**Static Analysis Review of code**



FAST NUCES ISLAMABAD

[Due Date]

**Group Members** :  
Muhammad Awais Rafique 22i-2511

Hassan 22i-2434

Azmar Kashif 22i-2716

**File # 1:**

|  |  |
| --- | --- |
| **Package Name:** | **Com.shashi.service.impl** |
| **File name:** | **UserServiceImpl.java** |
| **Class/ Interface name:** | **UserServiceImpl** |

| **S#** | **Category** | **Checklist Item** | **Yes/No** | **Issue** | **Fix** |
| --- | --- | --- | --- | --- | --- |
| 1 | Naming Conventions | Are class names written in PascalCase? | No | Some class names do not follow PascalCase convention. | Rename class names to PascalCase (e.g., orderService to OrderService). |
|  |  | Are variable and method names written in camelCase? | Yes | Variable and method names are following camelCase convention. | No changes required. |
|  |  | Are constants written in uppercase with underscores? | No | Constants are not written in uppercase with underscores. | Update constants to uppercase with underscores (e.g., MAX\_DISCOUNT for constants). |
| 2 | Code Structure | Are access modifiers used correctly for encapsulation? | No | Access modifiers are missing for some fields and methods, causing potential encapsulation issues. | Add appropriate access modifiers (e.g., private, public, protected). |
|  |  | Are classes and methods properly organized? | Yes | The classes and methods appear well-organized with clear responsibilities. | No changes required. |
| 3 | Method Design | Do methods have a single responsibility? | Yes | Most methods follow the single responsibility principle. | No changes required. |
|  |  | Are method parameters kept to a reasonable limit (preferably fewer than 5)? | Yes | Method parameters are within a reasonable limit, avoiding excessive complexity. | No changes required. |
| 4 | Exception Handling | Are exceptions handled correctly? | No | Some methods may not handle exceptions properly, especially when dealing with external resources or inputs. | Implement appropriate try-catch blocks where necessary, particularly for external resource handling (e.g., file I/O, database calls). |
|  |  | Are specific exceptions used instead of generic Exception? | No | Some generic exceptions (e.g., Exception) are used where specific ones could be more appropriate. | Replace generic exceptions with more specific ones (e.g., IllegalArgumentException, IOException). |
| 5 | Code Readability | Are meaningful and descriptive comments added for complex logic? | No | Complex logic is missing comments to explain what the code is doing. | Add descriptive comments to explain the purpose of complex logic and any non-obvious steps. |
|  |  | Is there consistent indentation (4 spaces or a tab size of 4)? | No | Some parts of the code have inconsistent indentation, making it harder to read. | Ensure consistent 4-space indentation throughout the code. |
|  |  | Are blank lines used appropriately to separate code blocks for better readability? | No | Some logical sections of the code are not separated by blank lines, which affects readability. | Add blank lines between logical code blocks to improve readability. |
| 6 | Performance | Are data structures chosen based on their performance characteristics? | No | Some data structures (like lists or arrays) could be optimized for better performance. | Evaluate and replace data structures with those that are better suited to the task (e.g., use LinkedList instead of ArrayList if frequent insertions/removals are required). |
|  |  | Are costly operations minimized inside loops? | No | Some costly operations (like database calls or I/O operations) are inside loops, which can degrade performance. | Move costly operations outside loops to minimize redundant processing. |
| 7 | Memory Management | Are unnecessary object references set to null to aid garbage collection? | No | Some object references are not cleared, which could lead to memory leaks in certain cases. | Explicitly set unused object references to null to aid garbage collection. |
|  |  | Are large objects or collections handled to avoid memory leaks? | No | Large collections are not efficiently managed, potentially leading to memory issues. | Use appropriate data structures (e.g., WeakHashMap) to handle large collections and avoid memory leaks. |
| 8 | Security | Is user input validated to prevent security issues (e.g., SQL injection, XSS)? | Yes | User input appears to be validated before use in the code. | No changes required. |
| 9 | Maintainability | Are there long methods or deeply nested loops? | No | No long methods or deeply nested loops present. | No changes required. |
|  |  | Is there duplicated code that could be refactored? | No | There does not appear to be duplicate code in the sections provided. | No changes required. |
| 10 | Test Coverage | Are unit tests provided for all public methods and critical functionalities? | No | Unit tests are missing for some public methods. | Add unit tests to cover all public methods and critical functionalities. |
|  |  | Do the unit tests cover all branches and edge cases? | No | Some edge cases or branches may not be covered in the unit tests. | Add unit tests to cover edge cases and all possible branches. |
| 11 | Test Design | Are tests written to follow the AAA pattern (Arrange, Act, Assert)? | No | The test methods do not consistently follow the AAA pattern. | Refactor tests to follow the AAA pattern for better clarity. |
| 12 | Assertions | Are assertions used to verify expected results? | No | Assertions are missing or incomplete in some test cases. | Add assertions to verify the expected outcomes in all relevant test cases. |
| 13 | Mocking and Stubbing | Are mocks or stubs used to isolate the unit under test? | No | Mocking is not used to isolate the unit of work in some tests. | Use mocking frameworks (e.g., Mockito) to isolate external dependencies in unit tests. |
| 14 | Performance Testing | Are there tests to check performance for critical methods? | No | Performance tests are missing for critical methods that may impact the user experience. | Implement performance tests to measure execution times for critical methods. |
| 15 | Test Maintainability | Are test methods well-organized and modular, avoiding duplication of code? | No | Some test methods are not modular and contain duplicated code. | Refactor test methods to reduce duplication and improve organization. |

This checklist can help identify and address potential issues in your code, ensuring that it adheres to best practices for readability, maintainability, and performance.

**File # 2:**

|  |  |
| --- | --- |
| **Package Name:** | **Com.shashi.service.impl** |
| **File name:** | **TrainServiceImpl.java** |
| **Class/ Interface name:** | **TrainServiceImpl** |

