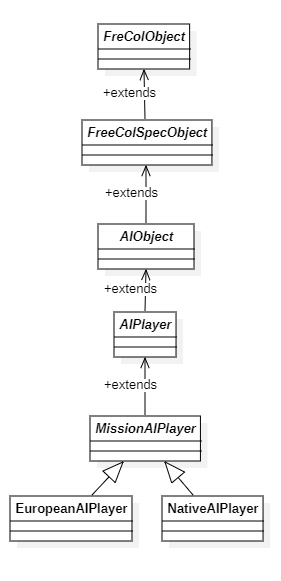
José Trigueiro 58119

**Design Patterns**

1. **Template method pattern -** The pattern has a skeleton of operations, and the details are implemented by the child classes. This means that the overall structure and sequence of the algorithm are preserved by the parent class. The pattern encapsulates the algorithms in separate classes.



\src\net\sf\freecol\server\ai\AIObject.java



\src\net\sf\freecol\common\model\FreeColSpecObject.java



\src\net\sf\freecol\server\ai\AIObject.java



\src\net\sf\freecol\server\ai\AIPlayer.java



\src\net\sf\freecol\server\ai\MissionAIPlayer.java



\src\net\sf\freecol\server\ai\EuropeanAIPlayer.java



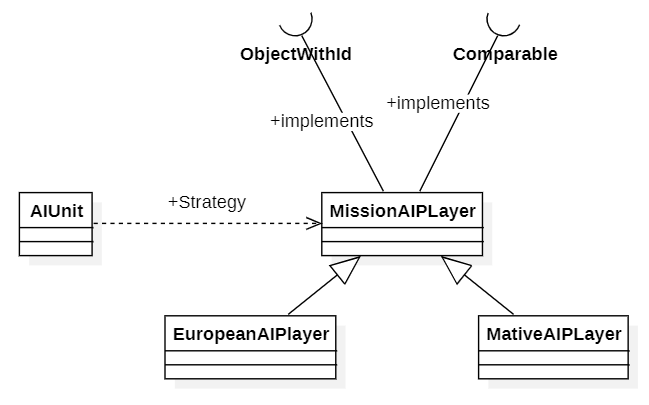
\src\net\sf\freecol\server\ai\NativeAIPlayer.java



1. **Strategy Design Pattern -**

It provides a flexible way to encapsulate and swap behavior of an object.

The behavior of the object’s behavior can change dynamically.



\src\net\sf\freecol\server\ai\MissionAIPlayer.java

\src\net\sf\freecol\server\ai\EuropeanAIPlayer.java

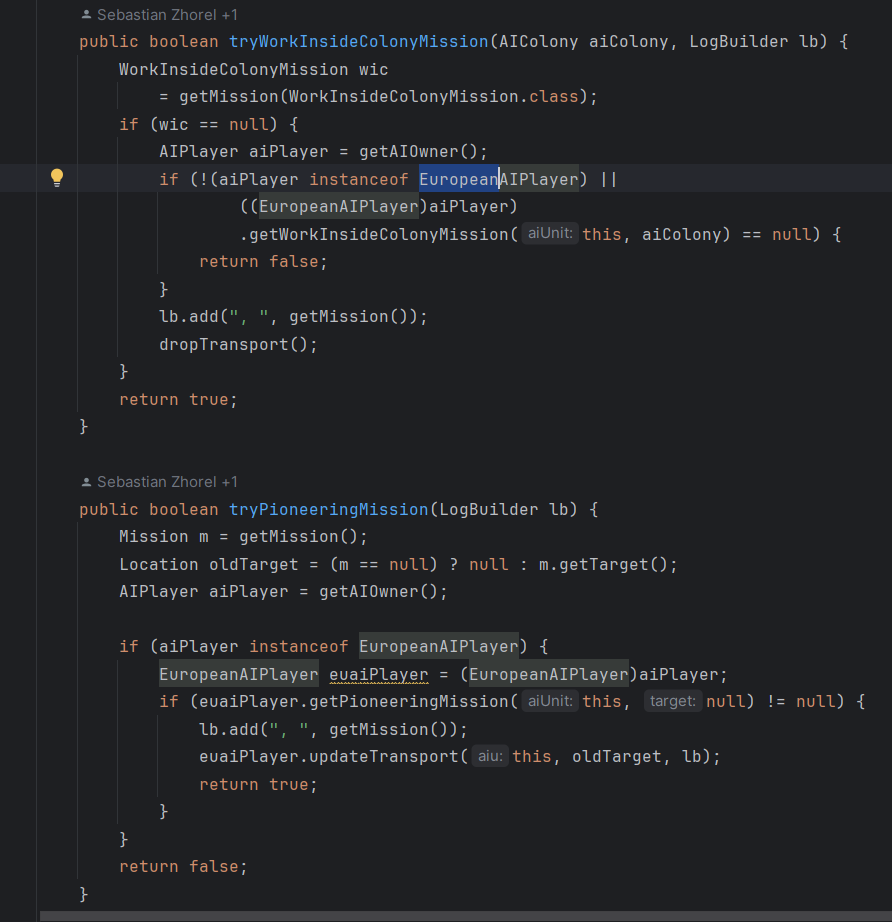


\src\net\sf\freecol\server\ai\NativeAIPlayer.java

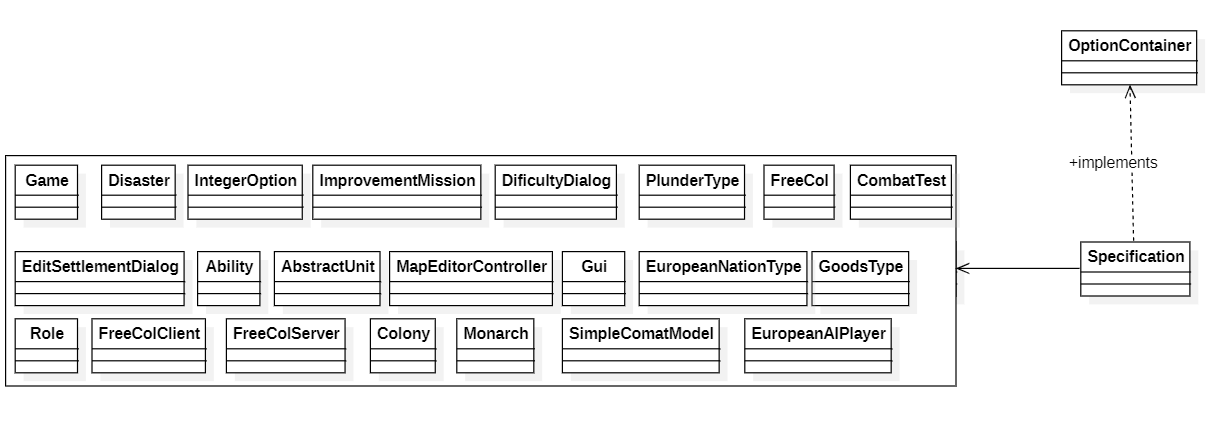


src/net/sf/freecol/server/ai/AIPlayer.java



-

1. **Factory Method Pattern -** It has an interface “OptionContainer.java” for creating objects, but allows subclasses like “Specification” to decide which class to instantiate. It encapsulates the logic required to instantiate several complex objects.



src/net/sf/freecol/common/option/OptionContainer.java



src/net/sf/freecol/common/model/Specification.java



