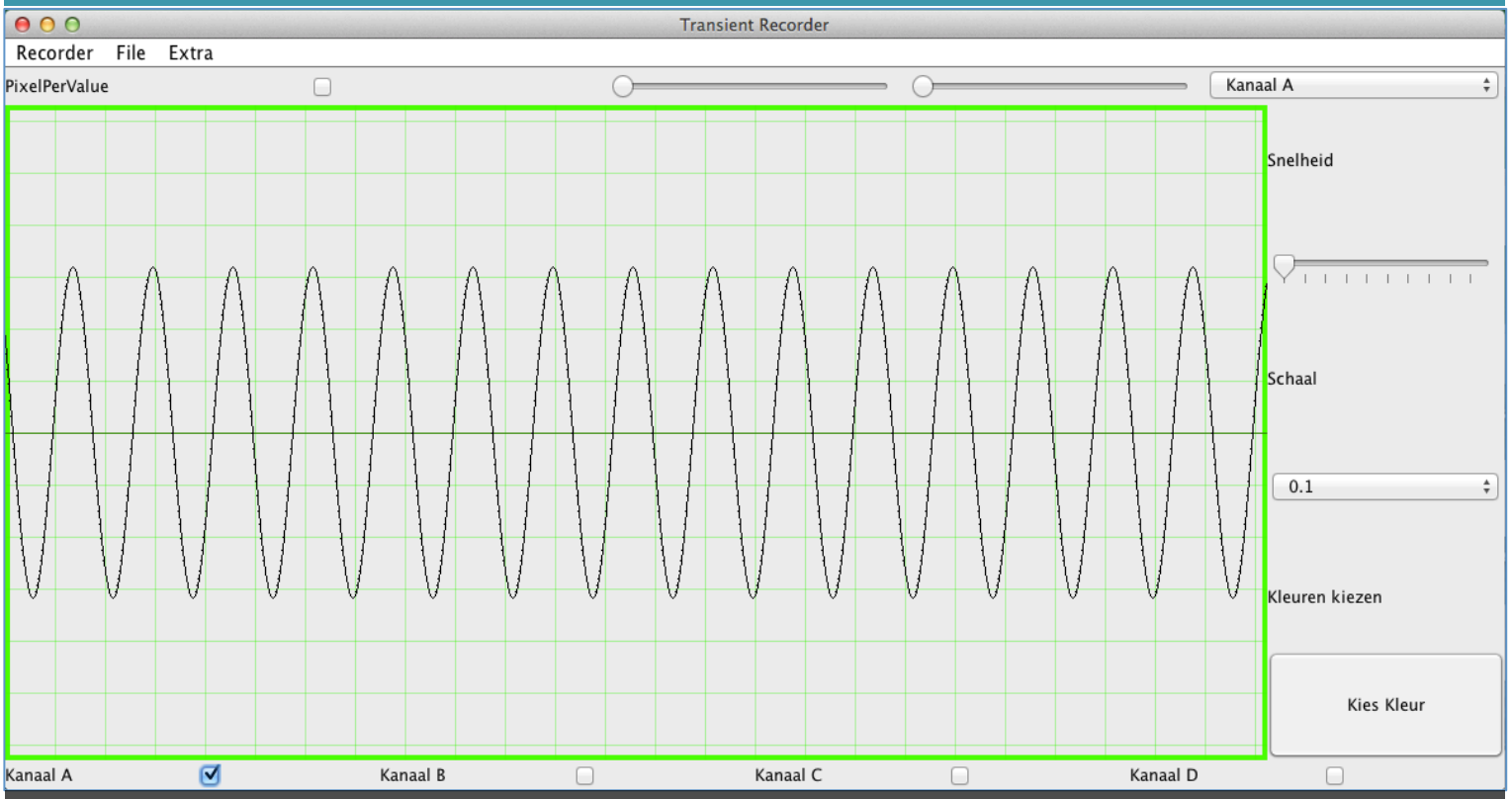


Transient Recorder

Martijn de Vries en Rob Bonhof

Begeleiders: Christiaan Slot en Andre Fiselier

Java User Interfaces Saxion 2014

Inhoudsopgave

H1. Samenvatting

Dit verslag richt zich op het proces voor het maken van een transient recorder. Het ontwikkelproces wordt verder belicht, waar wordt ingegaan op zowel de voorbereidings- als ontwikkelfase. In de voorbereidingsfase zijn de specificaties opgesteld en op basis hiervan zijn enkele UML diagrammen tot stand gekomen, onder meer een UseCase- en klassendiagram, dit kan beschouwd worden als een ontwerpdocument. Daarnaast wordt van de ontwikkelfase de gemaakte keuzes onderbouwt en eventuele aanpassingen besproken ten opzichte van het originele ontwerp. Dit alles aan de hand van het MVC ontwerp patroon waar gegevens, eventhandling en presentatie van elkaar gescheiden zijn. De controller kent zowel het model als view en meldt deze laatste daar dan ook bij aan. Vervolgens wordt via de update methode, die wordt gegeven door de observer interface die op hun beurt door de view geïmplementeerd wordt, de gegevens van het model aangeleverd. Dit gebeurt telkens opnieuw op het moment er een gegevens verandert is in dit model

H2. Inleiding

Aanleiding voor dit verslag is het vak Java User Interfaces waar in practicum vorm gewerkt wordt aan het maken van user interfaces voor het aansturen van Java programma's. Doel hiervan is dat het team vertrouwt raakt met het maken van user interfaces en deze in sommige gevallen weet te implementeren volgens het MVC model. Ook het juist documenteren hiervan, te zien in de vorm van dit document, is een doel hiervan. Dit document spits zich op de eindopdracht van dit vak, namelijk het programmeren en documenteren van een transient recorder. Aan de hand van een aantal verplichte en optionele specificaties is deze recorder tot stand gekomen. Een van de hoofdpunten hierbij is dat dit gedaan wordt aan de hand van het eerder genoemde MVC ontwerp patroon.

H3. Analyse

