GHENT UNIVERSITY

FACULTY OF ENGINEERING AND ARCHITECTURE

Lab 7 : The Grand Finale

4 mei 2018

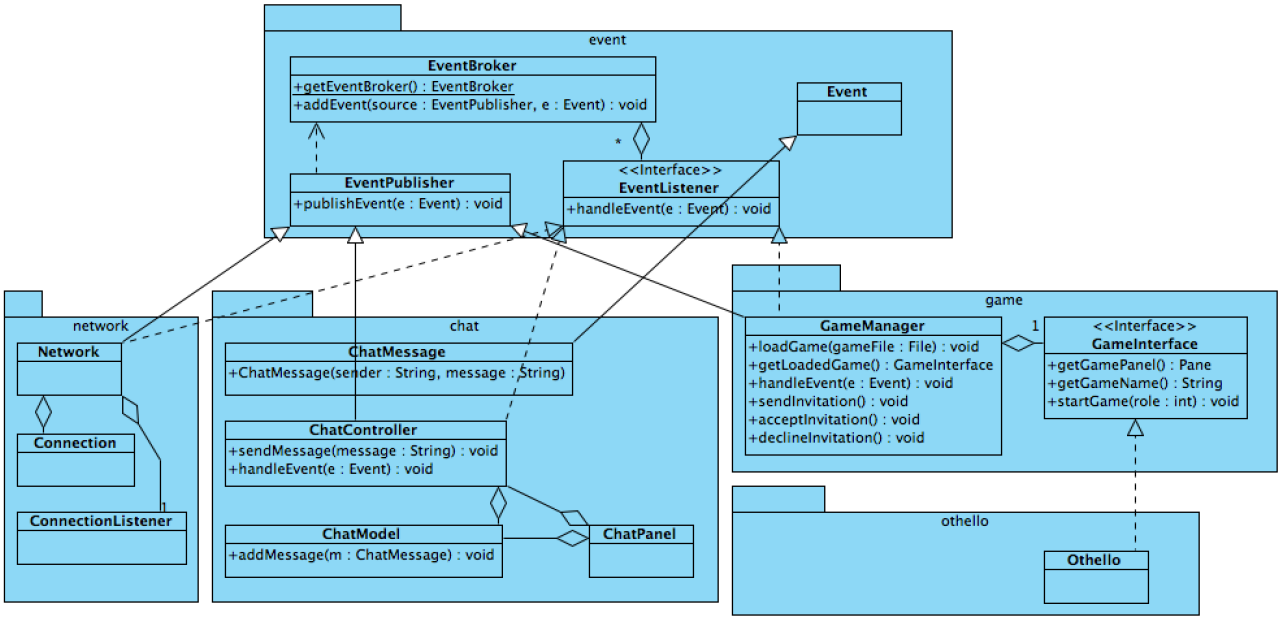
Cedric De Boom
(e-mail: {voornaam.familienaam}@ugent.be)

1 Inleiding

In deze laatste labsessie worden alle stukken code uit de vorige labsessies geïntegreerd tot één applicatie. Twee gebruikers zijn in staat om in te loggen en te chatten met behulp van de chat client uit labsessie 4. Beiden kunnen at runtime een Othellospel inladen, zoals beschreven in labsessie 5. Het Othellospel bestaat uit de spelcomponenten geïmplementeerd in sessie 5 en de spellogica uit labsessie 6. Dit alles steunt op de `EventBroker` uit de eerste sessie, die multithreaded en gedistribueerd gemaakt is in respectievelijk sessies 2 en 3. Deze laatste sessie bestaat er in om de spelers elkaar te laten uitdagen voor een spel Othello, en dit tegen elkaar te spelen over het netwerk. **Let erop dat je twee aparte projecten hebt**: het project `Framework`, dat alle framework code bevat, en het project `Othello`, dat de othello GUI en spellogica bevat, en de .jar voorziet die kan ingeladen worden in het framework.