| **S#** | **Category** | **Checklist Item** | **Yes/No** | **Issue** | **Fix** |
| --- | --- | --- | --- | --- | --- |
| 1 | Naming Conventions | Are class names written in PascalCase (e.g., OrderService)? | No | Class names are not following PascalCase convention. | Rename class names to follow PascalCase (e.g., orderService to OrderService). |
|  |  | Are variable and method names written in camelCase (e.g., calculateTotalPrice)? | No | Variable and method names are not written in camelCase. | Refactor all variables and methods to use camelCase (e.g., Calculateprice to calculatePrice). |
|  |  | Are constants written in uppercase with underscores (e.g., MAX\_DISCOUNT)? | No | Constants are not written in uppercase with underscores. | Update constants to use uppercase with underscores (e.g., maxDiscount to MAX\_DISCOUNT). |
| 2 | Code Structure | Are access modifiers (public, private, protected) used correctly according to Java standards? | No | Missing or incorrect use of access modifiers, leading to poor encapsulation. | Add appropriate access modifiers to all class members (e.g., use private for fields and public for methods). |
|  |  | Are classes and interfaces clearly separated (e.g., no mixed responsibilities)? | No | Classes have mixed responsibilities or unclear separation. | Refactor classes to ensure each class and interface has a single responsibility. |
|  |  | Are packages used appropriately to organize related classes (e.g., com.pos.services)? | No | Related classes are not grouped into meaningful packages. | Organize classes into packages based on functionality (e.g., com.project.services for service classes). |
| 3 | Method Design | Do methods have a single responsibility (i.e., they perform only one task)? | No | Some methods are performing multiple tasks. | Break down complex methods into smaller ones with single responsibilities. |
|  |  | Are method parameters kept to a reasonable limit (preferably fewer than 5)? | No | Methods have too many parameters, making them difficult to use and test. | Refactor methods to reduce parameters by grouping related ones into objects or using builder patterns. |
|  |  | Is method overloading used properly and not abused? | No | Overloaded methods are inconsistent or excessive, leading to confusion. | Simplify method overloading by ensuring clarity and avoiding redundant variations. |
| 4 | Exception Handling | Are exceptions handled correctly with try-catch blocks where needed? | No | Some exceptions are unhandled or handled incorrectly. | Add appropriate try-catch blocks to handle exceptions gracefully. |
|  |  | Are specific exceptions used instead of generic Exception or Throwable? | No | Generic exceptions (e.g., Exception, Throwable) are being used. | Replace generic exceptions with specific ones (e.g., IllegalArgumentException, IOException). |
|  |  | Is logging implemented within catch blocks for debugging purposes? | No | Missing logging in exception handling blocks. | Implement logging inside catch blocks to capture useful debug information. |
| 5 | Code Readability | Are meaningful and descriptive comments added for complex logic? | No | Complex logic lacks meaningful comments. | Add detailed comments to explain complex logic and algorithms. |
|  |  | Is there consistent indentation (4 spaces or a tab size of 4)? | No | Inconsistent indentation makes the code hard to read. | Reformat the code to use consistent 4-space indentation. |
|  |  | Are blank lines used appropriately to separate code blocks for better readability? | No | Lack of blank lines between logical sections of code. | Add blank lines between logical sections to enhance readability. |
|  |  | Is the code easy to read and understand? | No | Code readability is poor due to inconsistent naming and lack of formatting. | Improve naming conventions and formatting for better readability. |
|  |  | Are meaningful names used for variables, classes, and methods? | No | Some variable, class, and method names are non-descriptive or ambiguous. | Rename variables, classes, and methods to use meaningful and descriptive names. |
| 6 | Performance | Are data structures chosen based on their performance characteristics (e.g., ArrayList vs LinkedList)? | No | Suboptimal data structures are used in certain scenarios. | Evaluate and replace data structures with ones optimized for the specific use case. |
|  |  | Are costly operations (e.g., database calls) minimized inside loops? | No | Database calls are performed inside loops, impacting performance. | Move database calls outside loops and batch them where possible. |
|  |  | Is lazy initialization used to defer object creation until it is needed? | No | Objects are initialized unnecessarily, increasing memory usage. | Use lazy initialization to create objects only when required. |
|  |  | Are there redundant calculations or excessive logging? | No | Redundant calculations and logging are slowing down execution. | Eliminate redundant calculations and optimize logging statements. |
| 7 | Memory Management | Are unnecessary object references set to null to aid garbage collection? | No | Unused object references are not cleared, potentially causing memory leaks. | Set unnecessary object references to null to assist garbage collection. |
|  |  | Are large objects or collections properly handled to avoid memory leaks (e.g., using WeakHashMap)? | No | Large collections are not managed efficiently. | Use appropriate structures like WeakHashMap to manage large collections. |
|  |  | Are external resources like files, streams, and database connections properly closed to prevent leaks? | No | External resources are not consistently closed after use. | Ensure proper closure of resources using try-with-resources or explicit close calls. |
|  |  | Is the try-with-resources statement used for automatic resource management? | No | Manual resource management is used instead of try-with-resources. | Use try-with-resources for automatic resource management. |
|  |  | Are large files processed efficiently to avoid loading the entire file into memory? | No | Large files are being loaded entirely into memory. | Use streaming or buffered processing to handle large files. |
| 8 | Security | Is user input validated to prevent security issues (e.g., SQL injection, XSS)? | No | Lack of input validation exposes the system to security vulnerabilities. | Validate all user input using input sanitization and parameterized queries. |
|  |  | Are sensitive data (e.g., passwords) encrypted or hashed before storage? | No | Sensitive data is stored in plaintext. | Use encryption or hashing (e.g., bcrypt) to securely store sensitive data. |
| 9 | Maintainability | Are there long methods or deeply nested loops? | No | Some methods are overly long or have deeply nested loops, making them hard to maintain. | Refactor long methods into smaller ones and simplify deeply nested loops. |
|  |  | Is there duplicated code that could be refactored? | No | Duplicate code is present in multiple locations. | Refactor duplicate code into reusable methods or classes. |
|  |  | Are there any magic numbers or hard-coded values? | No | Hard-coded values are present in the code. | Replace hard-coded values with constants or configuration properties. |
| 10 | Test Coverage | Are unit tests provided for all public methods and critical functionalities? | No | Unit tests are missing for some public methods. | Write unit tests for all public methods and critical functionalities. |
|  |  | Do the unit tests cover all branches and edge cases? | No | Tests are not comprehensive, missing edge cases and branches. | Add test cases to cover all possible branches and edge cases. |
|  |  | Is code coverage measured and maintained at an acceptable level? | No | Code coverage is below the acceptable threshold (e.g., 80%). | Increase unit test coverage and measure it with a tool like JaCoCo. |
| 11 | Test Design | Are tests written to follow the AAA pattern (Arrange, Act, Assert)? | No | Test methods do not consistently follow the AAA pattern. | Rewrite test methods to follow the AAA pattern for better clarity. |
|  |  | Are individual test cases independent of each other? | No | Test cases are interdependent, causing flaky tests. | Refactor tests to ensure independence, avoiding shared states. |
|  |  | Are meaningful and descriptive names used for test methods? | No | Test method names are ambiguous or non-descriptive. | Rename test methods to reflect their purpose (e.g., testCalculatePrice\_WithValidInput). |
| 12 | Assertions | Are assertions used to verify expected results (assertEquals, assertTrue, assertThrows)? | No | Assertions are missing or incorrect in some test cases. | Add proper assertions to verify expected results in all test cases. |
|  |  | Are specific assertions used instead of general ones? | No | General assertions are used where specific ones are more appropriate. | Use specific assertions like assertArrayEquals for arrays. |
| 13 | Boundary and Edge Cases | Are edge cases and boundary conditions tested? | No | Edge cases and boundary conditions are not thoroughly tested. | Add tests to cover edge cases (e.g., empty input, null values) and boundary conditions (e.g., maximum/minimum values). |
| 14 | Mocking and Stubbing | Are mocks or stubs used to isolate the unit under test? | No | Dependencies are not mocked, leading to integration issues in unit tests. | Use mocking frameworks like Mockito to isolate dependencies. |
| 15 | Performance Testing | Are there tests to check performance for critical methods? | No | Performance tests for critical methods are missing. | Implement performance tests to measure execution time and response speed for critical methods. |
| 16 | Test Maintainability | Are test methods well-organized and modular, avoiding duplication of code? | No | Test methods have duplicate code and lack organization. | Refactor test methods to remove duplication and organize them into logical groups. |
|  |  | Is there a setup method (@BeforeEach) for initializing common objects and resources? | No | Common objects and resources are initialized in each test manually. | Add a @BeforeEach method to initialize common objects and resources. |
| 17 | Code Smells | Are there any code smells not covered by the checklist? | No | Code smells like long methods and lack of comments are present. | Refactor to remove code smells, ensuring shorter methods, better comments, and meaningful naming. |
| 18 | Coding Standards | Are there any violations of coding standards not covered by the checklist? | No | Coding standard violations like improper spacing and inconsistent naming are found. | Review and fix coding standard violations by adhering to industry best practices. |
| 19 | Performance Inefficiencies | Are there any performance inefficiencies not covered by the checklist? | No | Inefficiencies like redundant logging and unnecessary synchronization are present. | Optimize performance by reducing redundant logging and reviewing synchronization mechanisms. |

**File # 3:**

|  |  |
| --- | --- |
| **Package Name:** | **Com.shashi.service.impl** |
| **File name:** | **BookingServiceImpl.java** |
| **Class/ Interface name:** | **BookingServiceImpl** |