Voor de analyse is er gebruikt gemaakt van de opdrachtbeschrijving zoals deze is terug te vinden is in de modulehandleiding. Aan de hand van het registreren van zelfstandige naamwoorden en werkwoorden is er geprobeerd een zo'n compleet mogelijk beeld te vormen. Met de zelfstandige naamwoorden is gepoogd de meeste klassen en attributen te ontdekken. Daarnaast is met de analyse van de werkwoorden geprobeerd om de methoden boven water te halen. Dit resulterend in de twee tabellen waarin dit wordt weergegeven inclusief de aantallen en een korte omschrijving. In deze korte omschrijving wordt aangegeven of dit woord bruikbaar is om te implementeren in de recorder. Deze tabellen zijn terug te vinden in de bijlage.

H3.1. Specificaties

Na de uitkomsten van deze analyse zijn vervolgens de specificaties opgesteld waar de recorder aan moet gaan voldoen. Hierbij is rekening gehouden met welke specificaties haalbaar zijn voor dit team. De specificaties zijn terug te vinden op de volgende pagina rekening houdende dat volgens het MVC-model gewerkt dient te worden.

Allereerst dient er een recorderklasse te komen waar alle beschikbare kanalen in kunnen worden opgeborgen. Daarnaast komen hier de algemene waarden in die voor elk kanaal gelden. Dit zijn de pixel-per-value opties maar ook de minimum- en maximum waarden die aangeven of binnen een bepaald gebied de gegevens wel of niet dienen te worden opgeslagen.

Er moet een kanaalklasse komen waarde gemeten waardes per kanaal in worden opgeslagen. Ook moet in deze klasse de waarden van de opties terug te vinden zijn. De opties zijn, snelheid, gevoeligheid, kleur en zichtbaarheid.
De gemeten waardes voor elk kanaal dienen te worden aangeleverd als MeasuredValue object. Deze klasse moet de waarde inclusief de tijd deze gegenereerd is bij zich dragen.

Er moet een view komen voor het weergeven van de grafiek, deze moet onder meer een raster vakken met 'divisions' van 1cm wat ongeveer overeen komt met vijftig bij vijftig pixels. Ook kan er een minimum- en maximumwaarde getekend worden, die het opslaan en dus ook tekenen van de waardes pas toestaan als deze hierboven uitkomt.

Deze waarden die getekend worden moeten ook in een ander venster tekstueel kunnen worden weergegeven. Dit voor het uitlezen van deze waarden om zo eventuele fouten te kunnen ontdekken. Het venster moet geopend kunnen worden uit een menu, waar hier hieronder verder wordt ingegaan.

In het hoofdframe zal er een menu te vinden zijn, deze handelt enkele extra opties af zoals het opslaan van een CSV-bestand. Deze optie moet ook worden geactiveerd op het moment het venster wordt afgesloten. Ook kan er een screenshot mee gemaakt worden.

