在本次设计中选择了solitaire纸牌游戏作为设计题目，因为考虑到游戏设计较为复杂，因此设计分成两个部分来进行开发——前端（游戏界面）和后端（游戏逻辑），实现前后端分离开发。

在这里主要介绍的是后端部分的开发，前端的部分由另一个文档来说明。

首先我们分析一下这个题目，要实现一个纸牌游戏，那我们先观察有什么功能，首先要能初始化出那么多牌，然后这些牌要能够在各个组件中互相移动，同时要统计分数，并计算时间，还需要给出当前牌局的提示，并检查游戏是否成功或者失败。

那么要做一个纸牌游戏，由于无需考虑前端界面，只需考虑游戏逻辑，因此首先可以拟定一个游戏的整体接口CardGame，用于表示所有的游戏操作，具体的游戏操作再向下进行分发。

在这个游戏中，分为几个部分，分别是CardManagement, CardAnalyzer, CardInitializaer, PointCounter和Timer。CardManagement负责所有牌和牌堆的管理，包括牌的位置和牌的移动的管理，CardAnalyzer负责检查每次操作之后是否能够移动，是否游戏结束，还有给出移动的建议。CardInitializaer负责牌的分发，处理每次组件取牌的事件。PointCounter负责管理游戏的分数，统计每次操作后的得分。Timer负责统计游戏开始后的时间。

对于托管在CardManagement下的游戏组件，分为4个suitpiles（4个box），1个deck card堆和1个discard card堆（1个dealer），7个牌堆（7个heap），所有关于牌的具体的位置信息和移动的处理，最终都分发给这些组件来进行处理。

至于游戏的生成，由于初始化较为复杂，涉及到多个组件之间的初始化和构造的先后顺序问题，因此使用一个CardGameFactory来生成游戏。

对于具体的每张纸牌，都是一个Card对象，其中包含了很多与这张卡相关的方法。

对于这个游戏，我们给出了3个难度，2种游戏模式，总共6种游戏。游戏难度分为简单，困难和大师，游戏模式分为一次发1张牌和一次发3张牌。简单难度是基于现在的牌局，给出较好的牌，使得下一次尽可能的能够有牌可以移动；困难难度是基于从已经完成的牌局逆向移动从而得到的牌局，保证一定可解，但是解法可能会很少，就很容易失败；大师难度的发牌是随机的，就是没有任何干预。

有关每个方法的实现，具体请看javadoc，和具体代码的注释。