**Code Smells:**

Nádia Mendes 53175

1. **Long Method:**

O método *moveToDestination* presente na classe *InGameController.java* contida no package *src.net.sf.freecol.client.control* , é um método extremamente longo, este inicia-se na linha 1196 e acaba na linha 1256, contendo 60 linhas de código. O código alem de ser extenso verifica-se também, que contém muita lógica. Assim sendo verifica-se o code smell Long Method.

Uma solução possível a este problema seria dividirmos este método em submétodos mais pequenos e simples. Isto tronaria claramente o código mais fácil de ler e manter.

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 a 3 Representam o método *moveToDestination* na sua totalidade.

1. **Large Class**

Ao analisarmos a classe *InGameController* presente no package *src.net.sf.freecol.client.control* verificamos que a classes possui muitas responsabilidades e métodos, sendo uma classe extremamente longa, contendo um total de 5387 linhas. Deste modo verificamos estar na presença do code smell large class. Uma solução seria dividir esta classe em classes menores, cada uma com uma única responsabilidade. Por exemplo os métodos que nesta classe lidam com o movimento de unidades, tais como moveToDestination, movePath e moveDirection, poderiam ser colocados numa nova classe, que apenas lidasse com o movimento das unidades.

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

Descrição gerada automaticamente

Fig.4 Representa parte do método *moveToDestination*.

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

Descrição gerada automaticamente

Fig.5 Representação parcial do método *moveDirection.*

1. **Data Clumps**

Há evidencia de agrupamentos de dados, na classe *InGameController.java* contida no package *src.net.sf.freecol.client.control,* nesta existem métodos a receber muitos parâmetros, tais como o método *moveToDestination* quaseinicia na linha 1196, em que o unit é passado como um parâmetro para quase todas as chamadas de métodos relacionados a unidades, como *followTradeRout*e, *moveTile, moveAttack, etc.* Isto pode ser considerado um data clump uma vez que o objecto unit está sempre relacionado a essas operações de movimento, e os mesmos parâmetros são repetidamente passados.

Uma possível solução seria criar objetos para agrupar dados relacionados e tornar o código mais legível.

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 a 8 Representação do método *moveToDestination.*