Een panel moet zorgen voor de zichtbaarheid van elk kanaal in de grafiek. Hier moet kunnen worden aangegeven of een kanaal getekend wordt in het raster van de grafiek.

Daarnaast komt er een panel met alle opties voor het aanpassen van de grafiek. Zo'n panel moet uitwisselbaar zijn per kanaal, wat betekent dat moet kunnen worden aangegeven welk kanaal bewerkt dient te worden. Er moet dus van paneel geswitcht kunnen worden. In dit panel moet de snelheid, gevoeligheid, kleur kunnen worden aangepast.

Ook zal er een paneel dienen te komen voor de algemene opties zoals deze in de recorderklasse terug te vinden zijn. Het aansturen van de minimum- en maximumwaarde die in het raster van de grafiek worden getekend vindt hier plaats. Daarnaast zal hier de pixel-per-value optie terug te vinden moeten zijn en een manier om het te bewerken kanaal te selecteren zoals hierboven beschreven.

Per panel of view dient er een controller te worden gemaakt die elk inkomend event afhandelt. Dit panel dient het model en het paneel of view te kennen en daarmee de koppeling hiertussen de volbrengen.

H3.2. Use Case diagram en Klassendiagram

Voor er aan het schrijven van de software begonnen kan worden, wordt er aan de hand van de geformuleerde specificaties een enkel UML diagram opgesteld. Dit zal een Use Case diagram zijn inclusief beschrijvingen en een klassendiagram. Deze zijn wegens hun grootte terug te vinden in de bijlages.

H4. Keuzes

In dit hoofdstuk worden de gemaakte keuzes onderbouwt wat betreft het schrijven van de software. De volgorde waarin dit gebeurt is dat van het MVC model, beginnende met het model en eindigend met de controller.

H4.1. Model

Het model bestaat uit drie klassen, een klasse voor een gegenereerde waarde, een voor een kanaal en een die de voorgaande klassen bijhoudt in een lijst. Voor het opslaan van de gegenereerde waardes en de kanalen de in het model is gekozen voor een LinkedList in plaats van ArrayList. Dit in verband met de hogere snelheden wat betreft het bewerken van de lijst, lees het toevoegen of verwijderen van een waarde. Het snel toevoegen is van belang aangezien elke tien milliseconden een waarde gegenereerd wordt. Het snel kunnen verwijderen van de waardes is belangrijk omdat we met het tekenen en weergeven van waardes willen voorkomen dat deze meerdere keren weergegeven worden wanneer door de lijst gelopen wordt. Daarom worden de waardes ook aan meerdere lijsten toegevoegd, een lijst voor de te tekenen waardes in het hoofdframe, een voor de weer te geven waardes in het debugframe en een lijst die alle waardes bijhoudt. Die laatste lijst is voor het opslaan in een CSV bestand van zoveel mogelijk waardes voor zover de grootte van de lijst het toelaat.

H4.2. View

Voor de views of panels is gekozen deze te zetten in een BorderLayout. Dit heeft als voordeel dat het belangrijkste onderdeel in het midden gezet kan worden en de rest hierom heen. Dit onderdeel neemt dan de meeste prominente plek in, in de lay-out van de recorder is dit dan ook de te tekenen grafiek. Relatief kleinere en minder belangrijke panels zijn die voor de algemene instellingen zoals pixel-per-value en het panel wat aangeeft of een kanaal zichtbaar moet zijn. Deze zijn dan ook zowel aan de boven- als onderkant toegevoegd. Als laatste is er nog het panel voor de instellingen per kanaal, omdat deze ook relatief veel gebruikt wordt is deze aan de rechterkant van de view te vinden waarin de grafiek getekend wordt. Hierdoor neemt het nog een redelijke grote plek in de lay-out, zodat de componenten groot van stuk zijn en makkelijk te bedienen. Dit alles is in een hoofdpanel ondergebracht. In het volgende gedeelte wordt er verder in gegaan op elke view of panel.

