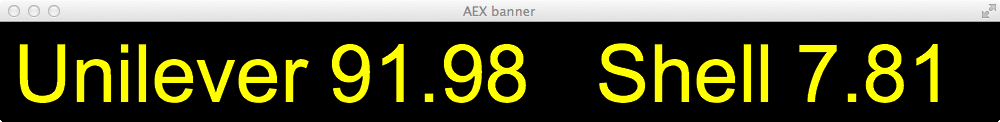
### GSO3 Opdracht AEX Banner

In deze opgave ga je een gedistribueerde applicatie bouwen met de Java techniek Remote Method Invocation (RMI). De bedoeling van de applicatie is om de actuele beurskoers informatie op een AEX-banner weer te geven.



De beurskoers informatie moet van rechts naar links voorbij schuiven over het scherm.

De koersinformatie moet steeds geactualiseerd worden. Deze informatie moet verstrekt worden door de effectenbeurs. Er zijn twee strategieën om de koersinformatie geactualiseerd te houden, namelijk:

* Client Pull

De client vraagt periodiek de actuele koers informatie op bij de server.

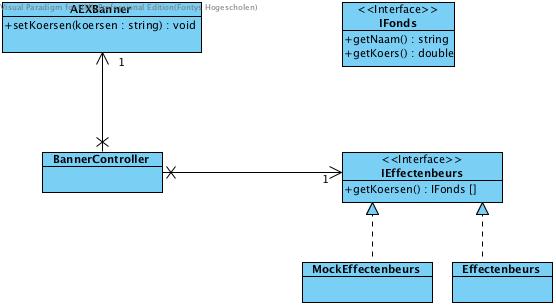
* Server Push

De server geeft elke verandering in de actuele koers informatie direct door aan de hierop geabonneerde cliënten.

Zowel de Client Pull als de Server Push strategie moet je gaan toepassen op de AEX-banner casus.

**Opdracht week 6**

Maak de AEX-Banner applicatie die koersinformatie actueel houdt door gebruik te maken van de Polling strategie. Hierbij moet je uitgaan van het onderstaand ontwerp.



De effectenbeurs beheert de actuele koersinformatie van verschillende beursgenoteerde bedrijven, in objecten van een klasse die voldoen aan de interface IFonds. We simuleren deze effectenbeurs m.b.v. een mock. De fluctuerende aandeelkoersen moeten in de klasse MockEffectenbeurs door een random getallengenerator gesimuleerd worden. De Effectenbeurs-klasse representeert de werkelijke beurs, die dus door de mock-klasse gesimuleerd wordt. De Effectenbeurs-klasse zelf hoeft niet gebouwd te worden. Periodiek vraagt de BannerController klasse de actuele koersinformatie op bij het object dat voldoet aan de interface IEffectenbeurs. Voor de animatie en weergave van de aandeelkoersen is de AexBanner-klasse verantwoordelijk. AexBanner erft over van javafx.application.Application. Binnen AEXBanner zul je i.v.m. de animatie een timer moeten activeren die regelmatig een relocate van een label uitvoert. Binnen MockEffectenbeurs zul je i.v.m. de fluctuerende koersen ook een timer moeten definiëren.

**Opdrachten week 7**

In de voorafgaande opgave heb je een stand-alone versie van de AEX-banner applicatie ontwikkeld. Deze week ga je een gedistribueerde variant maken op basis van het Client Pull principe.

1. Bestudeer het ontwerp van de RMI StudentAdministration zoals beschreven in RMIStudentAdministration.docx
2. Stel een componentendiagram op in Visual Paradigm.
3. Stel een klassendiagram op in Visual Paradigm zodat de RMI-techniek tot uitdrukking komt in het model. Hiertoe moet je van de klasse UnicastRemoteObject en de interfaces Remote en Serializable gebruik maken.
4. Pas RMI toe om de gedistribueerde applicatie binnen je eigen computer tot stand te brengen.
5. Laat het componentdiagram en het klassendiagram aan de docent zien en demonstreer de applicatie.

**Opdrachten week 8**

In de voorafgaande opgave heb je de AEX-banner applicatie ontwikkeld die periodiek de actuele koersinformatie bij de effectenbeurs opvraagt, ook als de effectenkoersen niet zijn veranderd. De andere strategie om de beurskoersen actueel te houden is dat de effectenbeurs elke verandering zelf doorgeeft aan de AEX-banner cliënten. Dit is de Server Push strategie.

1. Stel een componentendiagram op in Visual Paradigm waarbij de communicatie tussen de componenten gebaseerd is op de Server Push strategie.
2. Stel een klassendiagram op in Visual Paradigm zodat de Server Push strategie tot uitdrukking komt in het model. Hiertoe moet je gebruik maken van de klasse BasicPublisher en de interfaces RemotePublisher en RemotePropertyListener (zie FontysObserverJavadoc.zip). Geef in het klassendiagram aan hoe je gebruik maakt van de klasse UnicastRemoteObject en de interfaces Remote en Serializable.
3. Codeer de AEX-banner applicatie en pas RMI toe om de remote communicatie tot stand te brengen. Op sharepoint vind je de broncode van de FontysObserver die je nodig hebt voor de opdracht.
4. Laat het componentdiagram en het klassendiagram aan de docent zien en demonstreer de applicatie.