Quang Anh Nguyen

Grundlagen mobiler Anwendungen

Wintersemester 2021

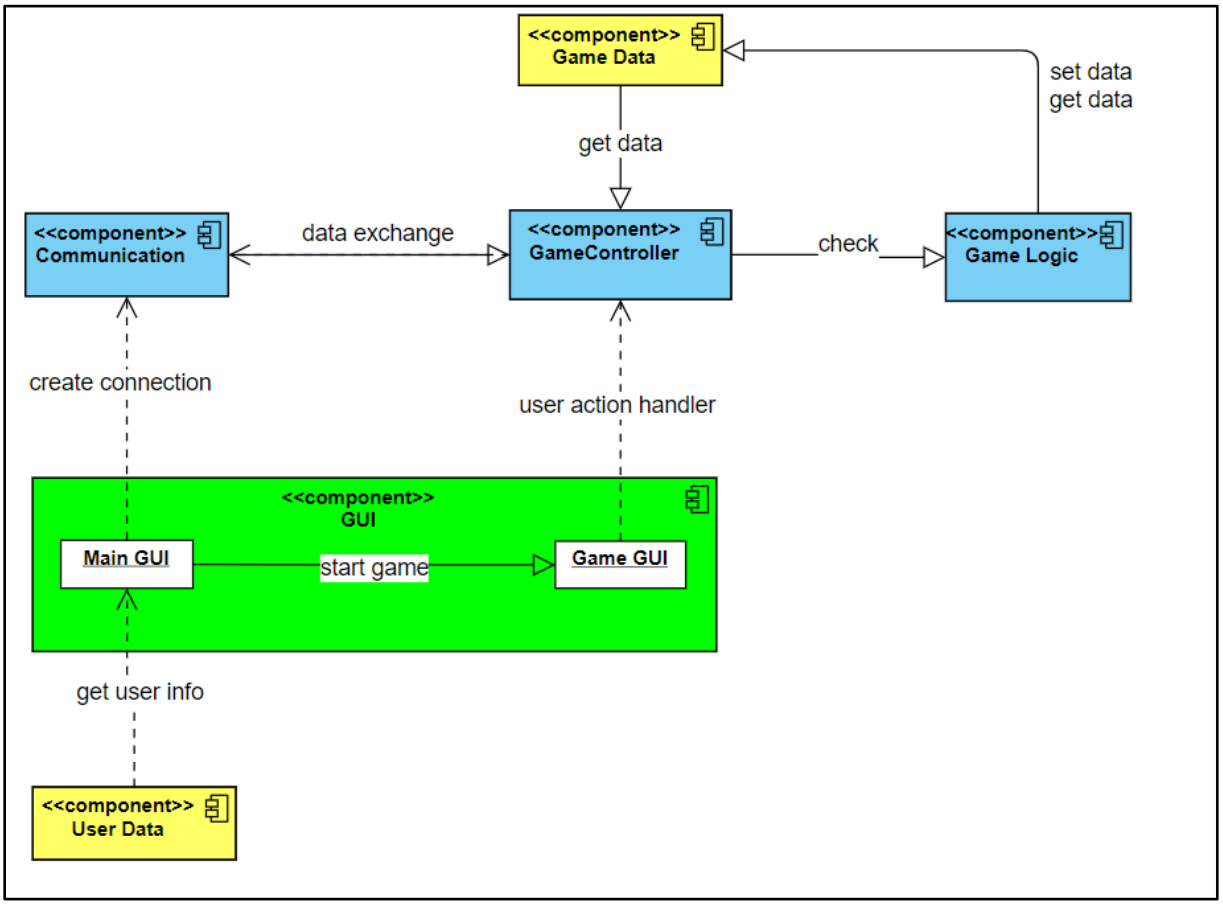
Semesterarbeit: Spiel ,,Vier Gewinnt“

1. Übersicht

Die App ist das Spiel ,,Vier gewinnt“.

Vier gewinnt (englisch: Connect Four oder Captain's mistress) ist ein Zweipersonen-Strategiespiel mit dem Ziel, als Erster vier der eigenen Spielsteine in eine Linie zu bringen.

