Requirements and Analysis Document for the Monopoly project (RAD)

Version: 1.0

Date: 2017-02-31

Author: Joachim von Hacht, hajo@chalmers.se

This version overrides all previous versions.

1 Introduction

The project aims to create a prototype for computer based generic version of the well known board game Monopoly by Parker brothers. Generic in the sense that it's should be possible to adapt the game to different locations and more, see further below. For definitions, terms and rules of the game see references.

The application will be a desktop, standalone (non-networked), multi-player application with a graphical user interface for the Windows/Mac/Linux platforms.

Some general characteristics:

- The application will be turn based. The actual player must explicitly end his or her turn. The next player is chosen by the application from a preset ordering. The ordering is generated randomly by the application at start of the round.
- There's no time constraints for a round.
- The application will end according to the rules or possibly be canceled.
- If the game is canceled the player with most resources will be the winner.
- The application will handle all of the bank's responsibilities.
- The application will use a GUI very similar to the original game.
- The application does not include a computer-player. It's impossible to play the game alone (a person can of course choose to play against herself).
- The application does not save interrupted games or collect any statistics (high score or other).

1.1 Definitions, acronyms and abbreviations

Technical definitions

- GUI, graphical user interface.
- Java, platform independent programming language.
- JRE, the Java Runtime Environment. Additional software needed to run an Java application.
- Host, a computer where the game will run.

Most definitions and terms regarding the core Monopoly game are as defined in the references section. Some terms not defined or redefined:

- 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.). Though, the player can be affected during other player's turns (i.e. pay to actual player, etc.)
- Resources (for players), the total value of the properties, buildings and cash of a single player. A player is bankruptcy when he or she has no more resources.

2 Requirements

2.1 Graphical user interface

The user interface will look like the real game. It should be possible later to add perspective or animations etc.

Application will use a fixed (non skinnable, non themeable) GUI following standard conventions. The GUI must take into account different screen sizes, possible very small (minimum size: 320 x 480 (HVGA) at 163 ppi). See APPENDIX for screens and navigational paths.

(just an example)

(Missing comments to GUI here)

2.2 Functional requirements

Functionality for the application includes:

1. Set options:
    a. Select how many player for the game
    b. What color for each player
    c. What piece for each player.
2. Start a new game using the options.
3. Do a turn. During the turn, he or she can
    a. Roll dices. This will possible trigger a response from the application (i.e. move the piece, show a dialog for Chance card, GO to Jail, get money when passing GO, etc.)
        i. NOTE: There's no explicit move command
    b. Respond to any cards etc.
    c. Buy or sell properties and building (or mortgage).
    d. End the turn.
4. Exit the application (will end turn and round).

Ordering of use cases by priority

- Move (this includes roll dices and move piece).
- End turn
- Buy property
- Sell property
- Respond to card
- Start game
- Set options

2.3 Non-functional requirements

Game

The game should be possible to play for at least two different locations (sets of streets etc. example: London and Gothenburg).

Usability is high priority. Normal users should be able to play the game within a very short period.

There should be an English built in user manual, how to play the game.

Technical

The application must be implemented so that the GUI is easily modifiable to suit other platforms (Web, Mobile Apps, Pads, e.t.c.).

The implementation should prepare for the dividing of the application into a client/server-architecture for net based games. It should be easy to partitioning the application into a client-server architecture. A time estimation for this should be included.

There should be automated test verifying all use cases. Code related to the GUI could be tested manually.

To achieve platform independence the application will use the Java environment. All hosts must have the JRE installed and configured. FOr now the application needs to be installed on all hosts where it will run (possibly downloaded).

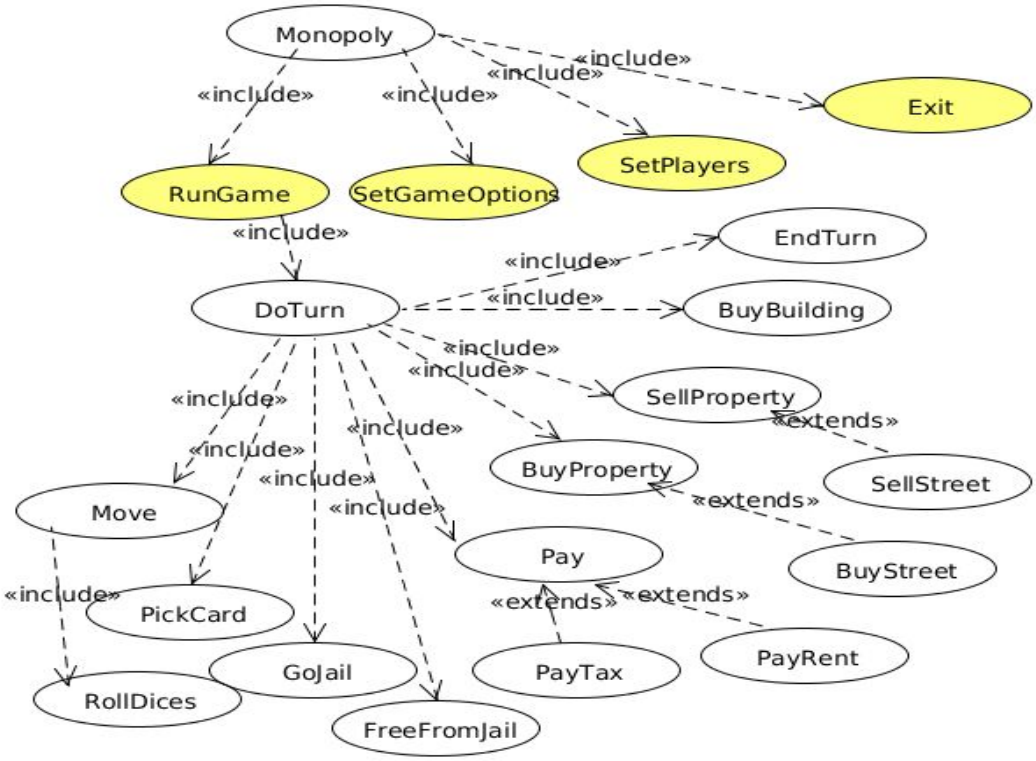
The application will be delivered as an jar-archive containing.