2 Een overzicht van de applicatie

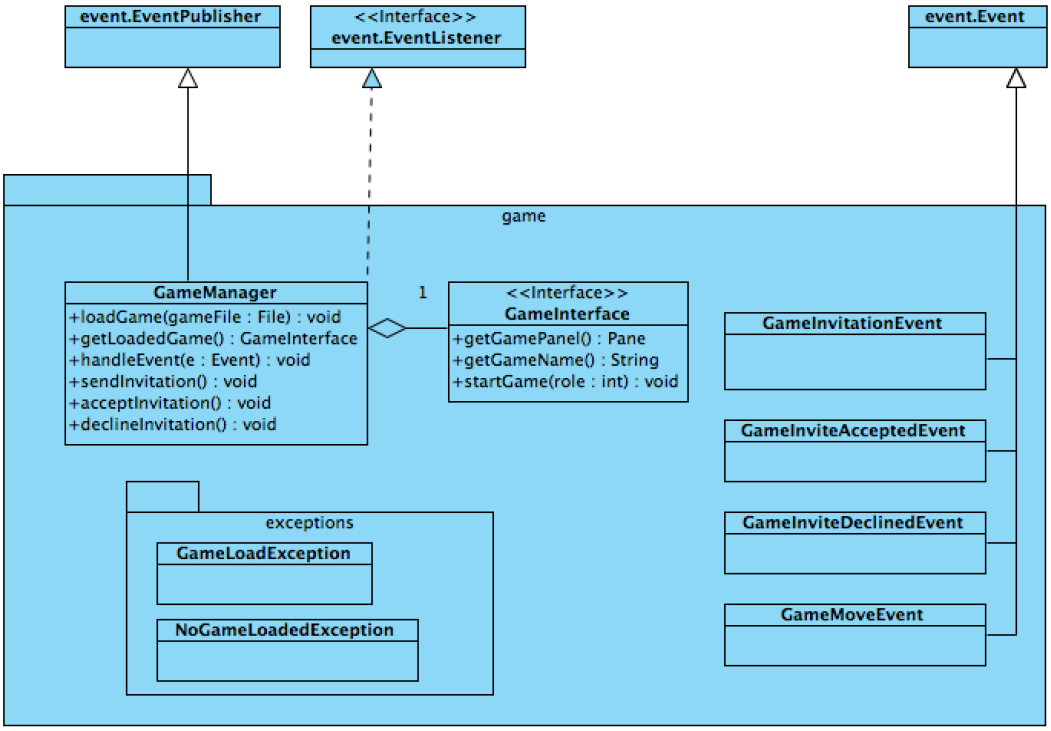
Een zicht op het volledige systeem wordt in Figuur 1 weergegeven, waarbij uiteraard niet alle details kunnen getoond worden. De subsystemen `network`, `chat` en `game` communiceren enkel met het subsysteem `event` en zijn dan ook enkel van dit laatste subsysteem afhankelijk (waardoor het aantal afhankelijkheden beperkt is, wat op zijn beurt de onderhoudbaarheid ten goede komt). In deze opgave worden de subsystemen `network`, `chat` en `event` hergebruikt, terwijl het subsysteem `game` volledig geïmplementeerd dient te worden. Het subsysteem `othello` wordt aangepast om in het spelraamwerk van het subsysteem `game` te passen.



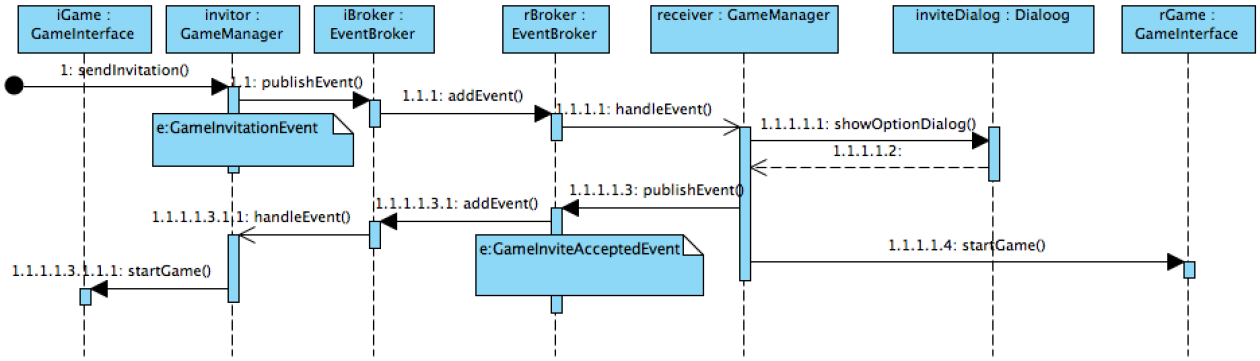
Figuur 1: UML-diagram van het volledige systeem.