| **S#** | **Category** | **Checklist Item** | **Yes/No** | **Issue** | **Fix** |
| --- | --- | --- | --- | --- | --- |
| 1 | Naming Conventions | Are class names written in PascalCase (e.g., OrderService)? | Yes | - | - |
| 1 | Naming Conventions | Are variable and method names written in camelCase (e.g., calculateTotalPrice)? | Yes | - | - |
| 1 | Naming Conventions | Are constants written in uppercase with underscores (e.g., MAX\_DISCOUNT)? | N/A | No constants in the provided code | - |
| 2 | Code Structure | Are access modifiers (public, private, protected) used correctly according to Java standards? | Yes | - | - |
| 2 | Code Structure | Are classes and interfaces clearly separated (e.g., no mixed responsibilities)? | Yes | - | - |
| 2 | Code Structure | Are packages used appropriately to organize related classes (e.g., com.pos.services)? | Yes | - | - |
| 3 | Method Design | Do methods have a single responsibility (i.e., they perform only one task)? | Yes | - | - |
| 3 | Method Design | Are method parameters kept to a reasonable limit (preferably fewer than 5)? | Yes | - | - |
| 3 | Method Design | Is method overloading used properly and not abused? | N/A | No method overloading in the provided code | - |
| 4 | Exception Handling | Are exceptions handled correctly with try-catch blocks where needed? | Yes | - | - |
| 4 | Exception Handling | Are specific exceptions used instead of generic Exception or Throwable? | No | SQLException is caught and rethrown as TrainException | Use more specific exception handling |
| 4 | Exception Handling | Is logging implemented within catch blocks for debugging purposes? | No | System.out.println is used instead of proper logging | Implement proper logging (e.g., SLF4J) |
| 5 | Code Readability | Are meaningful and descriptive comments added for complex logic? | Partially | Some comments present, but could be improved | Add more detailed comments for complex logic |
| 5 | Code Readability | Is there consistent indentation (4 spaces or a tab size of 4)? | Yes | - | - |
| 5 | Code Readability | Are blank lines used appropriately to separate code blocks for better readability? | Yes | - | - |
| 5 | Code Readability | Is the code easy to read and understand? | Yes | - | - |
| 5 | Code Readability | Are meaningful names used for variables, classes, and methods? | Yes | - | - |
| 6 | Performance | Are data structures chosen based on their performance characteristics (e.g., ArrayList vs LinkedList)? | Yes | ArrayList is used appropriately | - |
| 6 | Performance | Are costly operations (e.g., database calls) minimized inside loops? | Yes | - | - |
| 6 | Performance | Is lazy initialization used to defer object creation until it is needed? | N/A | No applicable scenarios in the provided code | - |
| 6 | Performance | Are there redundant calculations or excessive logging? | No | - | - |
| 7 | Memory Management | Are unnecessary object references set to null to aid garbage collection? | No | References could be set to null after use | Set references to null after use |
| 7 | Memory Management | Are large objects or collections properly handled to avoid memory leaks (e.g., using WeakHashMap)? | N/A | No large objects or collections in the provided code | - |
| 7 | Memory Management | Are external resources like files, streams, and database connections properly closed to prevent resource leaks? | Partially | PreparedStatement is closed, but Connection is not | Close Connection in finally block |
| 7 | Memory Management | Is the try-with-resources statement used for automatic resource management (e.g., FileInputStream, BufferedReader)? | No | try-with-resources not used | Implement try-with-resources for Connection and PreparedStatement |
| 7 | Memory Management | Are large files processed efficiently (e.g., using streams or buffered reading) to avoid loading the entire file into memory? | N/A | No file processing in the provided code | - |
| 7 | Memory Management | Is RandomAccessFile used appropriately for large files requiring specific access? | N/A | No file access in the provided code | - |
| 7 | Memory Management | Is error handling implemented to safely close resources in case of exceptions? | No | Resources not closed in catch block | Implement proper resource closing in finally block |
| 8 | Security | Is user input validated to prevent security issues (e.g., SQL injection, XSS)? | No | Prepared statements used, but no input validation | Implement input validation |
| 8 | Security | Are sensitive data (e.g., passwords) encrypted or hashed before storage? | N/A | No sensitive data handling in the provided code | - |
| 9 | Maintainability | Are there long methods or deeply nested loops? | No | - | - |
| 9 | Maintainability | Is there duplicated code that could be refactored? | No | - | - |
| 9 | Maintainability | Are there any magic numbers or hard-coded values? | No | - | - |
| 10 | Code Smells | Are there any code smells not covered by the checklist that are present in the code? | Yes | Use of System.out.println for error logging | Replace with proper logging framework |
| 11 | Coding Standards | Are there any violations of coding standards not covered by the checklist that are present in the code? | No | - | - |
| 12 | Performance Inefficiencies | Are there any performance inefficiencies not covered by the checklist that are present in the code? | No | - | - |

**File # 4:**

|  |  |
| --- | --- |
| **Package Name:** | **Com.shashi.constant** |
| **File name:** | **ResponseCode.java** |
| **Class/ Interface name:** | **ResponseCode** |

| **S#** | **Category** | **Checklist Item** | **Yes/No** | **Issue** | **Fix** |
| --- | --- | --- | --- | --- | --- |
| 1 | Naming Conventions | Are class names written in PascalCase (e.g., OrderService)? | Yes | - | - |
| 1 | Naming Conventions | Are variable and method names written in camelCase (e.g., calculateTotalPrice)? | Yes | - | - |
| 1 | Naming Conventions | Are constants written in uppercase with underscores (e.g., MAX\_DISCOUNT)? | Yes | - | - |
| 2 | Code Structure | Are access modifiers (public, private, protected) used correctly according to Java standards? | Yes | - | - |
| 2 | Code Structure | Are classes and interfaces clearly separated (e.g., no mixed responsibilities)? | Yes | - | - |
| 2 | Code Structure | Are packages used appropriately to organize related classes (e.g., com.pos.services)? | Yes | - | - |
| 3 | Method Design | Do methods have a single responsibility (i.e., they perform only one task)? | Yes | - | - |
| 3 | Method Design | Are method parameters kept to a reasonable limit (preferably fewer than 5)? | Yes | - | - |
| 3 | Method Design | Is method overloading used properly and not abused? | N/A | No method overloading in the provided code | - |
| 4 | Exception Handling | Are exceptions handled correctly with try-catch blocks where needed? | N/A | No exception handling required in this enum | - |
| 4 | Exception Handling | Are specific exceptions used instead of generic Exception or Throwable? | N/A | No exceptions used in this enum | - |
| 4 | Exception Handling | Is logging implemented within catch blocks for debugging purposes? | N/A | No exception handling in this enum | - |
| 5 | Code Readability | Are meaningful and descriptive comments added for complex logic? | No | No comments explaining the purpose of the enum or its usage | Add class-level JavaDoc comments |
| 5 | Code Readability | Is there consistent indentation (4 spaces or a tab size of 4)? | Yes | - | - |
| 5 | Code Readability | Are blank lines used appropriately to separate code blocks for better readability? | Yes | - | - |
| 5 | Code Readability | Is the code easy to read and understand? | Yes | - | - |
| 5 | Code Readability | Are meaningful names used for variables, classes, and methods? | Yes | - | - |
| 6 | Performance | Are data structures chosen based on their performance characteristics (e.g., ArrayList vs LinkedList)? | Yes | Arrays.stream is used appropriately | - |
| 6 | Performance | Are costly operations (e.g., database calls) minimized inside loops? | N/A | No loops or database calls in this enum | - |
| 6 | Performance | Is lazy initialization used to defer object creation until it is needed? | N/A | Not applicable for enum | - |
| 6 | Performance | Are there redundant calculations or excessive logging? | No | - | - |
| 7 | Memory Management | Are unnecessary object references set to null to aid garbage collection? | N/A | Not applicable for enum | - |
| 7 | Memory Management | Are large objects or collections properly handled to avoid memory leaks (e.g., using WeakHashMap)? | N/A | No large objects or collections in this enum | - |
| 7 | Memory Management | Are external resources like files, streams, and database connections properly closed to prevent resource leaks? | N/A | No external resources used in this enum | - |
| 7 | Memory Management | Is the try-with-resources statement used for automatic resource management (e.g., FileInputStream, BufferedReader)? | N/A | No resources to manage in this enum | - |
| 7 | Memory Management | Are large files processed efficiently (e.g., using streams or buffered reading) to avoid loading the entire file into memory? | N/A | No file processing in this enum | - |
| 7 | Memory Management | Is RandomAccessFile used appropriately for large files requiring specific access? | N/A | No file access in this enum | - |
| 7 | Memory Management | Is error handling implemented to safely close resources in case of exceptions? | N/A | No resources to close in this enum | - |
| 8 | Security | Is user input validated to prevent security issues (e.g., SQL injection, XSS)? | N/A | No user input handling in this enum | - |
| 8 | Security | Are sensitive data (e.g., passwords) encrypted or hashed before storage? | N/A | No sensitive data in this enum | - |
| 9 | Maintainability | Are there long methods or deeply nested loops? | No | - | - |
| 9 | Maintainability | Is there duplicated code that could be refactored? | No | - | - |
| 9 | Maintainability | Are there any magic numbers or hard-coded values? | No | All values are properly defined as enum constants | - |
| 10 | Code Smells | Are there any code smells not covered by the checklist that are present in the code? | No | - | - |
| 11 | Coding Standards | Are there any violations of coding standards not covered by the checklist that are present in the code? | No | - | - |
| 12 | Performance Inefficiencies | Are there any performance inefficiencies not covered by the checklist that are present in the code? | No | - | - |

