants (using final).
    - Add comments for every method, variable, constant, and attribute.

3.2.3. OrderDAO.java

Project code: book-me\src\main\java\model\OrderDAO.java

- **Reviewer:** Lê Nguyễn Hoàng Nhân
- **Defects found:**
    - Some variable name are not meaningful (rs)
    - There are not enough comment to help others understand the code
    - In programming we should utilize immutability to reduce the chance of accidentally modifying the value of variables. In this file there are many variables which are read-only after being created and are not immutable.
- **Suggestions:**
    - Change the declarations of those variables to constants (using final).
    - Add comments for every method, variable, constant, and attribute.
    - Change the name of variables to reflect the meaning and purpose of that variable.

3.2.4. SalesDAO.java

Project code: book-me\src\main\java\model\SalesDAO.java

- **Reviewer:** Trần Trí Đạt
- **Defects found:**

- Some variable names do not accord with naming convention.
- Variable "user_id" is not checked before being used, hence it can be null.
- Method "checkout" has no comment.
- Variables have no comment.
- Variable “Double total_sales” should be local variable of method “getTopFiveCustomers”
- Exceptions "SQLException" are handled inappropriately, when catching an exception, the code prints the exception stack trace and then continues execution.

- **Suggestions:**
  - Fix attribute names to accord with naming convention, such as “total_sales” to “totalSales”, “top_five” to “topFive”
  - Check variable “user_id” to make sure that it not null

```
if (user_id == null) {
   throw new NullPointerException("User id parameter cannot be null");
}
```

  - Add comment for method “checkout” to clarify its usage and syntax.

```
/**
 * Checkout the shopping cart
 * @param sales   The sales information
 * @param user_id The user that checking out
 */
```

  - Add comments for variables to clarify its usage.
  - Change position of variable “total_sales” into method “getTopFiveCustomers”
  - Change exception handling for "SQLException" to stop executing if exception found

```
        try {

        } catch (SQLException e) {

            throw "Error";

        }
```

3.2.5. UserDAO.java

Project code: book-me\src\main\java\model\UserDAO.java

- **Reviewer: Mai Hữu Nghĩa**
- **Defects found:**
    - Some variables are being named un-meaningful (such as pw, rs, …)
    - There is too much repeated code, the creator just calls 1 method for 1 variable and so on.
    - The type of the function is not compatible with the param in which input to it. In this case, the `is_manager` has the boolean type, it should not be used with the integer type when calling the getInt method.
    - In the updateUser method, there is a case when the function already returns a value but the SQL statement does not close.

```
                if (pst.executeUpdate() == 1) {
                    return true;
                }
                pst.close();
```

    - Lack of comments for methods and variables.
- **Suggestions:**
    - Change the name to a more meaningful name such as passwordEncryptService, resultSet, prepareStmt.
    - We can update the constructor in the User.java so only need to call the constructor 1 time for all the variables in the user class.

- change the method from setInt() to setBoolean()
- Add a pst.close() before the return statement in the if condition body.
- Add more appropriate comments for the class, methods, and variables.

3.3. Services

3.3.1. PasswordEncryptionService.java

Project code: book-me\src\main\java\services\PasswordEncryptionService.java

- **Reviewer:** Nguyễn Ngọc Hưng
- **Defects found:**
  - Some variables should be constants as they are not being changed in the code.
  - Does not have comments for every class, method, variable, attribute and constant.
- **Suggestions:**
  - Change the declarations of those variables to constants (using final).
  - Add comments for every class, method, variable, attribute and constant.

3.4. Servlets

3.4.1. BookUpdate.java

Project code: book-me\src\main\java\servlets\BookUpdate.java

- **Reviewer: Mai Hữu Nghĩa**
- **Defects found:**
  - Some variables and methods not following the naming convention.
  - There is too much repeated code when the creator wants to check if the author exists or not.
  - The doPost() method is too long, in the method is incharge of updating the book and adding the new book.

- **Suggestions:**
    - Replace the defect names with new names that accord with the naming convention.
    - We can improve it by adding it into a for loop, the creator already initialize a array for us to improve it

```
ArrayList<String> authors = new ArrayList<>();

int maxAuthors = 6; // Maximum number of authors to extract

for (int i = 1; i <= maxAuthors; i++) {

    String authorParam = request.getParameter(""author"" + i);

    if (authorParam != null && !authorParam.isEmpty()) {

        authors.add(authorParam);

    }

}
```

