**Code Smells:**

Nádia Mendes 53175

1. **Long Method:**

The method *moveToDestination* present in *the InGameController.java* class contained in the *src.net.sf.freecol.client.control* package is an extremely long method, it starts on line 1196 and ends on line 1256, containing 60 lines of code. The code, in addition to being extensive, also contains a lot of logic. Therefore, the Long Method code smell is verified.

A possible solution to this problem would be to divide this method into smaller and simpler submethods. This would clearly make the code easier to read and maintain.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Fig. 1 to 3 Represent the *moveToDestination* method in its entirety.

1. **Large Class**

When we analyze the *InGameController* class present in the *src.net.sf.freecol.client.control* package, we see that the class has many responsibilities and methods, being an extremely long class, containing a total of 5387 lines. This way we verify that we are in the presence of the large class code smell. One solution would be to split this class into smaller classes, each with a single responsibility. For example, the methods that in this class deal with the movement of units, such as *moveToDestination*, *movePath* and *moveDirection*, could be placed in a new class, which only deals with the movement of units.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Fig.4 Represents part of the *moveToDestination* method.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Fig.5 Partial representation of the *moveDirection* method.

1. **Data Clumps**

There is evidence of data groupings, in the *InGameController*.*java* class contained in the *src.net.sf.freecol.client.control* package, there are methods to receive many parameters, such as the *moveToDestination* method almost starts on line 1196, where the unit is passed as a parameter to almost all unit-related method calls such as *followTradeRoute*, *moveTile*, *moveAttack*, etc. This can be considered a data clump since the unit object is always related to these movement operations, and the same Parcels are repeated past.

A possible solution would be to create objects to group related data and make the code more readable.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Fig. 6 to 8 Representation of the *moveToDestination* method.