**File # 5:**

|  |  |
| --- | --- |
| **Package Name:** | **Com.shashi.constant** |
| **File name:** | **UserRole.java** |
| **Class/ Interface name:** | **UserRole** |

| **S#** | **Category** | **Checklist Item** | **Yes/No** | **Issue** | **Fix** |
| --- | --- | --- | --- | --- | --- |
| 1 | Naming Conventions | Are class names written in PascalCase (e.g., OrderService)? | Yes | - | - |
| 1 | Naming Conventions | Are variable and method names written in camelCase (e.g., calculateTotalPrice)? | N/A | No variables or methods in this enum | - |
| 1 | Naming Conventions | Are constants written in uppercase with underscores (e.g., MAX\_DISCOUNT)? | Yes | Enum constants are in uppercase | - |
| 2 | Code Structure | Are access modifiers (public, private, protected) used correctly according to Java standards? | Yes | - | - |
| 2 | Code Structure | Are classes and interfaces clearly separated (e.g., no mixed responsibilities)? | Yes | - | - |
| 2 | Code Structure | Are packages used appropriately to organize related classes (e.g., com.pos.services)? | Yes | - | - |
| 3 | Method Design | Do methods have a single responsibility (i.e., they perform only one task)? | N/A | No methods in this enum | - |
| 3 | Method Design | Are method parameters kept to a reasonable limit (preferably fewer than 5)? | N/A | No methods in this enum | - |
| 3 | Method Design | Is method overloading used properly and not abused? | N/A | No methods in this enum | - |
| 4 | Exception Handling | Are exceptions handled correctly with try-catch blocks where needed? | N/A | No exception handling required in this enum | - |
| 4 | Exception Handling | Are specific exceptions used instead of generic Exception or Throwable? | N/A | No exceptions used in this enum | - |
| 4 | Exception Handling | Is logging implemented within catch blocks for debugging purposes? | N/A | No exception handling in this enum | - |
| 5 | Code Readability | Are meaningful and descriptive comments added for complex logic? | No | No comments explaining the purpose of the enum | Add class-level JavaDoc comments |
| 5 | Code Readability | Is there consistent indentation (4 spaces or a tab size of 4)? | Yes | - | - |
| 5 | Code Readability | Are blank lines used appropriately to separate code blocks for better readability? | N/A | Single-line enum, no need for blank lines | - |
| 5 | Code Readability | Is the code easy to read and understand? | Yes | - | - |
| 5 | Code Readability | Are meaningful names used for variables, classes, and methods? | Yes | - | - |
| 6 | Performance | Are data structures chosen based on their performance characteristics (e.g., ArrayList vs LinkedList)? | N/A | No data structures used in this enum | - |
| 6 | Performance | Are costly operations (e.g., database calls) minimized inside loops? | N/A | No operations or loops in this enum | - |
| 6 | Performance | Is lazy initialization used to defer object creation until it is needed? | N/A | Not applicable for enum | - |
| 6 | Performance | Are there redundant calculations or excessive logging? | No | - | - |
| 7 | Memory Management | Are unnecessary object references set to null to aid garbage collection? | N/A | Not applicable for enum | - |
| 7 | Memory Management | Are large objects or collections properly handled to avoid memory leaks (e.g., using WeakHashMap)? | N/A | No objects or collections in this enum | - |
| 7 | Memory Management | Are external resources like files, streams, and database connections properly closed to prevent resource leaks? | N/A | No external resources used in this enum | - |
| 7 | Memory Management | Is the try-with-resources statement used for automatic resource management (e.g., FileInputStream, BufferedReader)? | N/A | No resources to manage in this enum | - |
| 7 | Memory Management | Are large files processed efficiently (e.g., using streams or buffered reading) to avoid loading the entire file into memory? | N/A | No file processing in this enum | - |
| 7 | Memory Management | Is RandomAccessFile used appropriately for large files requiring specific access? | N/A | No file access in this enum | - |
| 7 | Memory Management | Is error handling implemented to safely close resources in case of exceptions? | N/A | No resources to close in this enum | - |
| 8 | Security | Is user input validated to prevent security issues (e.g., SQL injection, XSS)? | N/A | No user input handling in this enum | - |
| 8 | Security | Are sensitive data (e.g., passwords) encrypted or hashed before storage? | N/A | No sensitive data in this enum | - |
| 9 | Maintainability | Are there long methods or deeply nested loops? | No | - | - |
| 9 | Maintainability | Is there duplicated code that could be refactored? | No | - | - |
| 9 | Maintainability | Are there any magic numbers or hard-coded values? | No | All values are properly defined as enum constants | - |
| 10 | Code Smells | Are there any code smells not covered by the checklist that are present in the code? | No | - | - |
| 11 | Coding Standards | Are there any violations of coding standards not covered by the checklist that are present in the code? | No | - | - |
| 12 | Performance Inefficiencies | Are there any performance inefficiencies not covered by the checklist that are present in the code? | No | - | - |

**File # 6:**

|  |  |
| --- | --- |
| **Package Name:** | **Com.shashi.constant** |
| **File name:** | **BookingService.java** |
| **Class/ Interface name:** | **BookingService** |

| **S#** | **Category** | **Checklist Item** | **Yes/No** | **Issue** | **Fix** |
| --- | --- | --- | --- | --- | --- |
| 1 | Naming Conventions | Are class names written in PascalCase (e.g., OrderService)? | Yes | - | - |
| 1 | Naming Conventions | Are variable and method names written in camelCase (e.g., calculateTotalPrice)? | Yes | - | - |
| 1 | Naming Conventions | Are constants written in uppercase with underscores (e.g., MAX\_DISCOUNT)? | N/A | No constants in this interface | - |
| 2 | Code Structure | Are access modifiers (public, private, protected) used correctly according to Java standards? | Yes | - | - |
| 2 | Code Structure | Are classes and interfaces clearly separated (e.g., no mixed responsibilities)? | Yes | - | - |
| 2 | Code Structure | Are packages used appropriately to organize related classes (e.g., com.pos.services)? | Yes | - | - |
| 3 | Method Design | Do methods have a single responsibility (i.e., they perform only one task)? | Yes | - | - |
| 3 | Method Design | Are method parameters kept to a reasonable limit (preferably fewer than 5)? | Yes | - | - |
| 3 | Method Design | Is method overloading used properly and not abused? | N/A | No method overloading in this interface | - |
| 4 | Exception Handling | Are exceptions handled correctly with try-catch blocks where needed? | N/A | Exception handling is delegated to implementing classes | - |
| 4 | Exception Handling | Are specific exceptions used instead of generic Exception or Throwable? | Yes | TrainException is used | - |
| 4 | Exception Handling | Is logging implemented within catch blocks for debugging purposes? | N/A | No exception handling in this interface | - |
| 5 | Code Readability | Are meaningful and descriptive comments added for complex logic? | No | No comments explaining the purpose of the interface or methods | Add JavaDoc comments |
| 5 | Code Readability | Is there consistent indentation (4 spaces or a tab size of 4)? | Yes | - | - |
| 5 | Code Readability | Are blank lines used appropriately to separate code blocks for better readability? | Yes | - | - |
| 5 | Code Readability | Is the code easy to read and understand? | Yes | - | - |
| 5 | Code Readability | Are meaningful names used for variables, classes, and methods? | Yes | - | - |
| 6 | Performance | Are data structures chosen based on their performance characteristics (e.g., ArrayList vs LinkedList)? | N/A | No specific data structure implementation in this interface | - |
| 6 | Performance | Are costly operations (e.g., database calls) minimized inside loops? | N/A | No implementation details in this interface | - |
| 6 | Performance | Is lazy initialization used to defer object creation until it is needed? | N/A | No object creation in this interface | - |
| 6 | Performance | Are there redundant calculations or excessive logging? | N/A | No calculations or logging in this interface | - |
| 7 | Memory Management | Are unnecessary object references set to null to aid garbage collection? | N/A | No object references in this interface | - |
| 7 | Memory Management | Are large objects or collections properly handled to avoid memory leaks (e.g., using WeakHashMap)? | N/A | No object handling in this interface | - |
| 7 | Memory Management | Are external resources like files, streams, and database connections properly closed to prevent resource leaks? | N/A | No resource handling in this interface | - |
| 7 | Memory Management | Is the try-with-resources statement used for automatic resource management (e.g., FileInputStream, BufferedReader)? | N/A | No resource management in this interface | - |
| 7 | Memory Management | Are large files processed efficiently (e.g., using streams or buffered reading) to avoid loading the entire file into memory? | N/A | No file processing in this interface | - |
| 7 | Memory Management | Is RandomAccessFile used appropriately for large files requiring specific access? | N/A | No file access in this interface | - |
| 7 | Memory Management | Is error handling implemented to safely close resources in case of exceptions? | N/A | No resource handling in this interface | - |
| 8 | Security | Is user input validated to prevent security issues (e.g., SQL injection, XSS)? | N/A | No input handling in this interface | - |
| 8 | Security | Are sensitive data (e.g., passwords) encrypted or hashed before storage? | N/A | No data storage in this interface | - |
| 9 | Maintainability | Are there long methods or deeply nested loops? | No | - | - |
| 9 | Maintainability | Is there duplicated code that could be refactored? | No | - | - |
| 9 | Maintainability | Are there any magic numbers or hard-coded values? | No | - | - |
| 10 | Code Smells | Are there any code smells not covered by the checklist that are present in the code? | No | - | - |
| 11 | Coding Standards | Are there any violations of coding standards not covered by the checklist that are present in the code? | No | - | - |
| 12 | Performance Inefficiencies | Are there any performance inefficiencies not covered by the checklist that are present in the code? | No | - | - |