Die App läuft unter Android Betriebssystem und tauscht die Daten über Bluetooth-Verbindung aus.



2. GUI

2.1 Zusammenfassung

MVC: View

Android-Abhängigkeiten: Ja

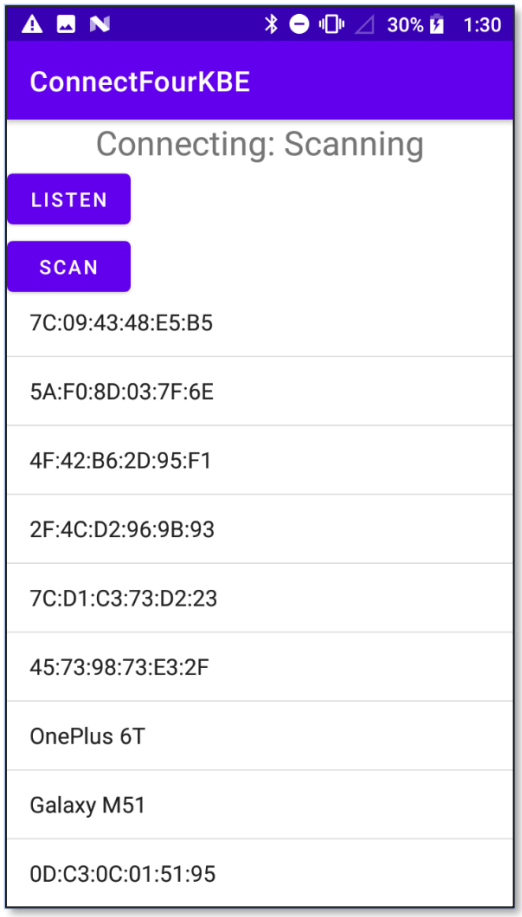
Es gibt 2 GUIs, zu denen jeweils eine Activity gehört.

2.2 Main GUI

2.2.1 Struktur/ Technik

Stellt Benutzeroberfläche mit dem Button „Listen“ und „Scan“ bereit, die die Komponente „Communication“ aufruft, um eine Bluetooth-Verbindung zu herstellen.

Der Nutzer muss hier das Bluetooth einschalten, um die Mitspieler zu entdecken. Den gefundenen Geräten sind durch eine ListView gezeigt. Der Nutzer wählt dann das entsprechende Gerät von dem Mitspieler aus.



2.2.3 Testkonzept für Main GUI

Die GUIs werden in Android Studio mit Hilfe von Espresso Framework getestet.

Diese Art von Tests nennt sich UI Tests.

Die Handlung besteht aus drei Schritten: das Element (View) finden, die Funktion betätigen und das Ergebnis überprüfen.

onView(Matcher<View>).perform(ViewAction).check(ViewAssertion)

MainActivityTest:

	Testfall	Testname	Erfolg
1	Abruf von dem Namen des neuen Spielers. Element: EditText, Button „OK“	testGetName()	Ja
2	Anzeigen von allen Elementen. Element: ListView „listdevices“, Button „scan“, Button „Listen“, Button „status“.	testElementsAreDisplayed()	Ja
3	Start zum Scann beim Klicken von Button „scan“	testScanButton()	Ja
4	Start zum Hören beim Klicken von Button „listen“	testListenButton()	ja

2.3 Game-GUI

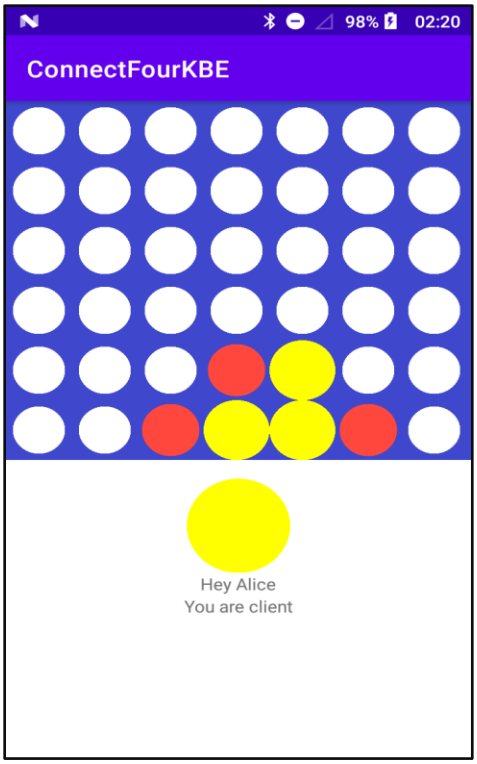
2.3.1 Struktur/ Technik

Ein RecyclerView für das Spielbrett ist ausgewählt.