RecorderCenterView, de klasse waarmee de grafiek getekend wordt, bevat drie methodes om het geheel te tekenen. Deze krijgen door een aanroep in de paintComponent methode de grafische context mee waar volgens de drie onderdelen mee getekend kunnen worden. Dit zijn het raster, de extreme waardes en de gegenereerde waardes door deze te verbinden met elkaar. Bij elke van deze onderdelen wordt er een kopie gemaakt van de grafische context zodat de originele parameters van de context bewaarde blijven. Dit door het vele bewerken ervan, waardoor een ander onderdeel niet met ongewenste parameters getekend wordt. Voor het raster is gekozen deze als hokjes van vijftig bij vijftig pixels te tekenen wat op een gemiddeld scherm ongeveer hokjes van een bij een centimeter oplevert. De extreme waardes worden als horizontale lijn getekend aan de hand van waarde die het via de update methode van het model meekrijgt. Als laatste onderdeel dient de grafiek nog te worden getekend, dit wordt gedaan door binnenkomende lijst met waardes die het net zoals de extreme waardes mee krijgt van het model via de update methode, te verbinden met elkaar door er een lijn tussen te tekenen. Op het moment deze waarde getekend is wordt deze verwijdert uit het model.

Bij de RecorderNorthView is de keuze gevallen op om deze zoals de naam al zegt als view te implementeren in plaats van panel. Dit omdat in deze view de twee JSliders zich bevinden om de minimum- en maximum waarde te tekenen. Hoe groot deze waardes in pixels zijn is afhankelijk van de grootte van de grafiek. Door middel van de update methode wordt dit via het model meegegeven. De range van de slider wordt hiermee aangepast. Ook bevindt zich in deze view de JComboBox waar in de namen van de cards van het RecorderWestPanel staan. In het stuk over dit panel wordt daar verdere uitleg over gegeven. Verder bestaat de view nog een uit een JCheckBox waarmee kan worden aangegeven of de grafiek als afzonderlijke pixels getekend dient te worden. Dit geldt voor alle kanalen vandaar het zich in dit panel bevindt.

Over de RecorderSouthPanel, zoals de naam zegt het panel wat zich onderin de user interface begeeft, kunnen we kort zijn. Door middel van JCheckboxes wordt via de controller aan het model doorgegeven of een kanaal zichtbaar moet zijn of niet.

Als laatste component is er nog de RecorderWestPanel die eigenlijk bestaat uit twee onderdelen. Een RecorderWestCardPanel waarvan er vier aangemaakt wordt, voor elk kanaal een. Tussen deze kanaalpanelen kan geswitcht worden door middel van de CardLayout, die wordt aangestuurd door een JComboBox uit de RecorderNorthView. Dit heeft als voordeel dat er gemakkelijk voor elk onderdeel apart een listener aangemaakt kan worden, die ook nog eens kanaal onafhankelijk werken. Want ondanks dat er niets aan de layout verandert is wordt er onderliggend een andere listener aangeroepen voor dezelfde component op het moment er van kanaal verandert is. Door de events allemaal apart af te handelen, houdt dat de code makkelijk leesbaar.

Voor de in te stellen snelheid is gebruik gemaakt van een JSlider, waarvan de range aan waardes bij een begint. Dit omdat bij nul de grafiek niet getekend zou worden wat een ongewenste situatie oplevert. Bij het kiezen van de gevoeligheid of schaal kan gebruik gemaakt worden van een JComboBox waarin de waardes te selecteren zijn zoals opgegeven in de opdracht. Voor het selecteren van de kleur is gekozen om gebruik te maken van een JButton, wat vervolgens event oplevert waar de kleur geselecteerd kan worden. Hierover is meer te vinden onder het kopje control.

Daarnaast bevindt zich in het recorderframe nog een menu. Deze biedt opties om het debugframe te laten zien, een screenshot te maken, het programma af te sluiten of de gegevens op te slaan. Voor het opslaan is ook een accelerator toegevoegd waardoor de combinatie control-s het menu item activeert.

H4.3. Control

De controller handelt de events af per panel of view. Dit is gedaan om de controllers overzichtelijk te houden en zo duidelijkheid te scheppen in welke events worden afgehandeld. De meeste handlers zijn basic en lezen de waarde uit van de component die de event triggert en geeft dit door aan het model. Er zijn enkele handlers die nadere aandacht verdienen.

Een daarvan is de windowadapter die gehangen is aan het recorderframe, op het moment een gebruiker het frame probeert af te sluiten wordt hier het event afgehandeld. Dit is het laten zien van een dialog waar gevraagd wordt of de gebruiker de gegevens wil opslaan. Als de gebruiker hiervoor kiest komt vervolgens een dialog tevoorschijn waar een bestandslocatie mee gekozen kan worden. Vervolgens wordt het bestand hier opgeslagen. Dit zelfde geldt voor de savehandler waarop op dezelfde manier een bestand kan worden opgeslagen.