3 Elkaar uitdagen

Figuur 2 geeft een klassendiagram van het `game`-subsysteem. Dit subsysteem biedt een abstractie voor bordspellen, waarvan Othello een voorbeeld is. Dit subsysteem bevat logica en gegevens om het spelen van een spel tussen twee personen mogelijk te maken, zonder de details van het spel zelf te bevatten. Een belangrijk deel van deze logica vormt het opzetten van een spelsessie, waarbij de ene speler de andere speler uitnodigt. De logica van deze uitnodiging wordt in het sequentiediagram van Figuur 3 in kaart gebracht, waarbij enkel de succesvolle afloop weergegeven wordt. Indien het spel aan de ontvangstzijde niet beschikbaar is, of indien de ontvanger van de uitnodiging weigert, wordt het spel uiteraard niet opgestart (en krijgen de gebruikers van de toepassing een foutmelding te zien).



Figuur 2: UML-diagram van het `game`-subsysteem.



Figuur 3: Scenario voor het uitnodigen voor een spel.

3.1 Opgave 1

- Breidt de interface `GameInterface` uit met de methoden:
  - `getGameName()` geeft de naam van het spel terug. Deze naam kan gebruikt worden bij het versturen van een uitdaging en om te kijken of het juiste spel is ingeladen alvorens een uitdaging geaccepteerd kan worden.

  - `startGame(int role)` start het spel, en krijgt een `role` variabele mee met de rol die de speler heeft in het spel (i.e. -1 voor de zwarte speler en 1 voor de witte speler).

- Maak de event-klassen `game.events.GameInvitationEvent`, `game.events.GameInviteAcceptedEvent`, `game.events.GameInviteDeclinedEvent` en `game.events.GameMoveEvent`, die een zet in het spel weergeeft.
- Maak de exception-klassen `game.exceptions.GameLoadException` (een instantie van deze klasse wordt opgegooid wanneer er een fout is opgetreden bij het inladen van een spel) en `game.exceptions.NoGameLoadedException` (een instantie van deze klasse wordt opgegooid wanneer men een spel wil starten terwijl er nog geen is ingeladen).
- Maak de klasse `GameManager` die overerft van `EventPublisher` en `EventListener`. Deze houdt het (eventueel) ingeladen spel bij en is verantwoordelijk voor:
  - Het inladen van een spel m.b.v. de methode `loadGame(File gameFile)`, en een getter-methode om het ingeladen spel op te vragen.
  - Het afhandelen van een binnenkomend event dat te maken heeft met de uitdaging, en de juiste acties ondernemen/dialoogvensters tonen.
  - Het sturen (`sendInvitation()`), accepteren (`acceptInvitation()`) of weigeren (`declineInvitation()`) van een uitdaging. Deze methoden worden opgeroepen na klikken op het menu-item “Start game” of na antwoord op een van de dialoogvensters.

4 Othello aanpassen

4.1 Opgave 2

- Voeg een veld `role` toe aan de `OthelloController`, dat weergeeft welke speler deze controller bedient.
- Breidt de klasse `Othello` uit labsessie 5 (die `GameInterface` implementeert) uit met de nieuwe methoden geïntroduceerd in opgave 1. Hierbij zorg je dat bij de methode `startGame()` het speelveld correct geïnitialiseerd wordt en de rol van de speler gezet wordt.
- Aangezien een zet van de ene speler ook moet doorgegeven worden aan de andere speler zal OthelloController moeten luisteren naar `GameMoveEvent`s en deze zelf ook versturen. Laat `OthelloController` overerven van `EventPublisher` en ook de `EventListener` interface implementeren (let erop dat je hem ook ergens registreert bij de `EventBroker`). In Eclipse voeg je hiertoe het Framework-project toe aan het build path van je Othello-project, zodat het eventbroker-package zichtbaar wordt. De `OthelloController` moet nu:
  - Enkel wanneer `role` gelijk is aan degene die aan de beurt is (`getTurn()`) mag een zet als geldig gezien worden.
  - Wanneer een zet uitgevoerd wordt (`doMove()`) dient dit gepubliceerd te worden als `GameMoveEvent`.
  - Wanneer er een `GameMoveEvent` binnenkomt, dient deze zet uitgevoerd te worden op het model. Let wel, deze zet is van de andere speler, dus hier kan een lichtjes ander gedrag gewenst zijn (deze zet moet bv. niet opnieuw gepubliceerd te worden).
  - Na een zet check je of er nog zetten mogelijk zijn. Indien niet toon je een dialoogvenster met de uitslag van het spel, waarna het bord terug op de beginsituatie gezet wordt.