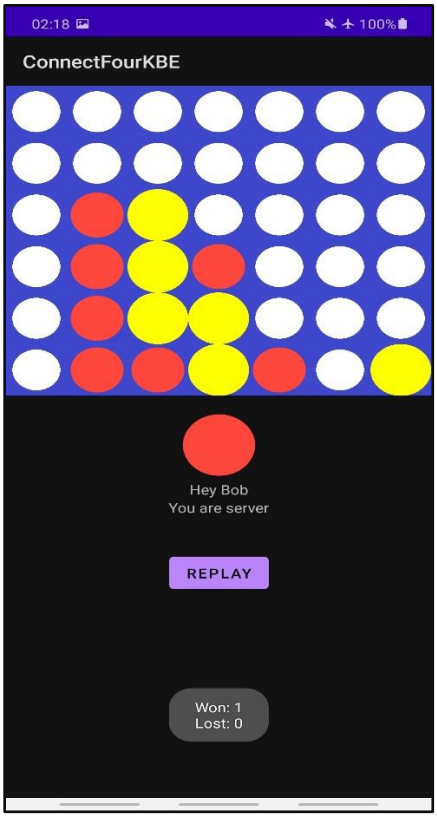
Für die Implementierung der RecyclerView braucht ein Objekt als „BoardColumn.xml“ . In der Datei „BoardColumn.xml“ gibt es 6 ImageViews, die entspricht 6 Kreise in einer Spalte und ein Adapter für das Objekt als ,,BoardColumnAdapter“.

Standardmäßig zeigt die ImageView ein weißes Bild, das später zu rot oder gelb gewechselt wird.

Die RecyclerView hat dann ein Arraylist von 7 Objekten als Column, bedeutet 7 Reihen im Spielbrett.



Nach dem Spielende werden die Spielergebnisse und ein Button „Replay“ erscheinen, um das Spiel wieder zu spielen.



2.3.2 Testkonzept für Game GUI

Ähnlich wie Main GUI wird das Espresso Framework auch genutzt.

	Testfall	Testname	Erfolg
1	Anzeige von allen Elementen. Elemente: RecyclerView, ImageView von Player, TextView „name of player“	testElementsAreDisplayed()	ja
2	Klicken auf die 4. Spalte von dem Spielbrett	testBoardColum4Click()	ja
3	Keine Anzeige von dem Button „replay“	testReplayButton()	ja
4	weißes Bild von ImageView von Spalte 4, Säule 1	testImageViewColumn4Row1()	ja

3. Game Data

3.1 Zusammenfassung

Definition von dem Spielbrett und Speicherung der aktuellen Spieldaten

MVC: Model

Android-Abhängigkeiten: Nein

3.2 Struktur/ Technik

Diese Komponente ist aus reines Java-code geschrieben und wird zur Datei CFcore.jar exportiert.

Board.java: Alle Bewegungen von den Spielern in einem Spiel sind definiert und gespeichert.

3.2 Schnittstelle

```
package GameData;


public interface Board {

/**
Every place on the board is marked;
NONE: not filled,
ONE: filled by the first player,
TWO: filled by the second player,
*/
enum Player {
NONE,
ONE,
TWO
}
```

```
int COLUMNS = 7;
int ROWS = 6;
Player[][] board = new Player[ROWS][COLUMNS];

/**
print the current game board in console
*/
void printBoard();

/**
all places in board are set to NONE to start a new game.
*/
void cleanBoard();

/**
@return the next player.
*/
Player getCurrentPlayer();

/**
set the current player
@param p
*/
void setCurrentPlayer(Player p);

/**
@return the number of the movements.
*/
int getMovesCounter();


/**
set the number of the movements
@param counter
*/
void setMovesCounter(int counter);

/**
get the number of the column by the last movement
@return the number of column
*/
int getLastColumn();

/**
set the number of the column by the last movement
@param column
*/
void setLastColumn(int column);

/**
get the number of the row by the last movement
@return the number of the row
*/
int getLastRow();

/**
set the number of the row by the last movement
@param row
*/
void setLastRow(int row);
```