Bij het maken van een screenshot wordt gebruik gemaakt van de Robot klasse in Java. Deze klasse bedoeld voor automatiseren van bepaalde handelingen heeft een functie om een screenshot te maken. Als hiervoor gekozen wordt heeft de gebruiker de optie om deze op te slaan in een bestandslocatie naar keuze, zoals beschreven in de vorige alinea.

Voor het kiezen van de kleur wordt door de controller een JColorChooser dialog getoond op het moment via een klik op de knop de methode van de handler aangeroepen wordt. De keuze voor een kleur wordt vervolgens opgeslagen in het model.

H5. Reflectie

Binnen het team was de samenwerking goed, ondanks enig verschil in kennis over Java tussen de beide leden. Dit werd goed opgevangen door het ontwerp samen te maken en vervolgens met enige uitleg dit te implementeren in de code. Echter doordat er verschil was in voortgang met de andere opdrachten was het soms lastig een geschikt moment te vinden. Wanneer er een zo'n moment gevonden was verliep de samenwerking soepel en werd er goed doorgewerkt aan het eindresultaat. De uitkomst is dan ook na tevredenheid van beide teamleden. Er valt wel op te merken dat er te laat is begonnen met het voorbereidende werk, waardoor uiteindelijk niet alle vragen gesteld kon worden die eventueel nodig waren voor de implementatie. Door veel research is dit als nog gelukt maar het had team enigszins tijd kunnen besparen mocht er eerder begonnen zijn en hadden deze vragen op tijd in de lesuren kunnen worden gesteld.

H6. Conclusie

Het doel van het maken van een transient recorder was kennis op doen van het MVC ontwerp patroon en het maken van een gebruiksvriendelijk user interface. Er kan geconcludeerd worden dat aan de hand van dit model met behulp van het tekenen met de grafische context en het implementeren van ActionListeners de opdracht naar behoren voltooid kon worden. Echter voor het schrijven van de code voor de extra functionaliteit was research nodig. In wezen is deze extra functionaliteit qua opzet niet anders als de basisfunctionaliteit maar kregen we echter te maken met componenten en layouts waarvan de werking nog niet bekend was. Het projectdoel is dan ook bereikt om een werkende recorder te creëren volgens het MVC ontwerp patroon.