    - We can break down it into handleUpdateBook and handleAddNewBook like below

```
private void handleUpdateBook(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

    beans.Book book = new beans.Book();

    // Extract book information from request parameters and set it in the book
object.

    if (!extractBookInformation(request, book)) {

        // Handle validation errors and return if necessary.

        return;

    }

    // Extract authors and modify the book.

    ArrayList<String> authors = extractAuthors(request, book.getISBN());

    if (authors.isEmpty() && newAuthor.isEmpty()) {

        // Handle validation errors and return if necessary.
```

```
            return;
        }
        // Update the book and handle success or failure.
        if (BookDAO.modifyBook(book, Integer.parseInt(oldISBN), authors)) {
            request.setAttribute(""successMessage"", ""Updated book info
successfully."");
        } else {
            request.setAttribute(""errorMessage"", ""Updating book info failed."");
        }
        // Forward the request to the appropriate JSP page.
        request.getRequestDispatcher(""editBook.jsp"").forward(request, response);
    }
    private void handleAddNewBook(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        beans.Book book = new beans.Book();
        // Extract book information from request parameters and set it in the book
object.
        if (!extractBookInformation(request, book)) {
            // Handle validation errors and return if necessary.
            return;
        }
        // Extract authors and add the new book.
        ArrayList<String> authors = extractAuthors(request, null);
        if (authors.isEmpty()) {
            // Handle validation errors and return if necessary.
            return;
        }
        // Add the new book and handle success or failure.
        if (BookDAO.addNewBook(book)) {
```

```
        request.setAttribute(""successMessage"", ""Added book successfully."");
    } else {
        request.setAttribute(""errorMessage"", ""Adding book failed."");
    }
    // Forward the request to the appropriate JSP page.
    request.getRequestDispatcher(""addNewBook.jsp"").forward(request, response);
}
```

3.4.2. Checkout.java

Project code: book-me\src\main\java\servlets\Checkout.java

- **Reviewer: Mai Hữu Nghĩa**
- **Defects found:**
  - Some variables and methods not following the naming convention.
  - Some comments are not consistent with the context of the code like `// TODO` meaning that the code is still in implementation.
  - Mis-used the `&` and `&&` operator.
- **Suggestions:**
  - Replace the defect names with new names that accord with the naming convention.
  - Remove the comment or finish implementation for that part of the code.
  - Change the `&` to `&&`.

3.4.3. Login.java

Project code: book-me\src\main\java\servlets\Login.java

- **Reviewer:** Trần Trí Đạt
- **Defects found:**
  - Parameter "request" is not checked before being used, hence can have null values
  - Class has no comment

- Method “doPost” has no comment
- Variables have no comment
- Method "doGet" has no use

- **Suggestions:**
  - Check parameter “request” for non-null

```
        if (request == null) {
            throw new NullPointerException("Request parameter cannot be
null");
        }
```

  - Add appropriate comments for class, method, and variables
  - Remove method “doGet”

3.4.4. Logout.java

Project code: book-me\src\main\java\servlets\Logout.java

- **Reviewer:** Trần Trí Đạt
- **Defects found:**
  - Parameter "request" is not checked before being used, hence can have null values
  - Class has no comment
  - Method “doGet” has no comment
- **Suggestions:**
  - Check parameter “request” for non-null

```
        if (request == null) {
            throw new NullPointerException("Request parameter cannot be
null");
        }
```

  - Add appropriate comment for class

```
/**
```

```
 * A class that for Logout

*/
```

- Add appropriate comment for method “doGet”

```
    /**
     * Set the response content for logout and handle logout request
     *
     * @param request  The logout request
     * @param response The response to logout request
     */
```

3.4.5. Profile.java

Project code: book-me\src\main\java\servlets\Profile.java

- **Reviewer:** Lê Nguyễn Hoàng Nhân
- **Defects found:**
  - Some variables don’t follow naming convention, in this case: Camel case
  - Some variables are written in abbreviations, which may cause confusion between developers because it can mean different things to different people.
  - The file does not have the header comment to describe the purpose of the file.
  - The class “Profile” does not have any description.
  - Not all variables are commented
- **Suggestions:**
  - Changes non-standard variables name according to naming conventions
  - Don’t use abbreviations in variable names, use the full name as much as possible.
  - Add more comments to help others understand the code.

3.4.6. Register.java

Project code: book-me\src\main\java\servlets\Profile.java

- **Reviewer:** Lê Nguyễn Hoàng Nhân
- **Defects found:**
  - Some variables are written in abbreviations, which may cause confusion between developers because it can mean different things to different people.
  - The file does not have the header comment to describe the purpose of the file.
  - The class "Register" does not have any description.
  - Not all variables and methods are commented, which can help decrease the learning curve for new hire.
- **Suggestions:**
  - Don't use abbreviations in variable names, use the full name as much as possible.
  - Add more comments to help others understand the code.

3.4.7. Statistics.java

Project code: book-me\src\main\java\servlets\Statistics.java

- **Reviewer:** Nguyễn Ngọc Hưng
- **Defects found:**
  - Lack of comments, there is a comment that does not help understanding the code.
  - The doPost method has too many lines and the doGet method does not have any logic.
  - Some lines of code use the standard indentation inconsistently.
- **Suggestions:**
  - Add more comments to the code and make the comments more understandable.

- Try to break down the doPost to smaller methods. Remove the doGet method if not necessary or write consistent comments if it is intentionally blank.
- Apply the standard indentation correctly for the code.

4. CONCLUSION

In conclusion, our software testing school project has provided us with valuable insights into the world of quality assurance and software testing. Throughout this project, we have learned the importance of meticulous testing to ensure that software products meet the highest standards of functionality, reliability, and usability.