**Code Smells:**

José Morgado 59457

1. **Comments code smell:**

The FreeColClient.java class, located in the src/net/sf/freecol/client directory, contains a method that shows signs of having too many comments. Well-defined methods, as shown in Figure 1 and Figure 2, do not require redundant comments.

A possible solution to this issue would be to remove comments from functions that clearly specify their behavior solely through their names.

Fig. 1 - isLoggedIn redundant method

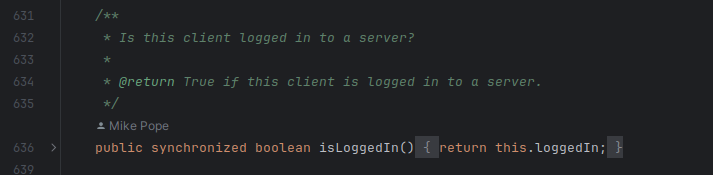
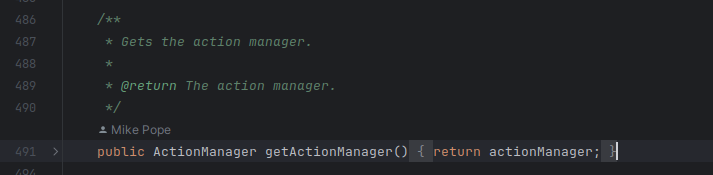


Fig. 2 - Action Manager redundant method

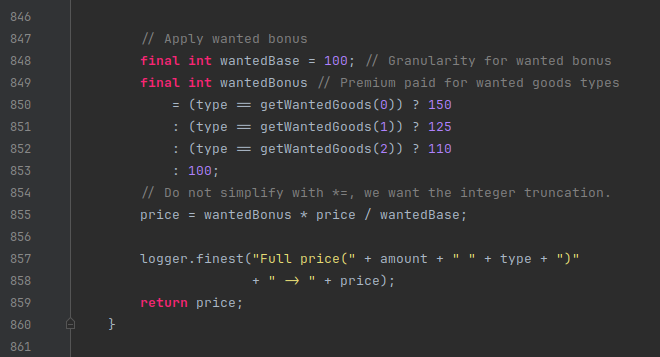
1. **Magic Numbers code smell**

The code within the IndianSettlement.java class in the src/net/sf/freecol/common/model directory may exhibit the "magic numbers" code smell, indicated by the presence of hard-coded numeric values lacking clear explanation or context.

Refactoring this code by replacing such magic numbers with named constants or variables having descriptive names can significantly enhance code readability and maintainability. By introducing named constants like PREMIUM\_WANTED\_PRICE = 150 and similar, the purpose of these numbers becomes explicit, improving the code's comprehensibility for future developers.

Addressing the reliance on magic numbers in the IndianSettlement.java class can lead to clearer and more maintainable code, facilitating easier comprehension and future modifications.

Fig. 3 - Some magic numbers



1. **Instance Type Checking code smell**

The class UnitWas in the src/net/sf/freecol/common/model directory appears to present the “Instance Type Checking” code smell, as the multiple chained ternary operators perform type checking (instanceof) for various types of FreeColGameObject to determine a specific change type. This practice violates the principles of polymorphism and object-oriented design by centralizing decision-making logic based on the concrete type of objects.

This leads to less maintainable code as any modification or addition of types “FreeColGameObject” will require changing the existing logic and thus violate the Open-Closed Principle (of the SOLID Principles).

We could fix this by adding the change function to each of the child objects and calling it directly from the given object, instead of checking the type in the parent function.

Fig. 4 - “non-OO nastiness”