  Pas de spellogica in `OthelloController` waar nodig aan.

5 De finale test

Om te controleren of de applicatie correct werkt, kan je volgende test uitvoeren:

- Start een eerste instantie van het framework. Log in als <naam student 1> en poortnummer 1024.
- Start een tweede instantie van het framework. Log in als <naam student 2> en poortnummer 1025.
- In de tweede instantie, kies in het menu “Connect”, en connecteer naar 127.0.0.1:1024.

- In het chatveld van de tweede instantie, typ “Hallo!”. Deze boodschap dient nu te verschijnen in beide chatvensters.
- In het chatveld van de eerste instantie, typ “Hey!”. Deze boodschap dient nu te verschijnen in beide chatvensters.
- In het menu van de eerste instantie, selecteer “Start game”. Een dialoogvenster wordt getoond dat er nog geen spel ingeladen is.
- In het menu van de eerste instantie, selecteer “Load game”. Selecteer nu de Othello .jar. Het spel is nu ingeladen en het speelveld wordt zichtbaar. Je kan echter nog geen zet doen.
- In het menu van de eerste instantie, selecteer “Start game”. De tweede instantie geeft nu het dialoogvenster dat er een uitdaging is toegekomen. Klik op “No”. De eerste instantie geeft aan dat de andere speler geweigerd heeft.
- In het menu van de eerste instantie, selecteer opnieuw “Start game”. De tweede instantie geeft terug het dialoogvenster dat er een uitdaging is toegekomen. Klik nu op “Yes”. Er verschijnt nu een dialoogvenster dat aangeeft dat niet geaccepteerd kan worden, aangezien het juiste spel niet is ingeladen. De eerste instantie geeft aan dat de andere speler geweigerd heeft.
- Laad nu ook het Othello spel in in de tweede instantie. Selecteer in de eerste instantie opnieuw “Start game” en accepteer nu de uitdaging in de tweede instantie. Het spel is nu gestart, enkel de eerste instantie (de uitdager is zwart) kan nu een zet doen.
- Speel een Othello spel uit door afwisselend de witte en de zwarte speler een zet te laten doen. Wanneer er geen zetten meer mogelijk zijn verschijnt er bij beide instanties een dialoogvenster dat aangeeft wie gewonnen is. Wanneer dit venster wordt weggeklikt wordt het speelveld teruggebracht naar de beginsituatie en kan opnieuw gespeeld worden.

Proficiat, je hebt een volledig werkende applicatie gebouwd! Om echt gedistribueerd te testen kan je eventueel vragen aan een andere groep om met elkaar te connecteren over het netwerk in de PC-klas (al doe je dit best via een lokaal netwerk verschillend van eduroam). Veel Othello-speelplezier!

6 Tot slot

Upload je bestanden naar Indianio. Het is de bedoeling een zip-file in te dienen, **die de twee projecten bevat**, namelijk de projecten `Othello` en `Framework`. We verwachten in deze zip-file dus **minstens** de bestanden:

- Othello\src\othello\OthelloModel.java
- Framework\src\main\Main.java

Zorg ervoor dat de upload **alle bestanden bevat zodat de volledige toepassing kan uitgevoerd worden**. Indien de toepassing niet volledig afgewerkt raakt tijdens labsessie 7, dan krijg je de gelegenheid een finale versie te uploaden in Indianio, **voor vrijdag 18 mei 2018**.