Version	Author	Date	Comments
1.0	Mari Rüütli	28.10.2010	Start report in Google Docs
1.1	Mari Rüütli	29.10.2010	Project plan
1.2	Martin Loginov	03.11.2010	Corrected timetable
2.0	Mari Rüütli	05.11.2010	History and rules of Mancala
3.0	Martin Loginov	19.11.2010	Adding class diagram
4.0	Martin Loginov	20.11.2010	Architectural overview
5.0	Peeter Jürviste	25.11.2010	Testing Plan
6.0	Mari Rüütli	30.11.2010	Repository guide

Systems Modelling

Mancala

Project report

Mari Rüütli (A72088)
Peeter Jürviste (A72084)
Martin Loginov (A72092)
Hans Mäesalu (A72107)
Sven Aller (A60197)

1 Project plan
    1.1 Introduction
    1.2 Management description
        1.2.1 Communication
        1.2.2 Division of labor
        1.2.3 Version management
    1.3 Project plan timetable
2 Requirements
    2.1 About Mancala
    2.2 Functional requirements
    2.3 Non-functional requirements
3 Architecture
4 Testing Plan
5 Repository guide

1 Project plan

1.1 Introduction

The aim of this project is to implement the game Mancala using objects first method, Fujaba and design patterns.

1.2 Management description

1.2.1 Communication

Team meetings take place at Liivi 2 on the fourth floor every Thursday at 14.00-15.00. If necessary and possible additional face-to-face meetings will be held. During the meetings the current situation of the project and possible problems will be discussed and new assignments will be dealt out.
For everyday discussions and more urgent topics e-mail, MSN, Skype and Google Docs will be used.

1.2.2 Division of labor

Martin Loginov (A72092) - architecture, programming
Hans Mäesalu (A72107) - model design, architecture, senior Fujaba specialist
Sven Aller (A60197) - requirements analysis, design
Mari Rüütli (A72088) - documentation
Peeter Jürviste (A72084) - testing

1.2.3 Version management

The Git version control system will be used for version management. A central Git repository will be published on github. For creating and updating some of the documents Google Docs will be used.

1.3 Project plan timetable

Time	Action or result	Person
04.11.2010	Creating GIT repository	Sven Aller
09.11.2010	User stories	Sven Aller, Peeter Jürviste
11.11.2010	Object diagrams	Sven Aller

13.11.2010	Use cases	Sven Aller
17.11.2010	Class diagram	Martin Loginov
20.11.2010	Storyboards	Hans Mäesalu
22.11.2010	First implementation	Martin Loginov
22.11.2010	Applying design patterns	Martin Loginov, Hans Mäesalu
23.10.2010	Tests	Hans Mäesalu
27.11.2010	GUI	Sven Aller
28.11.2010	Binding GUI to model in the controller	Peeter Jürviste, Martin Loginov
29.11.2010	User manual, help information	Mari Rüütli
30.11.2010	Final version of Mancala	Martin Loginov
30.11.2010	Testing Plan	Peeter Jürviste
30.11.2010	Report	Mari Rüütli
30.10.2010	Presentation materials	Sven Aller

2 Requirements

2.1 About Mancala

Mancala, also called Kalah or Kalaha, is a game in the mancala family invented by William Julius Champion Jr in 1940.

The game board of Mancala consists of 14 pits - 12 regular pits and 2 stores. Each player controls the six pits and their seeds on his side of the board. Player's score is the number of seeds in his store (the one on the right of his pits).

1. At the beginning of the game, three seeds are placed in each pit.
2. Players take turns *sowing* their seeds. On a turn, the player removes all seeds from one of the pits under his control. Moving counter-clockwise, the player drops one seed in each pit in turn, including the player's own store (but not his opponent's).
3. If the last sown seed lands in the player's store, the player gets an additional move. There is no limit on the number of moves a player can make in his turn.
4. If the last sown seed lands in an empty pit owned by the player, and the opposite pit contains seeds, both the last seed and the opposite seeds are captured and placed into the player's store.
5. When one player no longer has any seeds in any of his pits, the game ends. The other player moves all remaining seeds to his store, and the player with the most seeds in his store wins.