}

3.4 Testkonzept

Test: Board.java with JUnit 4

Ein Testklasse Boardtest.java werden geschrieben, um die Setter und Getter Funktionen zu überprüfen. Alle Testfunktionen wurden erfolgreich getestet.

4. User Data

4.1 Zusammenfassung

Speicherung von dem Namen des Spielers und den Spielergebnissen.

MVC: Model

Android-Abhängigkeit: Ja

4.2 Struktur/ Technik

User.java: Die Anzahl von den gewonnenen und verlorenen Spielen sind in SharedPreference gespeichert.

Ein Context von der Activity muss übergeben werden, um Funktion von der SharedPreference durchzuführen.

4.3 Schnittstelle

```
package com.quangbruder.connectfourkbe.UserData;

import android.content.Context;
import android.content.SharedPreferences;

public interface User {

    /**
     * @return the name of the user.
     */
    String getName(Context context);
    /**
     * set the name for a user.
     * @param name
     */
    void setName(Context context, String name);

    /**
     * @return the number of victories.
     */
    int getWon(Context context);

    /**
     * plus one win
```

```
     * @param context
     */
    void addWon(Context context);

    /**
     * @return the number of times lost
     */
    int getLost(Context context);

    /**
     * plus one loss.
     * @param context
     */
    void addLost(Context context);
}
```

4.4 Testkonzept

Das Mockito Framework und eine Bibliothek von Ivan Shafran werden benutzt um die SharedPreference zu testen.

	Testfall	Testname	Erfolg
1	Erhalt vom Namen von dem Spieler	getNameTest()	ja
2	Feststellung vom Namen von dem Spieler	setNameTest()	ja
3	Zurückholen von der Anzahl der gewonnenen Spiele	getWonTest()	ja
4	Addieren die Anzahl der gewonnenen Spiele	addWonTest()	ja
5	Zurückholen von der Anzahl der verlorenen Spiele	getLostTest()	ja
6	Addieren die Anzahl der gewonnenen Spiele	addLostTest()	ja

5. Game Logic

5.1 Zusammenfassung

Auflösen die logischen Aufgaben

MVC: Controller

Android-Abhängigkeit: Nein

5.2 Struktur/ Technik

Logic.java: Ruf die Spieldaten ab, zu überprüfen, ob eine Bewegung machbar ist und das Spiel beendet.

Diese Komponente ist aus reines Java-code geschrieben und wird zur Datei CFcore.jar exportiert.

5.3 Schnittstelle

```
package GameLogic;

import GameData.Board;

public interface Logic {

/**
		make multi moves. Used to test.
*
		@param columns
*/
void makeMultiMoves(int[] columns, Board boardGame);

/**
		make a move
*
		@param column_nr
		@return the number of row if successful; -1 if failed.
*/
int makeMove(int column_nr, Board boardGame);

/**
		@param column_nr
		@return true if this column can be filled.
*/
boolean isMovable(int column_nr, Board boardGame);

/**
		@return true, if the player is victory in vertical lines.
*/
boolean isVerticalWin(Board boardGame);

/**
		@return true, if the player is victory in horizontal lines.
*/
boolean isHorizontalWin(Board boardGame);

/**
		@return true, if the player is victory in diogonal lines with
		the upward direction.
*/
boolean isDiogonalUpWin(Board boardGame);

/**
		@return true, if the player is victory in diogonal lines with
		the downward direction.
*/
boolean isDiogonalDownWin(Board boardGame);

/**
		@param p1
		@param p2
		@param p3
```

```
                @param p4
                @return true, if all four places with the same player.
*/
boolean isSamePlayer(Board.Player p1, Board.Player p2, Board.Player p3,
Board.Player p4);

/**
                change to the next player.
*/
void changeCurrentPlayer(Board boardGame);

/**
                All types of victory are checked.
*
                @return true, if a type of victory happens.
*/
boolean isVictory(Board boardGame);

/**
                @return true, if the game is drawn.
*/
boolean isDrawn(Board boardGame);

/**
                @return true, if the game is finished.
*/
boolean isFinished(Board boardGame);


}
```

