Spezifikation

Daniel Riedl

14. November 2013

ComObjects

```
public class ComInitLobby implements ComObject,
    Serializable {
private List<String> playerList;
private Set<GameServerRepresentation> gameList;
...
    public void process(ClientModel model) {
        model.receiveMessage(this);
    }
    public void process(Player player, Server server) {
        server.receiveMessage(player,this);
    }
```

Receive/Send Messages

```
public class ClientModell extends Observable{
...
public void receiveMessage(ComRuleset msg) {
public void receiveMessage(ComInitLobby msg) {}

public void send(ComObject object) {}
...}
public class MessageListenerThread extends Thread {
...
public void run() {
...
object = (ComObject) in.readObject();
object.process(model);
...}
}
```

Ruleset

```
public abstract class ServerRuleset {
private GameServer server;
private GameState gameState;
private GamePhase gamePhase;
...
public void resolveMessage(MsgCard msgCard, String name)
    {}
protected abstract boolean isValidMove(Card card);
protected abstract void calculateRoundOutcome();
```

DummyKlassen

- TestGameServer
- TestLobbyServer
- TestMessageListenerThread
- TestObserver
- TestPlayer

TestPlayer

```
public class TestPlayer extends Player {
...
private List<ComObject> inputComObject;
public List<ComObject> getServerInput() {
return inputComObject;
}
public void injectComObject(ComObject object) {
object.process(this, server);
}
public void send(ComObject com) {
inputComObject.add(com);
}
...
}
}
```

Wizard

Wizard

Bei einem Spiel Wizard wo die erste Karte bereits auf dem Tisch liegt, soll geprüft werden dass nur noch regelkonforme Karten gespielt werden können

Wizard

```
playerState1 = ruleset.getPlayerState(player1);
...
ruleset.setFirstPlayer(playerState1);
ruleset.setTrumpCard(WizardCard.VierRot);

ruleset.giveACard(playerState1, WizardCard.DreiGruen);
ruleset.giveACard(playerState1, WizardCard.ZaubererRot);

ruleset.giveACard(playerState2, WizardCard.ZweiGruen);
ruleset.giveACard(playerState2, WizardCard.DreiRot);
...
```

Wizard

```
@Test
public void testRed3OnGreen3() {
ruleset.playCard(WizardCard.DreiGruen);
ruleset.setCurrentPlayer(playerState2);
assertFalse(ruleset.isValidMove(WizardCard.DreiRot));
}
@Test
public void testGreen2OnGreen3() {
ruleset.playCard(WizardCard.DreiGruen);
ruleset.setCurrentPlayer(playerState2);
assertTrue(ruleset.isValidMove(WizardCard.ZweiGruen);
}
```

Siegerbestimmung

Siegerbestimmung

Bei einem Spiel muss bei Spielende der korrekte Sieger bestimmt werden und an alle Mitspieler weitergeleitet werden.

Siegerbestimmung bei Hearts

```
heartsServerRuleset.addPlayerToGame("Mr. Blue");
  ...
  heartsServerRuleset.setPoints(heartsServerRuleset.
    getPlayerState("Mr. White"),20);
heartsServerRuleset.setPoints(heartsServerRuleset.
    getPlayerState("Mr. Brown"),110);
...
heartsServerRuleset.setGamePhase(GamePhase.Ending);
heartsServerRuleset.calculateRoundOutcome();
assertTrue(heartsServerRuleset.getWinner().equals("Mr.
    White"));
...
```

Siegerbestimmung bei Hearts

```
inputList = blue.getServerInput();
comObject = (ComRuleset) inputList.get(1);
endMsg = (MsgGameEnd) comObject.getRulesetMessage();
winner = endMsg.getWinner();
assertEquals("Nachricht an Blue", "Mr. White", winner);
...
```

Spieler verlässt Spiel

Spieler verlässt Spiel

Wenn ein Spieler ein Spiel verlässt, müssen alle anderen Spieler benachrichtigt werden und zurück in die Lobby gebracht werden.