It is possible for the game to end in a draw - each player has 18 seeds.

2.2 Functional requirements

2.2.1 Implement at least a hot seat version of Mancala (using two players).
2.2.2 Implement a history function for keeping scores of specific matches.

For more detailed functional requirements refer to the Use Cases document.

2.3 Non-functional requirements

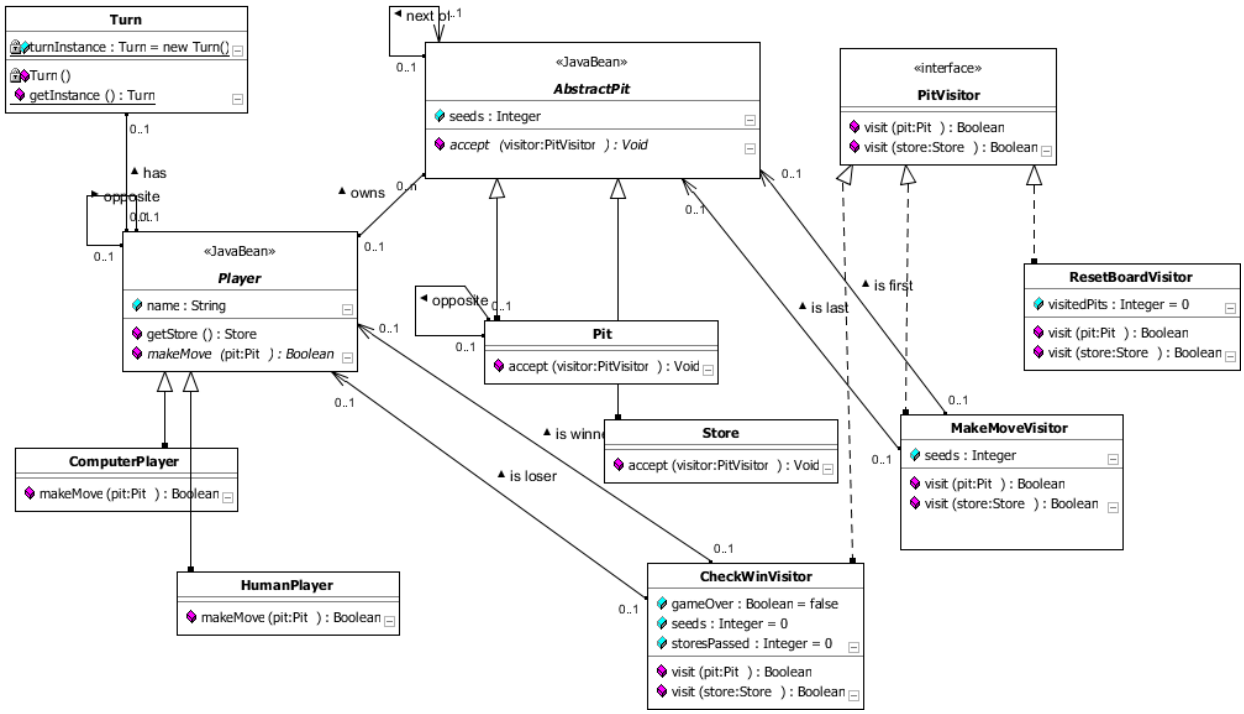
2.3.1 The objects first design method must be used:

1. creating scenarios;
2. deriving user stories;
3. creating object diagrams;
4. sorting object diagrams into usecases;
5. deriving class diagram;
6. using object game or note-pad test to define methods;
7. implementing methods

2.3.2 Fujaba should be used for modelling and implementation as much as possible

2.3.3 Design patterns should be used if possible and reasonable.

3 Architecture



The part of our class diagram that is used to represent the state of the game was directly based on the object diagrams. From there we derived the classes Pit, Store, Player and Turn. Since the Pit and Store class are so similar we extrapolated a common base class for them called AbstractPit.

During a game the imaginary game board consists of 14 AbstractPit instances (12 Pit's and 2 Store's) each of which has a reference to the next AbstractPit object - in this way they form a circle, with the last object referring to the first. Each AbstractPit instance is also connected to a corresponding instance of the Player class, which represents that a pit or store is owned by a specific player.