5.4 Testkonzept

Test: JUnit

	Testfall	Testname	Erfolg
1	Bewegung machen	Makemove()	ja
2	Gewinn in horizontale Linien	isHorizontalWin()	ja
3	Gewinn in vertikale Linien	isVerticalWin()	ja
4	Gewinn in diagonale Linien	isDiogonalUpWin()	Ja
5	Gewinn in diagonale Linien	isDiogonalDownWin()	Ja
6	Änderung des aktuellen Spielers	changeCurrentPlayerTest	Ja

6. Game Controller

6.1 Zusammenfassung

Nach dem Spielbeginn hat die Komponente „GameController“ die Rolle als Hauptsteuerung. Sie bearbeitet die Benutzeraktion, sendet die Daten über die Komponente „Communication“ und kontrolliert die Komponente „GUI“.

MVC: Controller

Android-Abhängigkeit: Ja

6.2 Struktur/ Technik

GameController.java:

```
public class GameController {


    /**
     * movement of the opponent
     * @param position
     * @throws IOException
     */
    public static void replyMove(int position) throws IOException;

    /**
     * click event handler
     * @param gameboard
     * @param player
     * @param holder
     * @param col_nr
     * @throws IOException
     */
    public static void boardClick(Board gameboard, Board.Player player,
BoardColumnsAdapter.ViewHolder holder, int col_nr) throws IOException ;

    /**
     * draw the new circle
     * @param player
     * @param holder
     * @param row
     */
    public static void drawCircle(Board.Player player,
BoardColumnsAdapter.ViewHolder holder, int row);

    /**
     * notify user, if the game finished
     * @param gameboard
     * @param holder
     */
    public static void statusNoti(Board gameboard,
BoardColumnsAdapter.ViewHolder holder);

    /**
     * create user object to get the name of the player
     */

    public void createUser(){
        // Create User
        User user  = new UserImpl();
        activity.tv_nameOfPlayer.setText("Hey "+user.getName(activity));
    }

    /**
     * set the  graphical user interface for the player
     */
    public void setUIforPlayer();

    /**
     * set the RecyclerView for the game board
```

```
     */
    public void setRecyclerView();
}
```

6.3 Testkonzept

Es handelt sich hier schon um den Integrationstest. Denn diese Komponente „GameController“ kommuniziert sich mit allen anderen Komponenten.

	Testfall	Testname	Erfolg
1	Füllen den Kreis(0,0) in Gelb	drawCircleTest()	ja
2	Informiere dem Spieler zum Spielende und Machen das Button „replay“ sichtbar	statusNotiTest()	ja
3	Stellen UI für den Spieler als Server mit rotem Kreis und dem Hinweis „You are server“	setUIforPlayerTest()	ja
4	Handlung des Klickens von Spieler. Die Funktion ist abhängig von der Komponente „Communication“. Nämlich wird die letzte Bewegung zu dem Gegner gesendet.	boardClickTest()	Noch nicht implemtiert

7. Communication

7.1 Zusammenfassung:

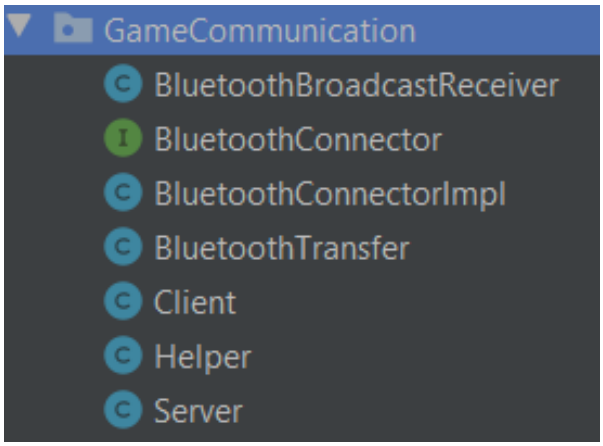
Aufbau der Bluetooth-Verbindung durch Implementierung von Google Bluetooth-API

