Hanabi Client: Final Project Report

CMPT 370 - Group F1

Minh Hoang - mph952

Sharjeel Humayun - shh738

Donovan Lavoie - dbl599

Georgi Nikolov - gnn308

Faiz Qureshi - faq588

Jason Wong - jaw845

1. User documentation

* Wanted a jar file in the end with no special instruction in installation (MINH)
* Need config file of ‘hanabisecret.txt’ to expose NSID hashes to Client
  + Needs a way to get the hashes for each NSID instead of getting it while creating a game
* As built requirement: (MINH)
  + No discard, log view
  + GUI changed to console
  + More detailed info from player changed to just Players left due to less info from the server

1. Programmer documentation
   1. Compilation instructions
   2. As built design
      1. Changed UI to use JavaFX instead of swing (Georgi + Shar)
         * Too many problems in swing
         * Alignment problems/overwrite
      2. Changed from JavaFX to console (Georgi? + Shar?)
         * Georgi/Shar needs to draft a fuller explanation of history of the construction of the view leading to the non-functional final (maybe updated view diagram too)
      3. Changed AIController in controller package to AIPlayer in model package with modified function interfaces and inner classes
      4. Added AbstractContainer class to handle the FireworksPile and DiscardPile
      5. HanabiController split up into 4 classes and gained interfaces to support testing (change our design to be testable)
         * MessageReceiver
         * ReceiveController
         * ServerReceiver
         * SendController
      6. Changed getStateForAI() to getAIMove() in HanabiGame
   3. The javadoc URL (Donovan)
      1. Update the Javadoc first
      2. Then use tools/generate Javadoc in IntelliJ
         * Use whole project
   4. Known bugs/Incomplete features/workarounds
      * + Sloppy playMove() impl. in HanabiGame class
        + Hanabi secret only confirmed to work on Windows
        + View not functional (refer to as built design section)
        + Ignoring end of draw pile in HanabiGame
        + Workaround
          1. Socket in serverReceiver can’t be closed due to blocking read

SolutIon: close connection on your side, setting an exception and catch it

* 1. Highlights (JASON)
     1. Used limited subset of maven
        + Used its dependency management tools to coordinate libraries (GSON, JUnit5)
        + Used Surefire plugin to enable running all JUnit tests in 1 go
     2. Used JUnit 5.0
     3. Made Mock Servers, Receivers, and Games to test controller components