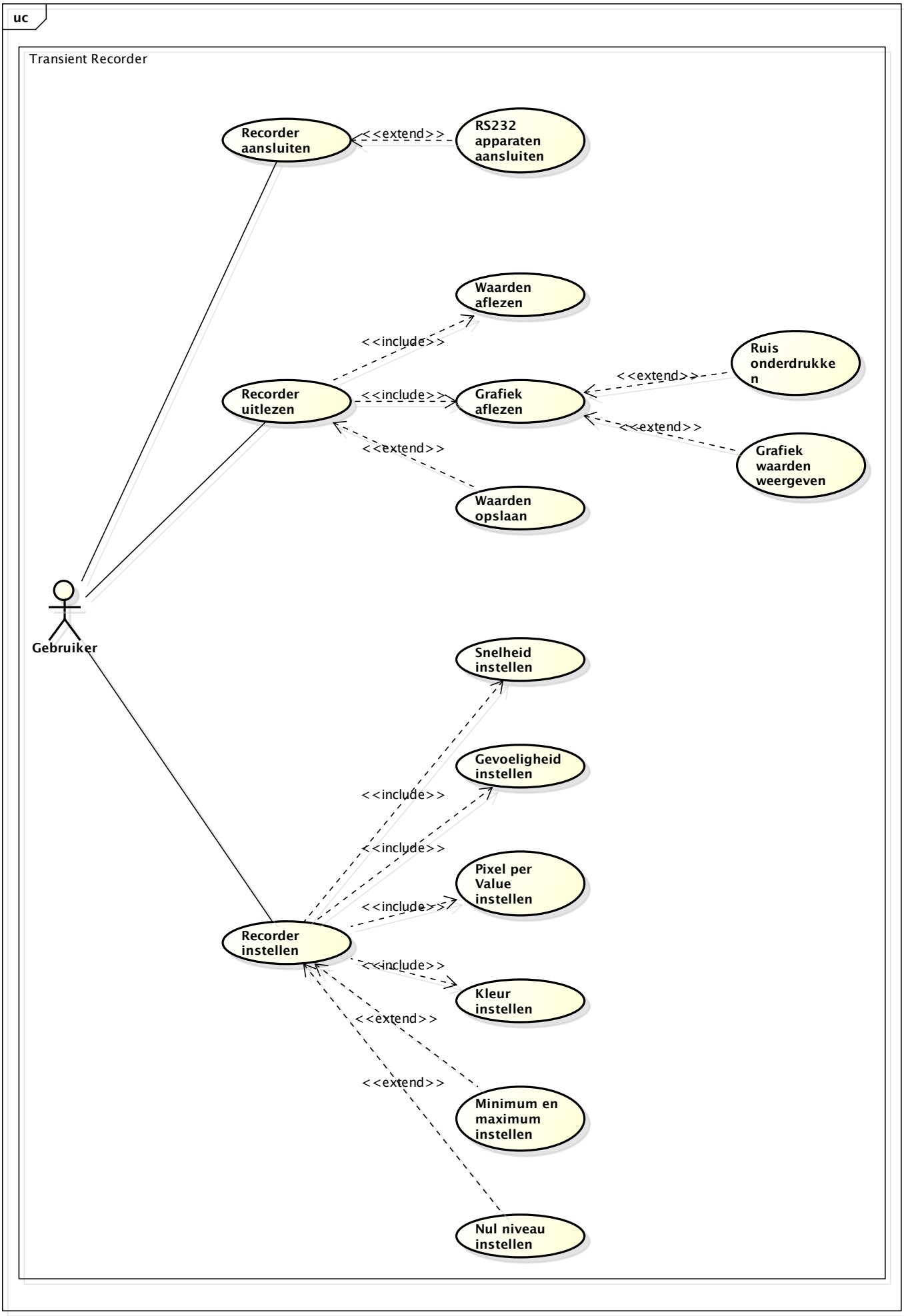
H7. Bijlagen

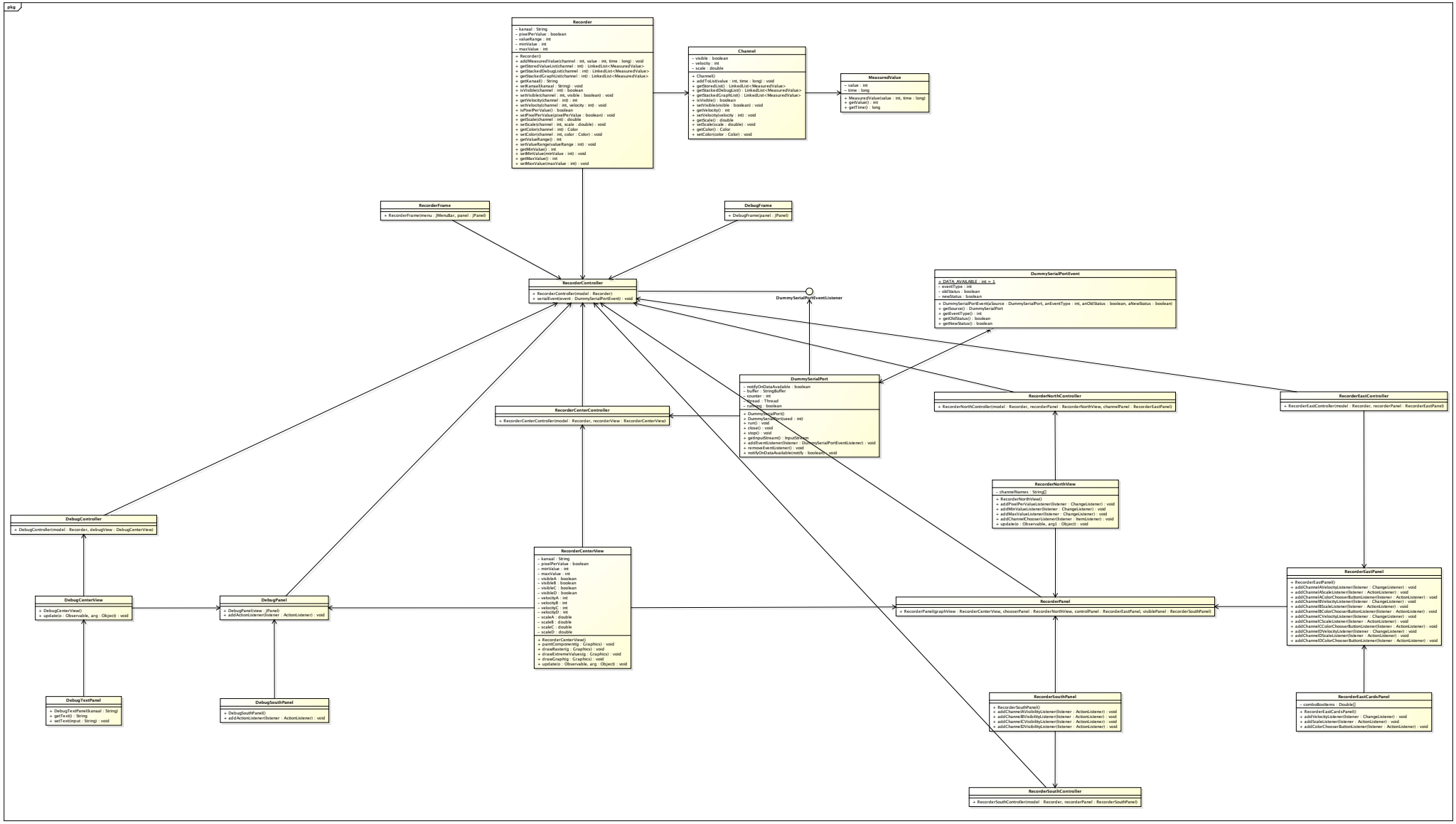
H7.1. Klassenanalyse

Verplicht onderdeel *Optioneel onderdeel* Geïmplementeerd Niet gebruikt

Zelfstandig naamwoord	**Aantal**	**Omschrijving**
(User)Interface	3	Software begrip
Actuator	1	Hardware begrip
Analoog	1	Hardware begrip
Applicatie	2	Software begrip
Arduino	3	Hardware begrip
Belang	1	Algemeen begrip
Bestand	1	Software begrip
Blackboard	1	Resource
Bordje	2	Hardware begrip
ChannelPanel	1	Panelklasse (ChannelCardsPanel)
Characters	2	Primitief type Java
Checkbox	1	GUI onderdeel
Component	3	GUI onderdeel
CSV bestand	3	*Attribuut (RecorderMenu)*
Data	3	Software begrip
Deel	1	Algemeen begrip
Dialog	1	GUI onderdeel
Digitaal	1	Hardware begrip
Division	1	Grafische context onderdeel
DummySerialPort	3	Controllerklasse (DummySerialPort)
Exit	1	*Attribuut (RecorderMenu)*
Fijnafstelling	1	*Attribuut*
Filters	1	*Attribuut*
Functie / Functionaliteit	3	Algemeen begrip
Gegevens	1	Algemeen begrip
Gevoeligheid	2	Attribuut (Channel)
Grafiek	8	Grafische context onderdeel
Graphics	1	Grafische context
Ground	1	*Attribuut (Recorder)*
Hardware	1	Algemeen begrip
Help	1	*Attribuut menu*
Instantie	2	Software begrip
Instellingen	4	Algemeen begrip
JCheckBox	1	GUI onderdeel
JComboBox	1	GUI onderdeel
JComponents	1	GUI onderdeel
JFileChooser	1	GUI onderdeel
JFrame	1	GUI onderdeel
JFrame	3	GUI onderdeel
JFreeGraph	1	GUI onderdeel
JSlider	1	GUI onderdeel