**File # 7:**

|  |  |
| --- | --- |
| **Package Name:** | **Com.shashi.constant** |
| **File name:** | **TrainService.java** |
| **Class/ Interface name:** | **TrainService** |

| **S#** | **Category** | **Checklist Item** | **Yes/No** | **Issue** | **Fix** |
| --- | --- | --- | --- | --- | --- |
| 1 | Naming Conventions | Are class names written in PascalCase (e.g., OrderService)? | Yes | - | - |
| 1 | Naming Conventions | Are variable and method names written in camelCase (e.g., calculateTotalPrice)? | Yes | - | - |
| 1 | Naming Conventions | Are constants written in uppercase with underscores (e.g., MAX\_DISCOUNT)? | N/A | No constants in this interface | - |
| 2 | Code Structure | Are access modifiers (public, private, protected) used correctly according to Java standards? | Yes | - | - |
| 2 | Code Structure | Are classes and interfaces clearly separated (e.g., no mixed responsibilities)? | Yes | - | - |
| 2 | Code Structure | Are packages used appropriately to organize related classes (e.g., com.pos.services)? | Yes | - | - |
| 3 | Method Design | Do methods have a single responsibility (i.e., they perform only one task)? | Yes | - | - |
| 3 | Method Design | Are method parameters kept to a reasonable limit (preferably fewer than 5)? | Yes | - | - |
| 3 | Method Design | Is method overloading used properly and not abused? | N/A | No method overloading in this interface | - |
| 4 | Exception Handling | Are exceptions handled correctly with try-catch blocks where needed? | N/A | Exception handling is delegated to implementing classes | - |
| 4 | Exception Handling | Are specific exceptions used instead of generic Exception or Throwable? | Yes | TrainException is used | - |
| 4 | Exception Handling | Is logging implemented within catch blocks for debugging purposes? | N/A | No exception handling in this interface | - |
| 5 | Code Readability | Are meaningful and descriptive comments added for complex logic? | No | No comments explaining the purpose of the interface or methods | Add JavaDoc comments |
| 5 | Code Readability | Is there consistent indentation (4 spaces or a tab size of 4)? | Yes | - | - |
| 5 | Code Readability | Are blank lines used appropriately to separate code blocks for better readability? | Yes | - | - |
| 5 | Code Readability | Is the code easy to read and understand? | Yes | - | - |
| 5 | Code Readability | Are meaningful names used for variables, classes, and methods? | Yes | - | - |
| 6 | Performance | Are data structures chosen based on their performance characteristics (e.g., ArrayList vs LinkedList)? | N/A | No specific data structure implementation in this interface | - |
| 6 | Performance | Are costly operations (e.g., database calls) minimized inside loops? | N/A | No implementation details in this interface | - |
| 6 | Performance | Is lazy initialization used to defer object creation until it is needed? | N/A | No object creation in this interface | - |
| 6 | Performance | Are there redundant calculations or excessive logging? | N/A | No calculations or logging in this interface | - |
| 7 | Memory Management | Are unnecessary object references set to null to aid garbage collection? | N/A | No object references in this interface | - |
| 7 | Memory Management | Are large objects or collections properly handled to avoid memory leaks (e.g., using WeakHashMap)? | N/A | No object handling in this interface | - |
| 7 | Memory Management | Are external resources like files, streams, and database connections properly closed to prevent resource leaks? | N/A | No resource handling in this interface | - |
| 7 | Memory Management | Is the try-with-resources statement used for automatic resource management (e.g., FileInputStream, BufferedReader)? | N/A | No resource management in this interface | - |
| 7 | Memory Management | Are large files processed efficiently (e.g., using streams or buffered reading) to avoid loading the entire file into memory? | N/A | No file processing in this interface | - |
| 7 | Memory Management | Is RandomAccessFile used appropriately for large files requiring specific access? | N/A | No file access in this interface | - |
| 7 | Memory Management | Is error handling implemented to safely close resources in case of exceptions? | N/A | No resource handling in this interface | - |
| 8 | Security | Is user input validated to prevent security issues (e.g., SQL injection, XSS)? | N/A | No input handling in this interface | - |
| 8 | Security | Are sensitive data (e.g., passwords) encrypted or hashed before storage? | N/A | No data storage in this interface | - |
| 9 | Maintainability | Are there long methods or deeply nested loops? | No | - | - |
| 9 | Maintainability | Is there duplicated code that could be refactored? | No | - | - |
| 9 | Maintainability | Are there any magic numbers or hard-coded values? | No | - | - |
| 10 | Code Smells | Are there any code smells not covered by the checklist that are present in the code? | No | - | - |
| 11 | Coding Standards | Are there any violations of coding standards not covered by the checklist that are present in the code? | No | - | - |
| 12 | Performance Inefficiencies | Are there any performance inefficiencies not covered by the checklist that are present in the code? | No | - | - |

**File # 8:**

|  |  |
| --- | --- |
| **Package Name:** | **Com.shashi.constant** |
| **File name:** | **TrainService.java** |
| **Class/ Interface name:** | **TrainService** |

| **S#** | **Category** | **Checklist Item** | **Yes/No** | **Issue** | **Fix** |
| --- | --- | --- | --- | --- | --- |
| 1 | Naming Conventions | Are class names written in PascalCase (e.g., OrderService)? | Yes | - | - |
| 1 | Naming Conventions | Are variable and method names written in camelCase (e.g., calculateTotalPrice)? | Yes | - | - |
| 1 | Naming Conventions | Are constants written in uppercase with underscores (e.g., MAX\_DISCOUNT)? | N/A | No constants in this interface | - |
| 2 | Code Structure | Are access modifiers (public, private, protected) used correctly according to Java standards? | Yes | - | - |
| 2 | Code Structure | Are classes and interfaces clearly separated (e.g., no mixed responsibilities)? | Yes | - | - |
| 2 | Code Structure | Are packages used appropriately to organize related classes (e.g., com.pos.services)? | Yes | - | - |
| 3 | Method Design | Do methods have a single responsibility (i.e., they perform only one task)? | Yes | - | - |
| 3 | Method Design | Are method parameters kept to a reasonable limit (preferably fewer than 5)? | Yes | - | - |
| 3 | Method Design | Is method overloading used properly and not abused? | N/A | No method overloading in this interface | - |
| 4 | Exception Handling | Are exceptions handled correctly with try-catch blocks where needed? | N/A | Exception handling is delegated to implementing classes | - |
| 4 | Exception Handling | Are specific exceptions used instead of generic Exception or Throwable? | Yes | TrainException is used | - |
| 4 | Exception Handling | Is logging implemented within catch blocks for debugging purposes? | N/A | No exception handling in this interface | - |
| 5 | Code Readability | Are meaningful and descriptive comments added for complex logic? | No | No comments explaining the purpose of the interface or methods | Add JavaDoc comments |
| 5 | Code Readability | Is there consistent indentation (4 spaces or a tab size of 4)? | Yes | - | - |
| 5 | Code Readability | Are blank lines used appropriately to separate code blocks for better readability? | Yes | - | - |
| 5 | Code Readability | Is the code easy to read and understand? | Yes | - | - |
| 5 | Code Readability | Are meaningful names used for variables, classes, and methods? | Yes | - | - |
| 6 | Performance | Are data structures chosen based on their performance characteristics (e.g., ArrayList vs LinkedList)? | N/A | No specific data structure implementation in this interface | - |
| 6 | Performance | Are costly operations (e.g., database calls) minimized inside loops? | N/A | No implementation details in this interface | - |
| 6 | Performance | Is lazy initialization used to defer object creation until it is needed? | N/A | No object creation in this interface | - |
| 6 | Performance | Are there redundant calculations or excessive logging? | N/A | No calculations or logging in this interface | - |
| 7 | Memory Management | Are unnecessary object references set to null to aid garbage collection? | N/A | No object references in this interface | - |
| 7 | Memory Management | Are large objects or collections properly handled to avoid memory leaks (e.g., using WeakHashMap)? | N/A | No object handling in this interface | - |
| 7 | Memory Management | Are external resources like files, streams, and database connections properly closed to prevent resource leaks? | N/A | No resource handling in this interface | - |
| 7 | Memory Management | Is the try-with-resources statement used for automatic resource management (e.g., FileInputStream, BufferedReader)? | N/A | No resource management in this interface | - |
| 7 | Memory Management | Are large files processed efficiently (e.g., using streams or buffered reading) to avoid loading the entire file into memory? | N/A | No file processing in this interface | - |
| 7 | Memory Management | Is RandomAccessFile used appropriately for large files requiring specific access? | N/A | No file access in this interface | - |
| 7 | Memory Management | Is error handling implemented to safely close resources in case of exceptions? | N/A | No resource handling in this interface | - |
| 8 | Security | Is user input validated to prevent security issues (e.g., SQL injection, XSS)? | N/A | No input handling in this interface | - |
| 8 | Security | Are sensitive data (e.g., passwords) encrypted or hashed before storage? | N/A | No data storage in this interface | - |
| 9 | Maintainability | Are there long methods or deeply nested loops? | No | - | - |
| 9 | Maintainability | Is there duplicated code that could be refactored? | No | - | - |
| 9 | Maintainability | Are there any magic numbers or hard-coded values? | No | - | - |
| 10 | Code Smells | Are there any code smells not covered by the checklist that are present in the code? | No | - | - |
| 11 | Coding Standards | Are there any violations of coding standards not covered by the checklist that are present in the code? | No | - | - |
| 12 | Performance Inefficiencies | Are there any performance inefficiencies not covered by the checklist that are present in the code? | No | - | - |

