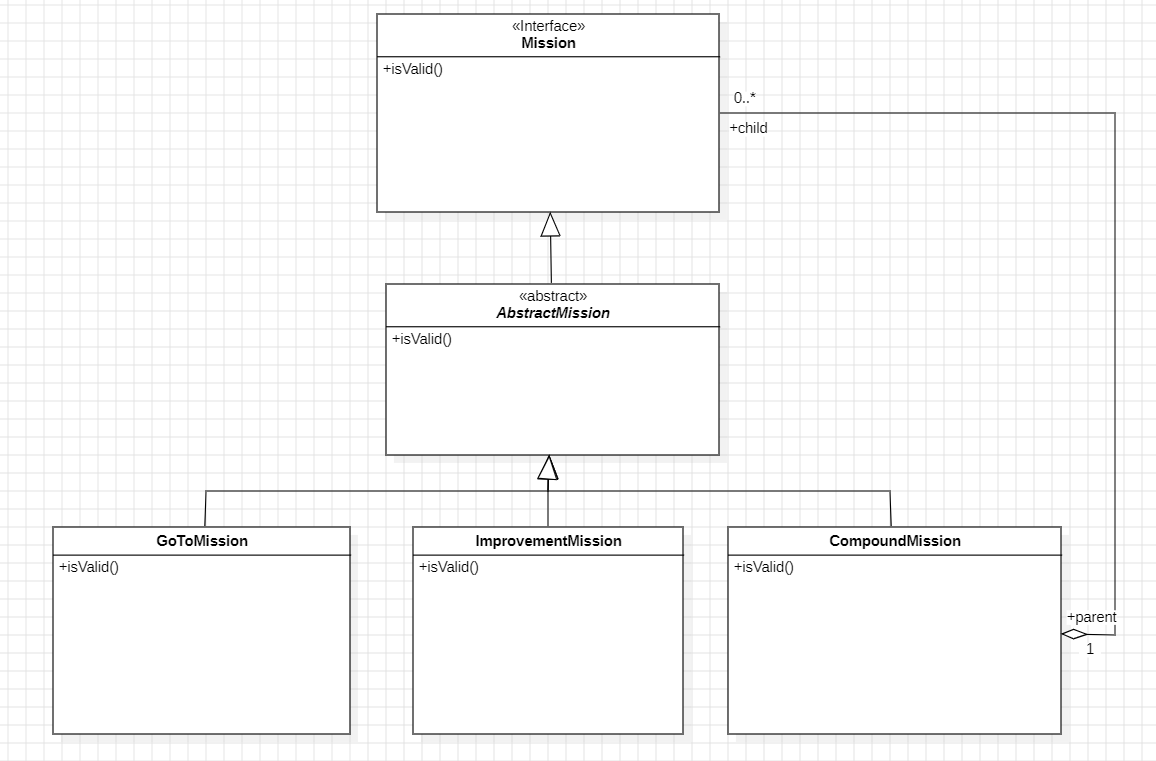
**Composite Pattern**



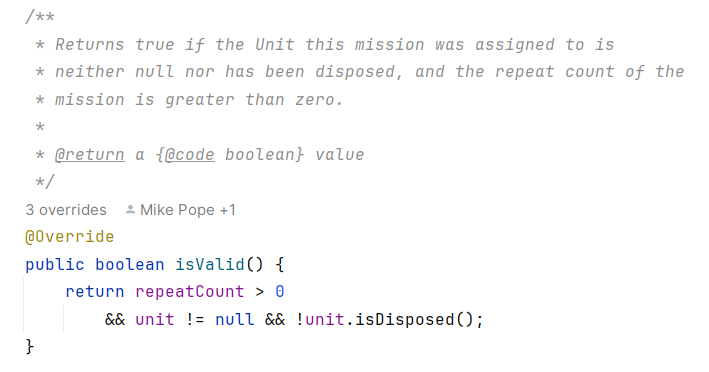
Localização: src/net/sf/freecol/common/model/mission

This class diagram represents a composite pattern because the class CompoundMisson has a List of Mission, where it stores other instances of the type Mission and in the method isValid() all the objects in the list are checked for validity. The method doMission works the same way.