Kanaal	18	Modelklasse (Channel)
Keuze	2	Algemeen begrip
Kleur	4	Attribuut (Channel)
Layout	2	Software begrip
Lijn	1	Grafische context onderdeel
Lijst	2	Attribuut (Channel)
Max waarde	2	*Attribuut (Recorder)*
Measured Value	1	Modelklasse (Measured Value)
Meetwaarde	9	zie Measured Value
Menu	1	*ModelKlasse (RecorderMenu)*
Min waarde	2	*Attribuut (Recorder)*
Monitor	3	zie Terminalvenster
Mousepointer	1	Software/Hardware begrip
MVC	2	Ontwerpmodel
Naam	1	Algemeen begrip
Nul-niveau	1	zie Ground
Oscilloscoop	3	Systeemnaam
Panels	1	Software begrip
PC	1	Software/Hardware begrip
Periode	1	Algemeen begrip
Pixel (per value)	2	Attribuut (Channel)
Plaatje	1	Algemeen begrip
Protocol	1	Software begrip
Raster	2	Grafische context onderdeel
Record/stop knop	1	*Attribuut (RecorderMenu)*
Recorder	3	Systeemnaam
RS232 instellingen	1	*Attribuut*
RS232 interface	1	*Viewklasse*
RS232 meters	1	*GUI onderdeel*
RS232 port	1	Software begrip
RS232 verbinding	1	Software begrip
Ruis	1	*Attribuut*
Schaal	2	Attribuut (Channel)
Screenshot	1	*Attribuut (RecorderMenu)*
Sensorgegevens	1	Algemeen begrip
Signalen	2	Algemeen begrip
Slider	2	GUI onderdeel
Snelheid	1	Attribuut (Channel)
Specificaties	2	Algemeen begrip
Stappen	2	Algemeen begrip
Terminal venster	1	Viewklasse (DebugView)
Tijd	5	Atribuut (Measured Value)
Transient recorder	4	Systeemnaam
Transparantie	1	Instelling grafische context
Verbinding	1	Algemeen begrip
Waarde	6	zie Measured Value


