

Course Code : CSE 404

Course Title : Software Engineering and ISD Laboratory

Project name : Bus ticket booking management system

Experiment no: 09

Experiment name: Applying Coding Standard principles to the Java/C# Code

Generated from UML Class Diagram

**Submitted To**

**Dr. Mohammad Zahidur Rahman**

Professor

**Dr. Md. Humayun Kabir**

Professor

Department of Computer Science & Engineering Jahangirnagar University, Savar.

Group No : 02

Group members :

|  |  |  |
| --- | --- | --- |
| **Sl** | Class Roll | Name |
| 01 | 342 | Tama Shil |
| 02 | 370 | Prokash Maitra |
| 03 | 374 | Mubasher adnan jihad |
| 04 | 375 | Pritam Saha |

**Pritam Saha**

**ID:375**

Applying coding standards to Java or C# code generated from a UML class diagram for a bus ticket management system is crucial for ensuring code consistency, readability, maintainability, and collaboration among developers.

Let's provide more detailed guidelines for applying coding standard principles to the Java and C# code generated from a UML class diagram for a bus ticket management system:

Naming Conventions:

1. Classes:

- Use clear, descriptive names for classes, reflecting their purpose.

- Follow Pascal Case for class names.

Example (Java):

class BusTicketManager {

// ...

}

1. Variables and Fields:

-Use meaningful, camelCase names for variables and fields.

-Prefix member variables with "this" (Java) for distinguishing them from local variables.

Example (Java):

private String passengerName;

1. \*\*Methods\*\*:

- Use descriptive names for methods that reflect their actions.

- Follow camelCase for method names.

Example (Java):

public void bookTicket(String passengerName) {// ... }

**Mubasher Adnan Jihad**

**Roll:374**

1. \*\*Constants\*\*:

- Use uppercase letters for constants.

- Separate words with underscores (Java) or use PascalCase (C#) for constant names.

Example (Java):

public static final int MAX\_TICKETS = 10;

**### Code Formatting and Style:**

1. Indentation and Bracing:

- Use consistent indentation (typically 4 spaces or a tab).

-Place opening braces on the same line (Java) or a new line (C#).

- Maintain consistent brace style throughout the codebase.

Example (Java):

if (condition) {

// Code block

} else {

// Code block

}

1. \*\*Comments and Documentation\*\*:

- Add comments to clarify complex logic, especially if it's not immediately obvious.

- Use JavaDoc (Java) or XML comments (C#) to document methods and classes for auto-generating documentation.

Example (Java):

/\*\*

\* This method books a ticket for the given passenger.

\* @ passengerName The name of the passenger.

\*/

public void bookTicket(String passengerName) {

// ...

}

**Prokash Maitra**

**Roll:370**

**### Code Structure and Organization:**

1. \*\*Modularity and Single Responsibility\*\*:

- Follow the Single Responsibility Principle (SRP). Keep classes focused on one task.

- Organize code into logical packages or namespaces.

- Use appropriate access modifiers (public, private, etc.) to control visibility and enforce encapsulation.

1. \*\*Code Reusability\*\*:

- Encapsulate reusable code into functions, methods, or libraries to promote code reuse and maintainability.

- Avoid duplicating code.

**### Error Handling and Exception Handling:**

1. \*\*Error Handling\*\*:

- Implement proper error handling using try-catch blocks.

- Provide meaningful error messages or log exceptions for debugging.

Example (Java):

try {

// Code that may throw an exception

} catch (SomeException e) {

// Handle the exception or log it

}

**Tama Shil**

**ID:342**

1. \*\*Version Control and Collaboration\*\*:

- Use a version control system (e.g., Git) for code management.

- Collaborate with team members through version control repositories.

- Follow branching and merging strategies as needed.

**### Code Analysis and Reviews:**

1. \*\*Static Code Analysis\*\*:

- Utilize static code analysis tools (e.g., Checkstyle for Java, StyleCop for C#) to enforce coding standards automatically.

1. \*\*Code Reviews\*\*:

- Conduct code reviews with team members to ensure adherence to coding standards and identify potential improvements.

By adhering to these coding standard principles and guidelines, you can produce well-structured, readable, and maintainable code for your bus ticket management system, regardless of whether it's generated from a UML class diagram or written manually. Consistency in coding standards helps improve collaboration among developers and reduces the chances of introducing bugs and defects.



Course Code : CSE 404

Course Title : Software Engineering and ISD Laboratory

Project name : Bus ticket booking management system

Experiment no: 09

Experiment name: Applying Coding Standard principles to the Java/C# Code

Generated from UML Class Diagram

**Submitted To**

**Dr. Mohammad Zahidur Rahman**

Professor

**Dr. Md. Humayun Kabir**

Professor

Department of Computer Science & Engineering Jahangirnagar University, Savar.

Group No : 02

Group members :

|  |  |  |
| --- | --- | --- |
| **Sl** | Class Roll | Name |
| 01 | 342 | Tama Shil |
| 02 | 370 | Prokash Maitra |
| 03 | 374 | Mubasher adnan jihad |
| 04 | 375 | Pritam Saha |