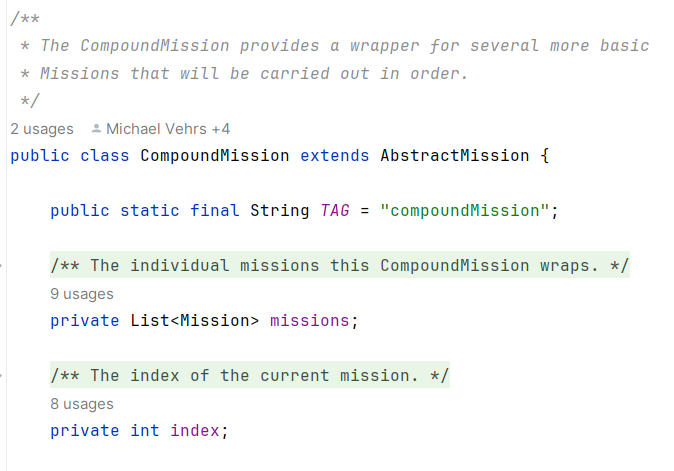
src/net/sf/freecol/common/model/mission/Mission.java:

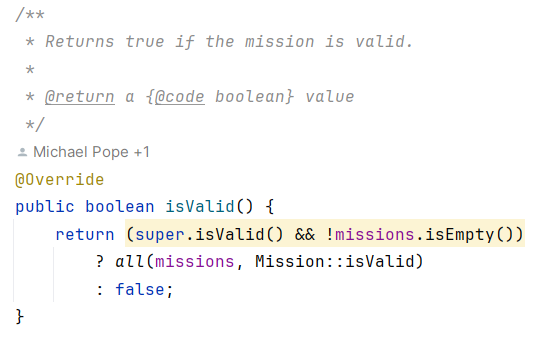


src/net/sf/freecol/common/model/mission/AbstractMission.java

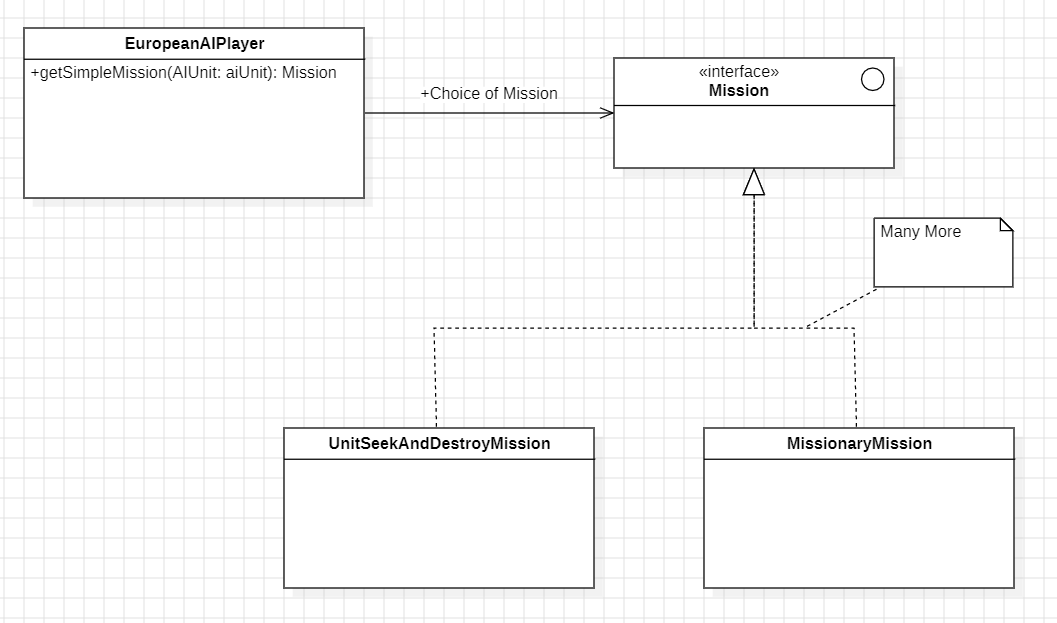


src/net/sf/freecol/common/model/mission/CompoundMission.java





**Strategy design Pattern**



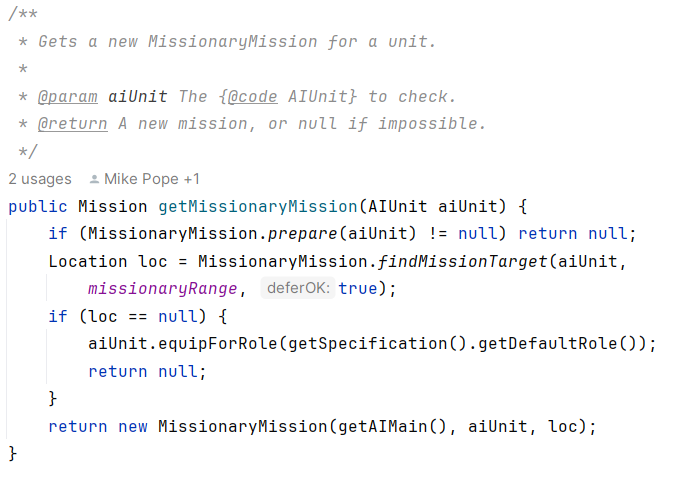
Localização: src/net/sf/freecol/server/ai/EuropeanAIPlayer.java

This relation is a Strategy Pattern because it has a class called getSimpleMission where, considering several conditions in game and of the unit itself, a unit is assigned a mission to perform which can be of many different types like UnitSeekAndDestroyMission or MissionaryMission. This will then change the behavior of the unit in game.