Spieler verlässt Spiel

```
@Test
public void testPlayerQuitGame() throws IOException{
    player1.changeServer(game);
    assertTrue(game.initLobby().getPlayerList().
        contains(player1.getName()));

    player1.injectComObject(new ComClientQuit());

    assertFalse(lobby.initLobby().getGameList().contains(game));
    assertTrue(lobby.initLobby().getPlayerList().
        contains(player1.getName()));
    assertTrue(lobby.initLobby().getPlayerList().
    contains(player2.getName()));
...
```

Chat

Chat

Nachrichten die vom Client an den Server geschickt werden, müssen an allen anderen Clients die sich im Server befinden ankommen.

ChatModel

```
@Before
    public void setUp() {
        testNetIO = new TestMessageListenerThread();
    testObserver = new TestObserver();
    testMessage = new ComChatMessage("Hello Test!");
    testModel = new ClientModel((MessageListenerThread)
        testNetIO);
    testNetIO.setModel(testModel);
    testModel.addObserver(testObserver);
    }
```

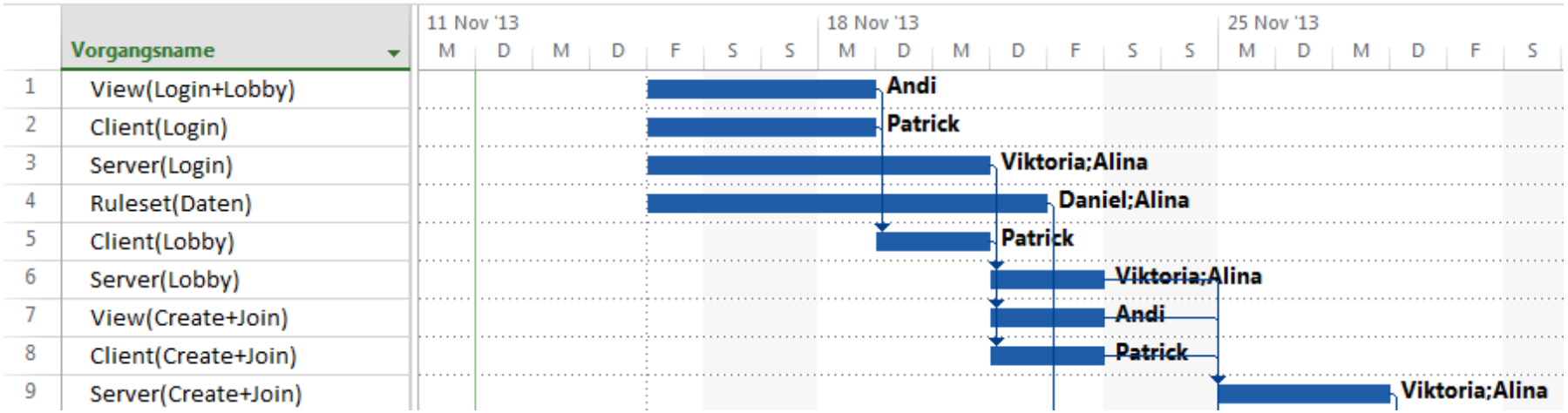
ChatModel

```
@Test
public void testSendChatMessage() {
    String inputText = "Hello Test!";
    testModel.sendChatMessage(inputText);
    testText = ((ComChatMessage)
    testNetIO.getModelInput().get(0)).getChatMessage();
    assertEquals("Vergleich der gesendeten
        Chatnachrichten", testText, inputText);
}
@Test
public void testReceiveChatMessage() {
testNetIO.injectComObject(testMessage);
    assertEquals("Vergleich der empfangenen
        Chatnachrichten", testObserver.getChatMessage(),
        testMessage.getChatMessage());
}
```

