System design document for the Challenge Accepted project (SDD) 1.0

Cecilia Edwall
Isabelle Frölich
Johan Gustavsson
Madeleine Appert

Contents

1. Introduction

1.1. Design goals

Our construction of the Challenge Accepted is tightly composed because nothing changed in our game. The majority are hard-coded to make sure that we don't allow letting the game do some improvisation.
For usability see RAD.

1.2. Definitions, acronyms and abbreviations

- GUI, graphical user interface.
- Java, platform independent programming language.
- JRE, the Java Run time Environment. Additional software needed to run an Java application.
- Host, a computer where the game will run.
- Round, one complete game ending in a winner or possible canceled.
- Turn, the turn for each player. The player can only act during his or her turn (roll dices, buy, sell, etc.). Thou, the player can be a□ected during other players turns (i.e. pay to actual player, etc.)
- MVC, a way to partition an application with a GUI into distinct parts avoiding a mixture of GUI-code, application code and data spread all over.
- cha, an abbreviation for Challenge Accepted.

2. System design

2.1. Overview

We use Javas framework, it isn't the best to used when you build games. But we don't have that complicated game so it works for our project.
Every event goes through our eventbus so our model and view aren't connected at all except through the event package.

2.2. Software decomposition

2.2.1. General

The application is decomposed to the following modules (see Figure 1):

- cha, is our main package.
- cha.gui is where all the views is designed.
- cha.domain has all model cod.
- cha.event containing our events, for example, our eventbus is in that package.

We have used MVC to build our program to make it easy to changes in the GUI without make changes in the model.

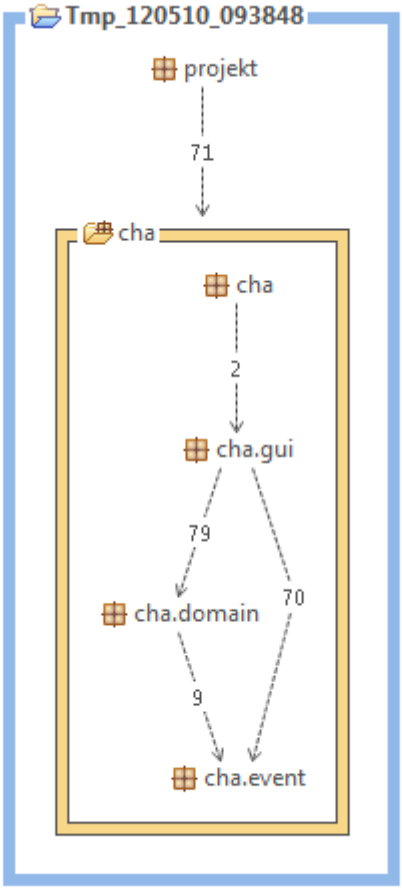

Figure 1

2.2.2. Layering

The layering is indicated in figure 1.

2.2.3. Dependency analysis

See figure 1. As you can see are there no circular dependencies.

2.3. Concurrency issues

NA. It's a single threaded application. Everything will be handled by the Swing event thread. For possible increased response there could be background threads. This will not raise any concurrency issues.

2.4. Persistent data management

NA.

2.5. Access control and security

NA. Our game is way to simpel, so we don't need to have access controll or security.

2.6. Boundary conditions

NA.

3. References

Board game, "upp till bevis": http://www.braspel.com/?id=317
MVC: http://www.braspel.com/?id=317