Between two Player instances an instance of the Turn class is passed back and forth, signifying whose turn is it to make a move on the "game board".

Summary of design patterns used:

- **Singleton**: The Turn class is implemented as a singleton for obvious reasons, since only one player can have the turn at any given time and using the singleton pattern helps us guarantee that.
- **Strategy:** It made sense to implement different player types using the strategy pattern. This way different types of players can be selected at runtime and each of them can have their own logic for making a move (each has their own implementation of the makeMove() method). Currently we have two player types: HumanPlayer and ComputerPlayer, but thanks to the use of the strategy pattern this can easily be extended to maybe add a NetworkPlayer in the future.
- **Visitor:** Using the visitor pattern allowed us to make modifying the state of the imaginary game board very modular and easily extendable. Since AbstractPit accepts a visitor

of type PitVisitor, we can use different types of concrete visitors to implement various changes in the game status. For example to carry out a move the MakeMoveVisitor is used and after each move the CheckWinVisitor is used to check if the last move didn't result in a game ending situation and take appropriate action. The ResetBoardVisitor can be used to reset the state of the imaginary game board.

- **MVC:** The model part consists of the pit and player classes. MancalaGUI and NewGameDialogGUI make up the view part and the controller is implemented as the GameController class. All the main logic is implemented in the controller, it acts as an intermediary between the objects containing the game state and the GUI.

The entire application is implemented in Java. Everything except the GameController and GUI classes was modelled in Fujaba and also implemented using Fujaba activities.

4 Testing Plan

Please also refer to the storyboards in the project.

Test case	Status and comments	Date	Tester
Starting a new game between two human players.	OK, can specify players and start a new human vs human game	30.11.2010	Peeter Jürviste
Making a move in a game against a human opponent	OK, seeds are moved to appropriate consequtive pits	30.11.2010	Peeter Jürviste
Starting a new game against a computer	OK, can specify a player name and game starts	30.11.2010	Peeter Jürviste
Making a move in a game against a computer	OK	30.11.2010	Peeter Jürviste
Testing if computer makes legitimate moves	OK	30.11.2010	Peeter Jürviste
Testing if computer makes clever moves	Fail, computer always chooses the farthest pit not empty and is very predictable	30.11.2010	Peeter Jürviste
Player making a move that ends in her own store	OK, player gets a new turn and one seed is placed in the store	30.11.2010	Hans Mäesalu, Peeter Jürviste
Player making a move that ends in her own empty pit, opposite pit not empty	OK, stealing works	30.11.2010	Hans Mäesalu, Peeter Jürviste
Player making a move that ends in her own empty pit, opposite pit also empty	OK, as required turn to the opponent, no stealing	30.11.2010	Hans Mäesalu, Peeter Jürviste
Draw scenario	OK	30.11.2010	Hans Mäesalu,

			Peeter Jürviste
Winning in Mancala	OK	30.11.2010	Hans Mäesalu, Peeter Jürviste
Writing to high scores table	OK, every round is logged	30.11.2010	Peeter Jürviste
Reading from high scores table	OK, logged information can be displayed	30.11.2010	Peeter Jürviste
Help information in the program	OK, seems sufficient	30.11.2010	Peeter Jürviste
Team information in the program	OK, correct but some encoding problems with Estonian letters	30.11.2010	Peeter Jürviste

Unit test are realised using Story Boards in Fujaba. Unit tests test making move with different parameters.

5 Repository guide

The project repository consists of the following directories:

- Mancala - the root directory of the Eclipse project. It also includes the class diagram and test cases in story boards.
- Documents - all the documents concerning the project
    - user_stories.pdf - the user stories of the game Mancala
    - object_diagrams/ - folder with original files of object diagrams, made by MS VIsio
    - object_diagrams.pdf - the object diagrams in one file
    - use_cases.pdf - the use cases derived from the object diagrams
    - presentation.ppt - materials for the project presentation
    - user_manual.pdf - the user manual introducing the repository and explaining how to download and run the program
    - report.pdf - the project report