**File # 9:**

|  |  |
| --- | --- |
| **Package Name:** | **Com.shashi.beans** |
| **File name:** | **UserBean.java** |
| **Class/ Interface name:** | **UserBean** |

| **S#** | **Category** | **Checklist Item** | **Yes/No** | **Issue** | **Fix** |
| --- | --- | --- | --- | --- | --- |
| 1 | Naming Conventions | Are class names written in PascalCase? | Yes | - | - |
| 1 | Naming Conventions | Are variable and method names written in camelCase? | No | Variable names use inconsistent naming (pWord, addR) | Rename to standard camelCase (password, address) |
| 1 | Naming Conventions | Are constants written in uppercase with underscores? | Yes | serialVersionUID is properly named | - |
| 2 | Code Structure | Are access modifiers used correctly? | Yes | - | - |
| 2 | Code Structure | Are classes and interfaces clearly separated? | Yes | - | - |
| 2 | Code Structure | Are packages used appropriately? | Yes | - | - |
| 3 | Method Design | Do methods have a single responsibility? | Yes | - | - |
| 3 | Method Design | Are method parameters kept to a reasonable limit? | Yes | - | - |
| 3 | Method Design | Is method overloading used properly? | N/A | No method overloading present | - |
| 4 | Exception Handling | Are exceptions handled correctly? | N/A | No exception handling needed in bean class | - |
| 4 | Exception Handling | Are specific exceptions used? | N/A | No exceptions used | - |
| 4 | Exception Handling | Is logging implemented within catch blocks? | N/A | No exception handling present | - |
| 5 | Code Readability | Are meaningful comments added? | No | Missing class-level JavaDoc | Add comprehensive JavaDoc |
| 5 | Code Readability | Is there consistent indentation? | Yes | - | - |
| 5 | Code Readability | Are blank lines used appropriately? | Yes | - | - |
| 5 | Code Readability | Is the code easy to read and understand? | No | Variable names are not intuitive | Use more descriptive variable names |
| 5 | Code Readability | Are meaningful names used? | No | Abbreviated names make code harder to understand | Use full descriptive names |
| 6 | Performance | Are data structures chosen appropriately? | N/A | No data structures used | - |
| 6 | Performance | Are costly operations minimized? | N/A | No complex operations present | - |
| 6 | Performance | Is lazy initialization used? | N/A | No initialization patterns needed | - |
| 6 | Performance | Are there redundant calculations? | No | - | - |
| 7 | Memory Management | Are object references handled properly? | Yes | - | - |
| 7 | Memory Management | Are large objects handled properly? | N/A | No large objects present | - |
| 7 | Memory Management | Are resources properly closed? | N/A | No resources to close | - |
| 7 | Memory Management | Is try-with-resources used? | N/A | No resource management needed | - |
| 7 | Memory Management | Are large files processed efficiently? | N/A | No file processing | - |
| 7 | Memory Management | Is RandomAccessFile used appropriately? | N/A | No file access | - |
| 7 | Memory Management | Is error handling implemented for resources? | N/A | No resources to handle | - |
| 8 | Security | Is user input validated? | No | No validation on setters | Add validation in setters |
| 8 | Security | Are sensitive data protected? | No | Password stored as plain String | Use char[] for password |
| 9 | Maintainability | Are there long methods? | No | - | - |
| 9 | Maintainability | Is there duplicated code? | No | - | - |
| 9 | Maintainability | Are there magic numbers? | No | - | - |
| 10 | Code Smells | Are there any code smells? | Yes | Poor naming conventions | Use proper naming |
| 11 | Coding Standards | Are there coding standards violations? | Yes | Missing JavaDoc, poor naming | Add documentation |
| 12 | Performance Inefficiencies | Are there performance inefficiencies? | No | - | - |

**File # 10:**

|  |  |
| --- | --- |
| **Package Name:** | **Com.shashi.beans** |
| **File name:** | **TrainExceptions.java** |
| **Class/ Interface name:** | **TrainExceptions** |

| **S#** | **Category** | **Checklist Item** | **Yes/No** | **Issue** | **Fix** |
| --- | --- | --- | --- | --- | --- |
| 1 | Naming Conventions | Are class names written in PascalCase? | Yes | - | - |
| 1 | Naming Conventions | Are variable and method names written in camelCase? | No | Variable names use inconsistent naming (pWord, addR) | Rename to standard camelCase (password, address) |
| 1 | Naming Conventions | Are constants written in uppercase with underscores? | Yes | serialVersionUID is properly named | - |
| 2 | Code Structure | Are access modifiers used correctly? | Yes | - | - |
| 2 | Code Structure | Are classes and interfaces clearly separated? | Yes | - | - |
| 2 | Code Structure | Are packages used appropriately? | Yes | - | - |
| 3 | Method Design | Do methods have a single responsibility? | Yes | - | - |
| 3 | Method Design | Are method parameters kept to a reasonable limit? | Yes | - | - |
| 3 | Method Design | Is method overloading used properly? | N/A | No method overloading present | - |
| 4 | Exception Handling | Are exceptions handled correctly? | N/A | No exception handling needed in bean class | - |
| 4 | Exception Handling | Are specific exceptions used? | N/A | No exceptions used | - |
| 4 | Exception Handling | Is logging implemented within catch blocks? | N/A | No exception handling present | - |
| 5 | Code Readability | Are meaningful comments added? | No | Missing class-level JavaDoc | Add comprehensive JavaDoc |
| 5 | Code Readability | Is there consistent indentation? | Yes | - | - |
| 5 | Code Readability | Are blank lines used appropriately? | Yes | - | - |
| 5 | Code Readability | Is the code easy to read and understand? | No | Variable names are not intuitive | Use more descriptive variable names |
| 5 | Code Readability | Are meaningful names used? | No | Abbreviated names make code harder to understand | Use full descriptive names |
| 6 | Performance | Are data structures chosen appropriately? | N/A | No data structures used | - |
| 6 | Performance | Are costly operations minimized? | N/A | No complex operations present | - |
| 6 | Performance | Is lazy initialization used? | N/A | No initialization patterns needed | - |
| 6 | Performance | Are there redundant calculations? | No | - | - |
| 7 | Memory Management | Are object references handled properly? | Yes | - | - |
| 7 | Memory Management | Are large objects handled properly? | N/A | No large objects present | - |
| 7 | Memory Management | Are resources properly closed? | N/A | No resources to close | - |
| 7 | Memory Management | Is try-with-resources used? | N/A | No resource management needed | - |
| 7 | Memory Management | Are large files processed efficiently? | N/A | No file processing | - |
| 7 | Memory Management | Is RandomAccessFile used appropriately? | N/A | No file access | - |
| 7 | Memory Management | Is error handling implemented for resources? | N/A | No resources to handle | - |
| 8 | Security | Is user input validated? | No | No validation on setters | Add validation in setters |
| 8 | Security | Are sensitive data protected? | No | Password stored as plain String | Use char[] for password |
| 9 | Maintainability | Are there long methods? | No | - | - |
| 9 | Maintainability | Is there duplicated code? | No | - | - |
| 9 | Maintainability | Are there magic numbers? | No | - | - |
| 10 | Code Smells | Are there any code smells? | Yes | Poor naming conventions | Use proper naming |
| 11 | Coding Standards | Are there coding standards violations? | Yes | Missing JavaDoc, poor naming | Add documentation |
| 12 | Performance Inefficiencies | Are there performance inefficiencies? | No | - | - |

