**Patterns**

Nádia Mendes 53175

1. **template method pattern:**

O método *followTradeRoute* parece apresentar o padrão template method pattern, uma vez que este define um esqueleto do algoritmo com passos concretos definidos em métodos concretos , como *unloadUnitAtStop*, *loadUnitAtStop* entre outros. Verifica-se deste modo que o método em si, esta mais preocupado com a atribuição de responsabilidades. As subclasses (ou neste caso os métodos privados) que estendem e substituiu em esses passos conforme necessário.

Este método inicia-se na linha 2145 da classe *InGameController.java* contida no package *src.net.sf.freecol.client.control* e finaliza na linha 2283, como é um método extenso a representação completa deste foi dividida desde a figura 3 até a figura 5.

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

Descrição gerada automaticamente

Fig.1 início do método *followTradeRoute*

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

Descrição gerada automaticamente

Fig.2 parte do código do método *followTradeRoute* onde se verifica a chamada aos métodos privados *unloadUnitAtStop* e *loadUnitAtStop.*

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

Descrição gerada automaticamente

Fig.3 Representação da parte inicial do método *followTradeRoute*

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

Descrição gerada automaticamente

Fig4. Continuação da representação do método *followTradeRoute.*

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

Descrição gerada automaticamente

Fig5. Continuação da representação do método *followTradeRoute.*

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

Descrição gerada automaticamente

Fig6. Continuação da representação do método *followTradeRoute.*

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

Descrição gerada automaticamente

Fig7. Finalização da representação do método *followTradeRoute.*

1. **command Pattern:**

Os métodos como *moveTile*, *moveScountColony, moveSpymoveTrade, moveTribute*, *moveUseMissionary* contidos na classe *InGameController.java* do package *src.net.sf.freecol.client.control* representam a encapsulação de uma solicitação como um objeto, o que se assemelha ao command Pattern, uma vez que estes encapsulam uma unidade de ação e os parâmetros necessários para realizar essa ação.

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

Descrição gerada automaticamente

Fig8. Método *moveScountColony.*

1. **Abstract Factory Pattern:**

Na classe *LanguageOptionUI,* contida no package *src.net.sf.freecol.client.gui.option* verifica-se que esta atua como uma fábrica abstrata para criar objetos relacionados à opção de idioma(LanguageOption)

A classe *LanguageOptionUI* cria e retorna uma instância de JComboBox <Language> que é uma parte da família de objetos de interface do usuário relacionados à opção de idioma.

A classe *LanguageOption* representa a opção de idioma, enquanto a classe Language representa os idiomas disponíveis.

Portanto, verifica-se o uso do *Abstract Factory Pattern,* este é usado para criar objetos relacionados de acordo com a escolha do idioma, e isso permite criar uma família de objetos coerentes relacionados à opção do idioma.

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

Descrição gerada automaticamente

Fig.9 Representação da parte inicial da classe *LangugeOptionUI*.

Uma imagem com texto, captura de ecrã, software

Descrição gerada automaticamente

Fig.10 Representação do código restante da classe *LangugeOptionUI*.