Design Patterns:

* Factory Method, ActionManager()

Uma imagem com texto, captura de ecrã, menu

Descrição gerada automaticamenteUma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Existe um factory method na classe ActionManager, das linhas 73 á 177

O método em questão tem a responsabilidade de criar todas as Actions no próprio, sendo este método chamado em diversos outros métodos

Tal como o comentário sugere, a possibilidade de remover alguns destes Actions e os colocar em grupos próprios (tal como OptionGroups) seria algo que facilitaria a manutenção desse codigo

* Decorator, ColopediaGameObjectTypePanel()

Uma imagem com texto, captura de ecrã

Descrição gerada automaticamente

ColopediaGameObjectTypePanel é uma classe abstrata que é utilizado em diversas classes (algumas dessas designadas na imagem, tal como “FatherDeatilPanel”, “GoodsDetailPanel”, etc) e ligada á interface ColopediaDetailPanel, sendo que a classe abstrata é usada nas subclasses para completar o código usado.

A classe em questão, ColopediaGameObjectTypePanel, permite uma composição com outras classes componentes e ao mesmo nível (ConceptDetailPanel) e também permite ser uma superclasse para as subclasses já mencionadas, impedindo assim a repetição de código

Devido a essas características, podemos indicar a classe abstrata como um Decorator

* Template, AbstractMission()

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

Descrição gerada automaticamente

AbstractMission é uma classe abstrata ligada á interface Mission e que conecta às subclasses “CompoundMission”, “GoToMission” e “Inprovement Mission”, sendo uma classe que ajusta métodos usados nas 3 classes   
  
A classe em questão, AbstractMission, permite uma composição com outras classes componentes do tipo Mission e também permite ser uma superclasse para as subclasses já mencionadas, impedindo assim a repetição de código

Assim, AbstractMission é um template