**File # 11:**

|  |  |
| --- | --- |
| **Package Name:** | **Com.shashi.beans** |
| **File name:** | **TrainBean.java** |
| **Class/ Interface name:** | **TrainBean** |

| **S#** | **Category** | **Checklist Item** | **Yes/No** | **Issue** | **Fix** |
| --- | --- | --- | --- | --- | --- |
| 1 | Naming Conventions | Are class names written in PascalCase? | Yes | - | - |
| 1 | Naming Conventions | Are variable and method names written in camelCase? | No | Variable names use underscores (tr\_no, from\_stn) | Rename to standard camelCase (trainNumber, fromStation) |
| 1 | Naming Conventions | Are constants written in uppercase with underscores? | Yes | serialVersionUID is properly named | - |
| 2 | Code Structure | Are access modifiers used correctly? | Yes | - | - |
| 2 | Code Structure | Are classes and interfaces clearly separated? | Yes | - | - |
| 2 | Code Structure | Are packages used appropriately? | Yes | - | - |
| 3 | Method Design | Do methods have a single responsibility? | Yes | - | - |
| 3 | Method Design | Are method parameters kept to a reasonable limit? | Yes | - | - |
| 3 | Method Design | Is method overloading used properly? | N/A | No method overloading present | - |
| 4 | Exception Handling | Are exceptions handled correctly? | N/A | No exception handling needed in bean class | - |
| 4 | Exception Handling | Are specific exceptions used? | N/A | No exceptions used | - |
| 4 | Exception Handling | Is logging implemented within catch blocks? | N/A | No exception handling present | - |
| 5 | Code Readability | Are meaningful comments added? | No | Missing class-level JavaDoc | Add comprehensive JavaDoc |
| 5 | Code Readability | Is there consistent indentation? | Yes | - | - |
| 5 | Code Readability | Are blank lines used appropriately? | Yes | - | - |
| 5 | Code Readability | Is the code easy to read and understand? | No | Variable names are not intuitive | Use more descriptive variable names |
| 5 | Code Readability | Are meaningful names used? | No | Abbreviated names make code harder to understand | Use full descriptive names |
| 6 | Performance | Are data structures chosen appropriately? | N/A | No data structures used | - |
| 6 | Performance | Are costly operations minimized? | N/A | No complex operations present | - |
| 6 | Performance | Is lazy initialization used? | N/A | No initialization patterns needed | - |
| 6 | Performance | Are there redundant calculations? | No | - | - |
| 7 | Memory Management | Are object references handled properly? | Yes | - | - |
| 7 | Memory Management | Are large objects handled properly? | N/A | No large objects present | - |
| 7 | Memory Management | Are resources properly closed? | N/A | No resources to close | - |
| 7 | Memory Management | Is try-with-resources used? | N/A | No resource management needed | - |
| 7 | Memory Management | Are large files processed efficiently? | N/A | No file processing | - |
| 7 | Memory Management | Is RandomAccessFile used appropriately? | N/A | No file access | - |
| 7 | Memory Management | Is error handling implemented for resources? | N/A | No resources to handle | - |
| 8 | Security | Is user input validated? | No | No validation on setters | Add validation in setters |
| 8 | Security | Are sensitive data protected? | N/A | No sensitive data present | - |
| 9 | Maintainability | Are there long methods? | No | - | - |
| 9 | Maintainability | Is there duplicated code? | No | - | - |
| 9 | Maintainability | Are there magic numbers? | No | - | - |
| 10 | Code Smells | Are there any code smells? | Yes | Poor naming conventions | Use proper naming |
| 11 | Coding Standards | Are there coding standards violations? | Yes | Missing JavaDoc, poor naming | Add documentation |
| 12 | Performance Inefficiencies | Are there performance inefficiencies? | No | - | - |

**File # 12:**

|  |  |
| --- | --- |
| **Package Name:** | **Com.shashi.beans** |
| **File name:** | **HistoryBean.java** |
| **Class/ Interface name:** | **HistoryBean** |

| **S#** | **Category** | **Checklist Item** | **Yes/No** | **Issue** | **Fix** |
| --- | --- | --- | --- | --- | --- |
| 1 | Naming Conventions | Are class names written in PascalCase (e.g., OrderService)? | No | Class names are not following PascalCase convention. | Rename class names to follow PascalCase (e.g., orderService to OrderService). |
|  |  | Are variable and method names written in camelCase (e.g., calculateTotalPrice)? | No | Variable and method names are not written in camelCase. | Refactor all variables and methods to use camelCase (e.g., Calculateprice to calculatePrice). |
|  |  | Are constants written in uppercase with underscores (e.g., MAX\_DISCOUNT)? | No | Constants are not written in uppercase with underscores. | Update constants to use uppercase with underscores (e.g., maxDiscount to MAX\_DISCOUNT). |
| 2 | Code Structure | Are access modifiers (public, private, protected) used correctly according to Java standards? | No | Missing or incorrect use of access modifiers, leading to poor encapsulation. | Add appropriate access modifiers to all class members (e.g., use private for fields and public for methods). |
|  |  | Are classes and interfaces clearly separated (e.g., no mixed responsibilities)? | No | Classes have mixed responsibilities or unclear separation. | Refactor classes to ensure each class and interface has a single responsibility. |
|  |  | Are packages used appropriately to organize related classes (e.g., com.pos.services)? | No | Related classes are not grouped into meaningful packages. | Organize classes into packages based on functionality (e.g., com.project.services for service classes). |
| 3 | Method Design | Do methods have a single responsibility (i.e., they perform only one task)? | No | Some methods are performing multiple tasks. | Break down complex methods into smaller ones with single responsibilities. |
|  |  | Are method parameters kept to a reasonable limit (preferably fewer than 5)? | No | Methods have too many parameters, making them difficult to use and test. | Refactor methods to reduce parameters by grouping related ones into objects or using builder patterns. |
|  |  | Is method overloading used properly and not abused? | No | Overloaded methods are inconsistent or excessive, leading to confusion. | Simplify method overloading by ensuring clarity and avoiding redundant variations. |
| 4 | Exception Handling | Are exceptions handled correctly with try-catch blocks where needed? | No | Some exceptions are unhandled or handled incorrectly. | Add appropriate try-catch blocks to handle exceptions gracefully. |
|  |  | Are specific exceptions used instead of generic Exception or Throwable? | No | Generic exceptions (e.g., Exception, Throwable) are being used. | Replace generic exceptions with specific ones (e.g., IllegalArgumentException, IOException). |
|  |  | Is logging implemented within catch blocks for debugging purposes? | No | Missing logging in exception handling blocks. | Implement logging inside catch blocks to capture useful debug information. |
| 5 | Code Readability | Are meaningful and descriptive comments added for complex logic? | No | Complex logic lacks meaningful comments. | Add detailed comments to explain complex logic and algorithms. |
|  |  | Is there consistent indentation (4 spaces or a tab size of 4)? | No | Inconsistent indentation makes the code hard to read. | Reformat the code to use consistent 4-space indentation. |
|  |  | Are blank lines used appropriately to separate code blocks for better readability? | No | Lack of blank lines between logical sections of code. | Add blank lines between logical sections to enhance readability. |
|  |  | Is the code easy to read and understand? | No | Code readability is poor due to inconsistent naming and lack of formatting. | Improve naming conventions and formatting for better readability. |
|  |  | Are meaningful names used for variables, classes, and methods? | No | Some variable, class, and method names are non-descriptive or ambiguous. | Rename variables, classes, and methods to use meaningful and descriptive names. |
| 6 | Performance | Are data structures chosen based on their performance characteristics (e.g., ArrayList vs LinkedList)? | No | Suboptimal data structures are used in certain scenarios. | Evaluate and replace data structures with ones optimized for the specific use case. |
|  |  | Are costly operations (e.g., database calls) minimized inside loops? | No | Database calls are performed inside loops, impacting performance. | Move database calls outside loops and batch them where possible. |
|  |  | Is lazy initialization used to defer object creation until it is needed? | No | Objects are initialized unnecessarily, increasing memory usage. | Use lazy initialization to create objects only when required. |
|  |  | Are there redundant calculations or excessive logging? | No | Redundant calculations and logging are slowing down execution. | Eliminate redundant calculations and optimize logging statements. |
| 7 | Memory Management | Are unnecessary object references set to null to aid garbage collection? | No | Unused object references are not cleared, potentially causing memory leaks. | Set unnecessary object references to null to assist garbage collection. |
|  |  | Are large objects or collections properly handled to avoid memory leaks (e.g., using WeakHashMap)? | No | Large collections are not managed efficiently. | Use appropriate structures like WeakHashMap to manage large collections. |
|  |  | Are external resources like files, streams, and database connections properly closed to prevent leaks? | No | External resources are not consistently closed after use. | Ensure proper closure of resources using try-with-resources or explicit close calls. |
|  |  | Is the try-with-resources statement used for automatic resource management? | No | Manual resource management is used instead of try-with-resources. | Use try-with-resources for automatic resource management. |
|  |  | Are large files processed efficiently to avoid loading the entire file into memory? | No | Large files are being loaded entirely into memory. | Use streaming or buffered processing to handle large files. |
| 8 | Security | Is user input validated to prevent security issues (e.g., SQL injection, XSS)? | No | Lack of input validation exposes the system to security vulnerabilities. | Validate all user input using input sanitization and parameterized queries. |
|  |  | Are sensitive data (e.g., passwords) encrypted or hashed before storage? | No | Sensitive data is stored in plaintext. | Use encryption or hashing (e.g., bcrypt) to securely store sensitive data. |
| 9 | Maintainability | Are there long methods or deeply nested loops? | No | Some methods are overly long or have deeply nested loops, making them hard to maintain. | Refactor long methods into smaller ones and simplify deeply nested loops. |
|  |  | Is there duplicated code that could be refactored? | No | Duplicate code is present in multiple locations. | Refactor duplicate code into reusable methods or classes. |
|  |  | Are there any magic numbers or hard-coded values? | No | Hard-coded values are present in the code. | Replace hard-coded values with constants or configuration properties. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | Code Smells | Are there any code smells? | Yes | Poor naming conventions | Use proper naming |
| 11 | Coding Standards | Are there coding standards violations? | Yes | Missing JavaDoc, poor naming | Add documentation |
| 12 | Performance Inefficiencies | Are there performance inefficiencies? | No | - | - |