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

|  |
| --- |
| /\*\* \* Choose a mission for an AIUnit. \* \* **@param** aiUnit The {**@code** AIUnit} to choose for. \* **@return** A suitable {**@code** Mission}, or null if none found. \*/ private Mission getSimpleMission(AIUnit aiUnit) {  final Unit unit = aiUnit.getUnit();  Mission m, ret;  final Mission old = ((m = aiUnit.getMission()) != null && m.isValid())  ? m : null;   if (unit.isNaval()) {  ret = (old instanceof PrivateerMission) ? old  : ((m = getPrivateerMission(aiUnit, null)) != null) ? m  : (old instanceof TransportMission) ? old  : ((m = getTransportMission(aiUnit)) != null) ? m  : (old instanceof UnitSeekAndDestroyMission) ? old  : ((m = getSeekAndDestroyMission(aiUnit, 8)) != null) ? m  : (old instanceof UnitWanderHostileMission) ? old  : getWanderHostileMission(aiUnit);   } else if (unit.isCarrier()) {  ret = getTransportMission(aiUnit);   } else {  // CashIn missions are obvious  ret = (old instanceof CashInTreasureTrainMission) ? old  : ((m = getCashInTreasureTrainMission(aiUnit)) != null) ? m   // Working in colony is obvious  : (unit.isInColony()  && old instanceof WorkInsideColonyMission) ? old  : (unit.isInColony()  && (m = getWorkInsideColonyMission(aiUnit, null)) != null) ? m    // Try to maintain local defence  : (old instanceof DefendSettlementMission && old.getTarget() instanceof Colony && !((Colony) old.getTarget()).isVeryWellDefended()) ? old  : ((m = getDefendCurrentSettlementMission(aiUnit)) != null) ? m   // REF override  : (unit.hasAbility(Ability.REF\_UNIT))  ? ((old instanceof UnitSeekAndDestroyMission) ? old  : ((m = getSeekAndDestroyMission(aiUnit, 12)) != null) ? m  : (m = getWanderHostileMission(aiUnit)))   // Favour wish realization for expert units  : (unit.isColonist() && unit.getSkillLevel() > 0  && old instanceof WishRealizationMission) ? old  : (unit.isColonist() && unit.getSkillLevel() > 0  && (m = getWishRealizationMission(aiUnit, null)) != null) ? m   // Ordinary defence  : ((m = getDefendSettlementMission(aiUnit, false, false)) != null) ? m   // Try nearby offence  : (old instanceof UnitSeekAndDestroyMission) ? old  : ((m = getSeekAndDestroyMission(aiUnit, 8)) != null) ? m   // Missionary missions are only available to some units  : (old instanceof MissionaryMission) ? old  : ((m = getMissionaryMission(aiUnit)) != null) ? m   // Try to satisfy any remaining wishes, such as population  : (old instanceof WishRealizationMission) ? old  : ((m = getWishRealizationMission(aiUnit, null)) != null) ? m   // Another try to defend, with relaxed cost decider  : ((m = getDefendSettlementMission(aiUnit, true, false)) != null) ? m   // Another try to attack, at longer range  : ((m = getSeekAndDestroyMission(aiUnit, 16)) != null) ? m    // Try again, even for well defended colonies.  : ((m = getDefendSettlementMission(aiUnit, true, true)) != null) ? m   // Leftover offensive units should go out looking for trouble  : (old instanceof UnitWanderHostileMission) ? old  : ((m = getWanderHostileMission(aiUnit)) != null) ? m   : null;  }  return ret; } |

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



Template Pattern

Uma imagem com texto, diagrama, file, captura de ecrã

Descrição gerada automaticamente

Location: src/net/sf/freecol/common/networking

This Relation is a template pattern because all the classes that implement the Interface MessageHandler perform different actions when called to do the same methods.

src/net/sf/freecol/common/networking/MessageHandler.java

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

Descrição gerada automaticamente

src/net/sf/freecol/client/control/ClientInputHandler.java

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

Descrição gerada automaticamente

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

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

Descrição gerada automaticamente