**Daniel Fernández**

Manage Dining Experience application is a command line application that helps manage dining experience and calculate the total cost based on selected meals and quantities. The report provides an overview of the application, its requirements, development process, and generated code following best practices.

The Dining Experience Manager application is designed to meet specific requirements and ensure compliance with best coding practices. The application is built in Java using the Eclipse IDE and provides the following key features:

* **Menu and Meal Selection:** The application displays a menu with dining options and their corresponding prices. Users can select multiple meals and specify quantities for each.
* **Meal Quantity Validation:** The system validates that the quantity entered for each meal is a positive integer greater than zero. If invalid quantities are entered, users are prompted to re-enter them.
* **Cost Calculation:** The base cost for a dining experience is $5, applied to every order. Discounts are applied based on the total quantity of meals ordered.
* **Special Offer Discounts:** The application can apply special offer discounts based on the total cost of the order, offering discounts for orders exceeding certain thresholds.
* **Meal Availability:** The system validates that the selected meals are available on the menu and prompts users to re-select meals if an unavailable meal is chosen.
* **Maximum Order Quantity:** The application handles orders with a maximum quantity of 100 meals and displays an error message if users exceed this limit.
* **User Confirmation:** Before finalizing the order, the system displays the selected meals, their quantities, and the total cost for user confirmation. Users can confirm the order or cancel and make changes.
* **Error Handling:** The system handles errors gracefully and provides clear error messages if users enter invalid input or if there are calculation errors.

The code was developed following best practices and various fixes were made to address violations of PMD rules and improve its readability and maintainability.

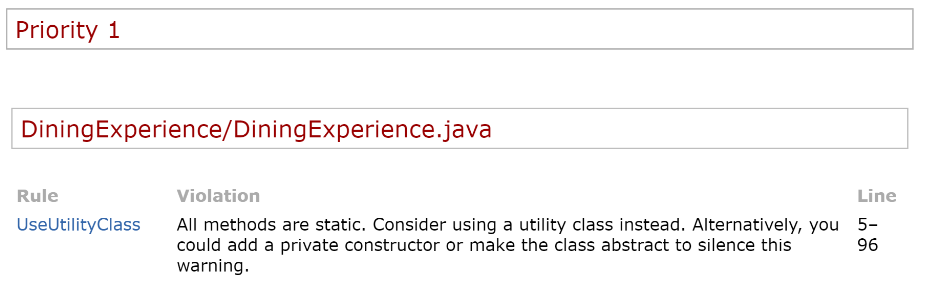


Figure 1.- First error to correct.



Figure 2.- Second error to fix.

The integration of PMD as a code quality assessment tool proved beneficial in identifying and resolving code. By addressing these issues, the development improved code readability, maintainability, and adherence to best practices.

**Challenge**

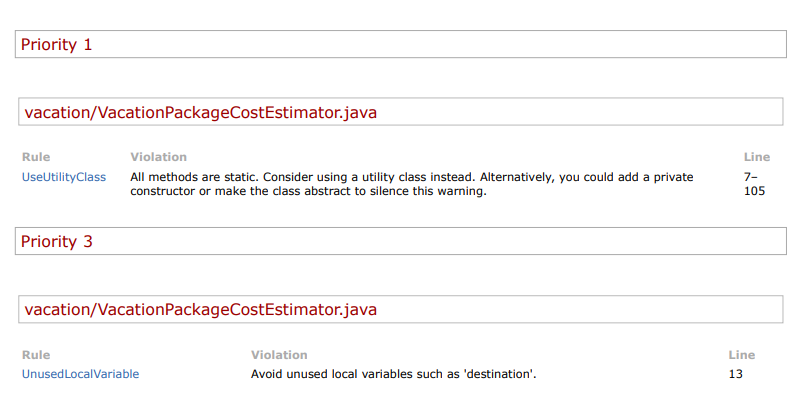


Figure .- PMD challenge

**References**

Reference 1: [PMD - SourceForge](https://pmd.github.io/)

Reference 2: [Java Coding Standards and Best Practices](https://www.oracle.com/java/technologies/javase/codeconventions-introduction.html)

**Github**

[**https://github.com/dafebust/CodeInspection**](https://github.com/dafebust/CodeInspection)

[**https://github.com/dafebust/Coding\_Standars/tree/feature/vacation-addons**](https://github.com/dafebust/Coding_Standars/tree/feature/vacation-addons)