**Patterns**

Nádia Mendes 53175

1. **template method pattern:**

The *followTradeRoute* method appears to present the template method pattern, as it defines a skeleton of the algorithm with concrete steps defined in concrete methods, such as *unloadUnitAtStop*, *loadUnitAtStop*, among others. It can thus be seen that the method itself is more concerned with the attribution of responsibilities. The subclasses (or in this case the private methods) extend and override these steps as needed.

This method starts on line 2145 of the InGameController.java class contained in the *src.net.sf.freecol.client.control* package and ends on line 2283, as it is an extensive method the complete representation of this was divided from figure 3 to figure 5.

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

Descrição gerada automaticamente

Fig.1 start of the *followTradeRoute* method.

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

Descrição gerada automaticamente

*Fig.2 part of the followTradeRoute method code where the call to the private unloadUnitAtStop and loadUnitAtStop methods takes place.*

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

Descrição gerada automaticamente

Fig.3 Representation of the initial part of the *followTradeRoute* method

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

Descrição gerada automaticamente

*Fig4. Continued representation of the followTradeRoute method.*Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Fig5. Continued representation of the followTradeRoute method.

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

Descrição gerada automaticamente

Fig6. Continued representation of the followTradeRoute method.

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

Descrição gerada automaticamente

*Fig7.* Finalization of the representation of thefollowTradeRoutemethod*.*

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*.