1. A file for the application code (a standard Java jar-file).
2. All needed resources, internationalization and localization files, icons, e.t.c.
3. Start programs (scripts) to start the game on the different platforms.
4. A README-file documenting installation and start of application.

The source code will NOT be part of the package:

3 Use cases

Use case overview



(clarifications/comments to use cases here)

Use Case Listing

1. Move

Summary: The game has started. Actual player is in turn
Priority: High
Extends: DoTurn
Includes: Roll dices
Participators: Actual player

	Actor	**System**
1	Click Roll button	
2		Result for two dices shown Piece removed from actual position and put in new position Roll button disabled

2.1 Passed Go		If player passed go, player balance flashes (updated) and a “cash”-sound is played
2.2 Landed on owned property	See Pay Rent	

2. End turn

….

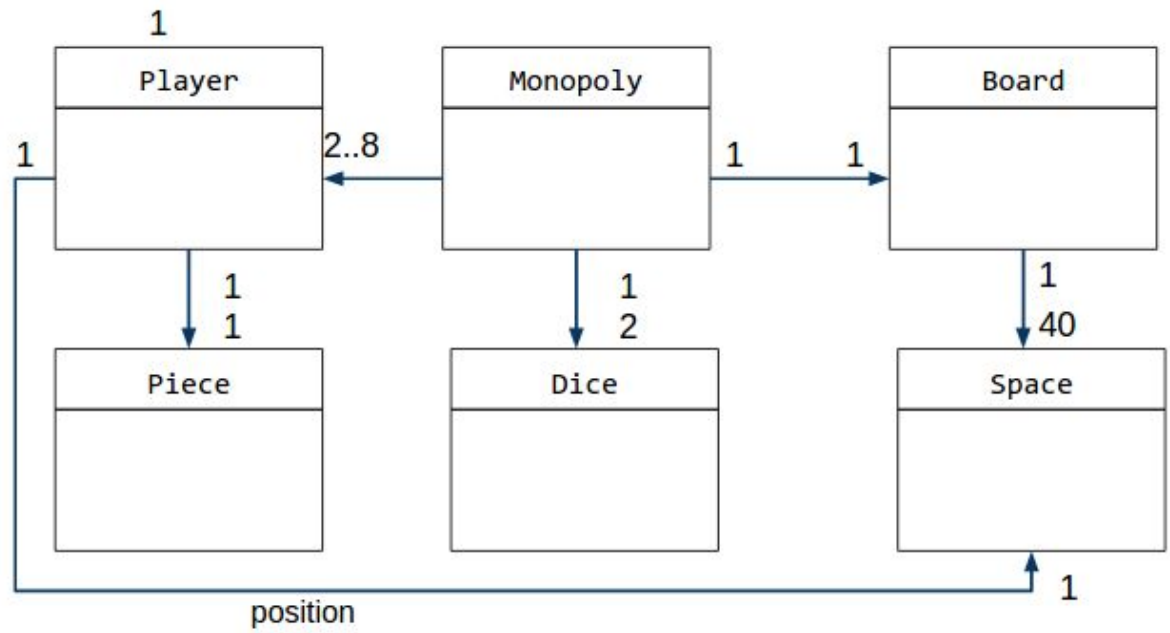
N. Pay rent

Summary: Player have moved piece and landed on property owned by other player
Priority: High
Extends: Roll dices 2.2
Includes:
Participators: Player

	Actor	**System**
1		Shows a dialog
2	Clicks Pay Rent button in dialog	
3		Dialog closes Player and owner balances updated (flashes), “cash”-sound
3.1 No cash	See Sell	
3.2 Broke	See Player Broke	

(more UC texts here)

4 Domain model



4.1 Class responsibilities

Monopoly: Is the overall representation of the game.
Space: Represents a location on the board. May hold visitors (players) and buildings
Board: Is a collections of spaces.
… (more)

5 References

[Monopoly, Wikipedia]
[History of Monopoly]
[Java 8]