**File # 13:**

|  |  |
| --- | --- |
| **Package Name:** | **Com.shashi.beans** |
| **File name:** | **BookingDetail.java** |
| **Class/ Interface name:** | **BookingDetail** |

| **Category** | **Checklist Item** | **Yes/No** | **Issue** | **Fix** |
| --- | --- | --- | --- | --- |
| Naming Conventions | Are class names written in PascalCase? | Yes | - | - |
| Naming Conventions | Are variable and method names written in camelCase? | Yes | - | - |
| Naming Conventions | Are constants written in uppercase with underscores? | Yes | - | - |
| Code Structure | Are access modifiers used correctly? | No | Fields are package-private | Make fields private and use public getters/setters |
| Code Structure | Are classes and interfaces clearly separated? | Yes | - | - |
| Code Structure | Are packages used appropriately? | Yes | - | - |
| Method Design | Do methods have a single responsibility? | Yes | - | - |
| Method Design | Are method parameters kept to a reasonable limit? | Yes | - | - |
| Method Design | Is method overloading used properly? | Yes | - | - |
| Exception Handling | Are exceptions handled correctly? | N/A | No exception handling in this class | - |
| Exception Handling | Are specific exceptions used? | N/A | No exception handling in this class | - |
| Exception Handling | Is logging implemented within catch blocks? | N/A | No exception handling in this class | - |
| Code Readability | Are meaningful comments added? | No | No comments present | Add JavaDoc comments for class and fields |
| Code Readability | Is there consistent indentation? | Yes | - | - |
| Code Readability | Are blank lines used appropriately? | Yes | - | - |
| Code Readability | Is the code easy to read and understand? | Yes | - | - |
| Code Readability | Are meaningful names used? | Yes | - | - |
| Performance | Are data structures chosen based on performance? | N/A | No collections used | - |
| Performance | Are costly operations minimized inside loops? | N/A | No loops or database operations | - |
| Performance | Is lazy initialization used? | N/A | No object creation in this class | - |
| Performance | Are there redundant calculations or excessive logging? | No | - | - |
| Memory Management | Are unnecessary object references set to null? | N/A | No object references in this class | - |
| Memory Management | Are large objects or collections properly handled? | N/A | No collections in this class | - |
| Memory Management | Are external resources properly closed? | N/A | No external resources used | - |
| Memory Management | Is try-with-resources used? | N/A | No resources used in this class | - |
| Memory Management | Are large files processed efficiently? | N/A | No file processing in this class | - |
| Memory Management | Is RandomAccessFile used appropriately? | N/A | No file access in this class | - |
| Memory Management | Is error handling implemented to safely close resources? | N/A | No resources or exception handling | - |
| Security | Is user input validated? | N/A | No user input handling in this class | - |
| Security | Are sensitive data encrypted or hashed? | N/A | No sensitive data handling | - |
| Maintainability | Are there long methods or deeply nested loops? | No | - | - |
| Maintainability | Is there duplicated code that could be refactored? | No | - | - |
| Maintainability | Are there any magic numbers or hard-coded values? | No | - | - |
| Code Smells | Are there any code smells not covered by the checklist? | No | - | - |
| Coding Standards | Are there any violations of coding standards not covered? | No | - | - |
| Performance Inefficiencies | Are there any performance inefficiencies not covered? | No | - | - |

**File # 14:**

|  |  |
| --- | --- |
| **Package Name:** | **Com.shashi.beans** |
| **File name:** | **Adminbean.java** |
| **Class/ Interface name:** | **AdminBean** |

| **Category** | **Checklist Item** | **Yes/No** | **Issue** | **Fix** |
| --- | --- | --- | --- | --- |
| Naming Conventions | Are class names written in PascalCase? | Yes | - | - |
| Naming Conventions | Are variable and method names written in camelCase? | No | Variables like pWord, addR have inconsistent naming | Rename to password, address |
| Naming Conventions | Are constants written in uppercase with underscores? | Yes | - | - |
| Code Structure | Are access modifiers used correctly? | Yes | Fields are private with public getters/setters | - |
| Code Structure | Are classes and interfaces clearly separated? | Yes | - | - |
| Code Structure | Are packages used appropriately? | Yes | - | - |
| Method Design | Do methods have a single responsibility? | Yes | - | - |
| Method Design | Are method parameters kept to a reasonable limit? | Yes | - | - |
| Method Design | Is method overloading used properly? | Yes | - | - |
| Exception Handling | Are exceptions handled correctly? | N/A | No exception handling in this class | - |
| Exception Handling | Are specific exceptions used? | N/A | No exception handling in this class | - |
| Exception Handling | Is logging implemented within catch blocks? | N/A | No exception handling in this class | - |
| Code Readability | Are meaningful comments added? | No | No comments present | Add JavaDoc comments for class and fields |
| Code Readability | Is there consistent indentation? | Yes | - | - |
| Code Readability | Are blank lines used appropriately? | Yes | - | - |
| Code Readability | Is the code easy to read and understand? | No | Variable names are abbreviated | Use full descriptive names |
| Code Readability | Are meaningful names used? | No | Variables use abbreviations | Use complete words |
| Performance | Are data structures chosen based on performance? | N/A | No collections used | - |
| Performance | Are costly operations minimized inside loops? | N/A | No loops or database operations | - |
| Performance | Is lazy initialization used? | N/A | No object creation in this class | - |
| Performance | Are there redundant calculations or excessive logging? | No | - | - |
| Memory Management | Are unnecessary object references set to null? | N/A | No object references in this class | - |
| Memory Management | Are large objects or collections properly handled? | N/A | No collections in this class | - |
| Memory Management | Are external resources properly closed? | N/A | No external resources used | - |
| Memory Management | Is try-with-resources used? | N/A | No resources used in this class | - |
| Memory Management | Are large files processed efficiently? | N/A | No file processing in this class | - |
| Memory Management | Is RandomAccessFile used appropriately? | N/A | No file access in this class | - |
| Memory Management | Is error handling implemented to safely close resources? | N/A | No resources or exception handling | - |
| Security | Is user input validated? | No | No validation for password or email | Add input validation |
| Security | Are sensitive data encrypted or hashed? | No | Password stored as plain text | Implement password hashing |
| Maintainability | Are there long methods or deeply nested loops? | No | - | - |
| Maintainability | Is there duplicated code that could be refactored? | No | - | - |
| Maintainability | Are there any magic numbers or hard-coded values? | No | - | - |
| Code Smells | Are there any code smells not covered by the checklist? | Yes | Abbreviated variable names | Use full descriptive names |
| Coding Standards | Are there any violations of coding standards not covered? | Yes | Inconsistent naming conventions | Follow Java naming conventions |
| Performance Inefficiencies | Are there any performance inefficiencies not covered? | No | - | - |