2016

Verslag opdracht 4

VICTOR VAN DE RIET & ROB LOGTENBERG

Inhoud

Aanpak

Nadat we eerst de opdracht hebben bestudeerd hebben we een paar dingen vastgesteld: er komt voor elk deel van de ring een agent (agent 1e ring noord, agent 2e ring zuid, enz). Ook moest er een Main class komen waarin we het systeem aanroepen. Later in de tijd hebben besloten dat we voor een vak, een rij en een stoel een aparte class maken. Ook kregen we als tip om een soort van configuratie class te maken waarin we bijvoorbeeld aantal stoelen of rijen gemakkelijk konden wijzigen.

Ontwerp

Over het ontwerp hebben we lang gediscussieerd en veel geprobeerd maar het werd het telkens niet bijvoorbeeld: we konden een stoel reserveren maar dan alleen voor 1 vak welke stoelnummers had van 1 tot en met 10. Maar elk vak had ook rijen dit werd ingewikkeld en hebben weer wat anders bedacht.

We pakken een paar belangrijke classes eruit:

Reservation class

Hierin wordt bijgehouden wie, voor welke stoel in welke rij en in welk vak, de reservatie heeft gemaakt. En ook wat er al gereserveerd is.

Section class

Deze is vooral belangrijk vanwege zijn eigen print methode. Zie volgende voorbeeld:

```
S<[  0][  1][  2][  3]>
0<[-33][-33][-50][-62]>
1<[--5][--5][-74][121]>
2<[-38][-38][-38][XXX]>
3<[-92][-92][-92][XXX]>
4<[-75][-75][-75][XXX]>
```

S betekent welk stoelnummer, en de 0 tot en met 4 zijn de rij nummers. Het getal tussen de blokhaken ([ ]) zijn de id's van de gene die de stoelen gereserveerd en betaald hebben. Het "-" is alleen gebruikt zodat je een betere uitlijning hebt.

DistributionAgent class

Hierin worden de verzoeken afgehandeld en naar de juiste section agent gestuurd. Mocht de section agent melden dat een vak al vol is dan verteld hij dat aan de DistributionAgent class en die verteld het weer door aan de klant. Een soort tussenpersoon dus.

Testen

We maken gebruik van asserties om te verzekeren dat iets niet gebeurt. Zie de volgende asserties:

1. `assert myReservation != null : "given reservation is null";` Deze assertie komt uit de class CustomerAgent en verzekerd dat `myReservation` niet null mag zijn. Omdat je dan verder in de methode een toString() aanroept op een null object en dit kan niet.
2. `assert request.getNrOfTickets() <= CustomerAgent.MAX_NUMBER_OF_TICKETS_PP: "too many tickets requested";` Bij deze assertie zorgen we er niet meer dan `MAX_NUMBER_OF_TICKETS_PP` worden besteld. Want in de opdracht stond dat er maar maximaal 4 kaarten per persoon gereserveerd/gekocht mochten worden. Deze assertie komt uit de DistributionAgent class.

We hebben ook de volgende tests gedaan:

Test 1: geen klanten

Wanneer er geen klanten zijn blijft het systeem draaien en zal er ook geen output zijn.

Test 2: 200 klanten 1 rij 4 stoelen

Iedereen doet eerst een request om te reserveren maar hij zal alleen de eerste behandelen, de rest een bericht geven dan er geen ruimte meer is.

Test 4: 10 klanten 0 rijen 5 stoelen

De 10 klanten doen een request maar het systeem zal niks reserveren omdat er ook geen rijen zijn.

Test 5: 100 procent kans dat een reservering geannuleerd wordt

Geen enkele stoel wordt uiteindelijk gereserveerd want alle klanten hebben de reservering geannuleerd.