**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:**

Methods such as *moveTile*, *moveScountColony*, *moveSpymoveTrade*, *moveTribute*, *moveUseMissionary* contained in the InGameController.java class of the *src.net.sf.freecol.client.control* package represent the encapsulation of a request as an object, which is similar to the command Pattern, a since these encapsulate a unit of action and the parameters necessary to carry out that action.

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

Descrição gerada automaticamente

Fig8. Method *moveScountColony.*

**3-Abstract Factory Pattern:**

In the LanguageOptionUI class, contained in the src.net.sf.freecol.client.gui.option package, it appears that it acts as an abstract factory to create objects related to the language option (LanguageOption)

The LanguageOptionUI class creates and returns an instance of JComboBox <Language> which is a part of the language option-related family of UI objects.

The LanguageOption class represents the language option, while the Language class represents the available languages.

Therefore, the use of the Abstract Factory Pattern is used to create related objects according to the language choice, and this allows the creation of a family of coherent objects related to the language choice.

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

Descrição gerada automaticamente

Fig.9 Representation of the initial part of the *LangugeOptionUI* class.

Uma imagem com texto, captura de ecrã, software

Descrição gerada automaticamente

Fig.10 Representation of the remaining code of the *LangugeOptionUI* class.