MVC: Controller

Android-Abhängigkeit: Ja

7.2 Struktur/ Technik

Paketstruktur von der Komponente Communication



7.2.1 Verbindungsaufbau

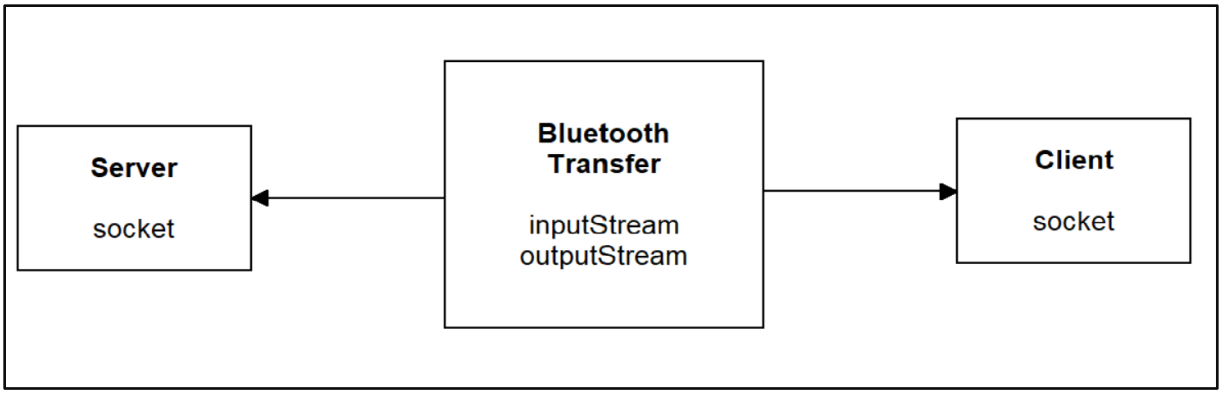
BluetoothConector.java:

Ein Objekt der Klasse BluetoothAdapter von Standard Bluetooth API wird erzeugt.

Mit diesem Objekt werden die Funktionsweisen wie Scannen und Entdecken weiter implementiert.

```
public interface BluetoothConnector {

    /**
     *
     * @return true if bluetooth is enabled
     */
    boolean isEnabled();

    /**
     * request user to turn on bluetooth
     * @param activity
     */
     void requestEnableBluetooth(Activity activity);

    /**
     * set the bluetooth broadcast receiver
     * @param activity
     */
     void setBTReceiver(MainActivity activity);

    /**
     * wait for the connection invitaion
     * @param activity
     */
     void waiting(MainActivity activity);

    /**
     * scan available bluetooth devices
     */
    void scanning();

    /**
     * accept the connection invitation
     * @param activity
     * @param position
     */
     void acceptDevice(MainActivity activity, int position);

    /**
     * make the device as discoverable
     */
    void makeDiscoverable();
   }
```

7.2.2 Datenaustausch



Das Server-Client Modell wird hier benutzt. Der Sender von Verbindungseinladung wird als Client und der Empfänger als Server definiert. Nach dem Verbindungsaufbau besitzen beide Geräten einen Socket erstellt ein Objekt von der Klasse BluetoothTransfer.

In der Klasse werden ein Lesen-Funktion durch die Inputstream und ein Schreiben-Funktion durch die Outputstream implementiert. Diese Streams wurden aus dem Socket genommen.

Die Daten, die von Server zu Client und umgekehrt gesendet werden, sind die Bewegungen von dem Spieler, die von der Komponente „GameController“ übergeben sind.

7.3 Testkonzept: Keine Implementierung.

Die Komponente „Communication“ wird durch die Standard Bluetooth API von Google implementiert, die durch manuelles Testen sehr stabil nachgewiesen ist. Für den Testen wurde leider kein passendes Framework gefunden.

8. Source Code

GitHub Repository:

https://github.com/QuangBruder27/ConnectFourAndroid

Package CFcore(enthält GameLogic und GameData):
https://github.com/QuangBruder27/ConnectFour-Core