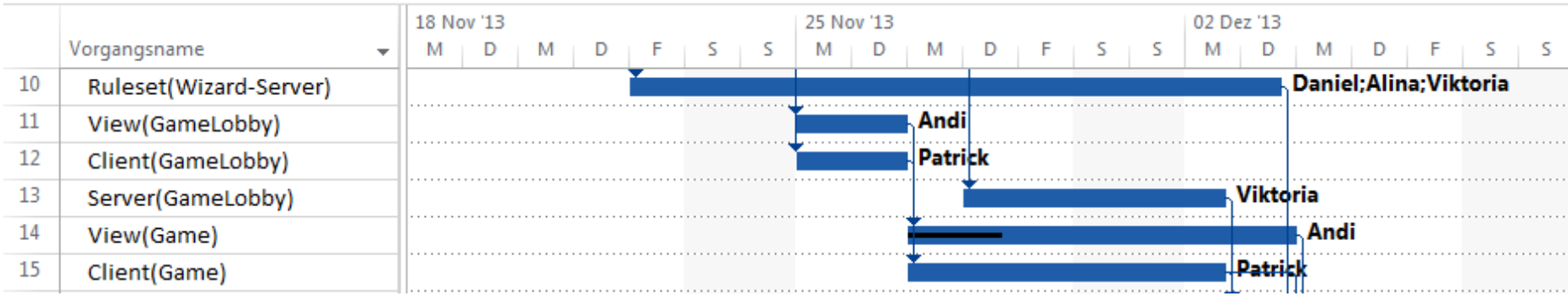
ChatServer

```
@Test
public void testReceiveMessagePlayerComChatMessage() {
    testMessage = new ComChatMessage("Hello Test!");
    String messageToMatch = testMessage.getChatMessage();
    testServer.addPlayer(player1);
    testServer.addPlayer(player2);
    player1.injectComObject(testMessage);
    testText1 = ((ComChatMessage)
        player1.getServerInput().get(0).getChatMessage();
    testText2 = ((ComChatMessage)
        player2.getServerInput().get(0)).getChatMessage();
    assertEquals("Nachricht an Spieler 1",
        messageToMatch, testText1);
    assertEquals("Nachricht an Spieler 2",
        messageToMatch, testText2);
}
```

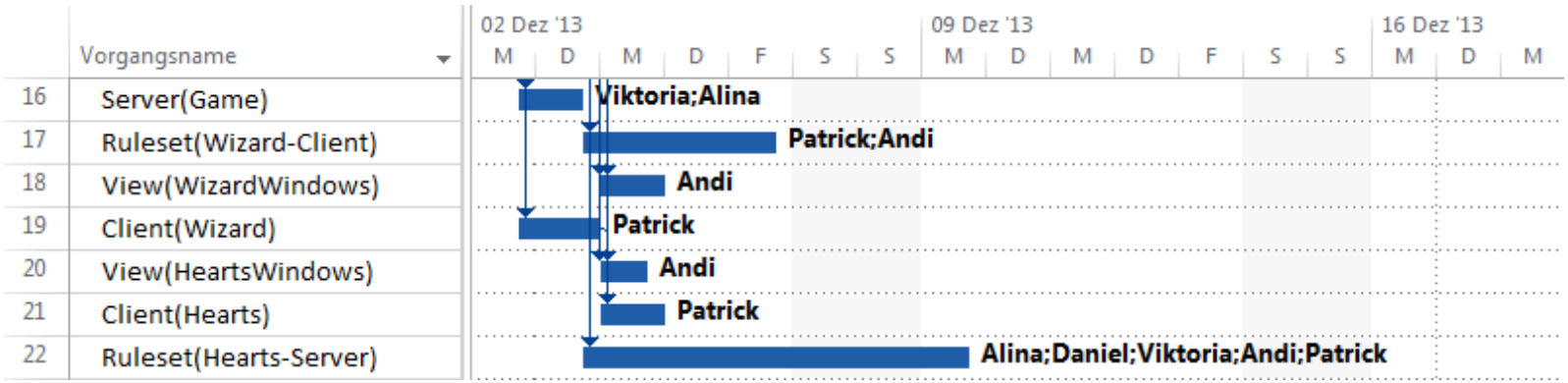
Milestone 1 (27.11.2013)



Milestone 2 (04.12.2013)



Milestone 3 (11.12.2013)



Finale Version (17.12.2013)