uc
Transient Recorder
Recorder aansluiten
<<extend>>
RS232 apparaten aansluiten
Waarden aflezen
<<include>>
Recorder uitlezen
<<include>>
Grafiek aflezen
<<extend>>
Ruis onderdrukken
<<extend>>
Grafiek waarden weergeven
<<extend>>
Waarden opslaan
Gebruiker
Snelheid instellen
Gevoeligheid instellen
<<include>>
<<include>>
Pixel per Value instellen
<<include>>
Recorder instellen
<<include>>
Kleur instellen
<<extend>>
<<extend>>
Minimum en maximum instellen
Nul niveau instellen


pkg
Recorder
- kanaal : String
- pixelPerValue : boolean
- valueRange : int
- minValue : int
- maxValue : int
+ Recorder()
+ addMeasuredValue(channel : int, value : int, time : long) : void
+ getStoredValueList(channel : int) : LinkedList<MeasuredValue>
+ getStackedDebugList(channel : int) : LinkedList<MeasuredValue>
+ getStackedGraphList(channel : int) : LinkedList<MeasuredValue>
+ getKanaal() : String
+ setKanaal(kanaal : String) : void
+ isVisible(channel : int) : boolean
+ setVisible(channel : int, visible : boolean) : void
+ getVelocity(channel : int) : int
+ setVelocity(channel : int, velocity : int) : void
+ isPixelPerValue() : boolean
+ setPixelPerValue(pixelPerValue : boolean) : void
+ getScale(channel : int) : double
+ setScale(channel : int, scale : double) : void
+ getColor(channel : int) : Color
+ setColor(channel : int, color : Color) : void
+ getValueRange() : int
+ setValueRange(valueRange : int) : void
+ getMinValue() : int
+ setMinValue(minValue : int) : void
+ getMaxValue() : int
+ setMaxValue(maxValue : int) : void
Channel
- visible : boolean
- velocity : int
- scale : double
+ Channel()
+ addToList(value : int, time : long) : void
+ getStoredList() : LinkedList<MeasuredValue>
+ getStackedDebugList() : LinkedList<MeasuredValue>
+ getStackedGraphList() : LinkedList<MeasuredValue>
+ isVisible() : boolean
+ setVisible(visible : boolean) : void
+ getVelocity() : int
+ setVelocity(velocity : int) : void
+ getScale() : double
+ setScale(scale : double) : void
+ getColor() : Color
+ setColor(color : Color) : void
MeasuredValue
- value : int
- time : long
+ MeasuredValue(value : int, time : long)
+ getValue() : int
+ getTime() : long
RecorderFrame
+ RecorderFrame(menu : JMenuBar, panel : JPanel)
DebugFrame
+ DebugFrame(panel : JPanel)
RecorderController
+ RecorderController(model : Recorder)
+ serialEventEvent : DummySerialPortEvent() : void
DummySerialPortEventListener
DummySerialPortEvent
+ DATA_AVAILABLE : int = 1
- eventType : int
- oldStatus : boolean
- newStatus : boolean
+ DummySerialPortEvent(aSource : DummySerialPort, anEventType : int, anOldStatus : boolean, aNewStatus : boolean)
+ getSource() : DummySerialPort
+ getEventType() : int
+ getOldStatus() : boolean
+ getNewStatus() : boolean
DummySerialPort
- notifyOnDataAvailable : boolean
- buffer : StringBuffer
- counter : int
- thread : Thread
- running : boolean
+ DummySerialPort()
+ DummySerialPort(aSet : int)
+ run() : void
+ close() : void
+ stop() : void
+ getInputStream() : InputStream
+ addEventListener(listener : DummySerialPortEventListener) : void
+ removeEventListener() : void
+ notifyOnDataAvailable(notify : boolean) : void
RecorderNorthController
+ RecorderNorthController(model : Recorder, recorderPanel : RecorderNorthView, channelPanel : RecorderEastPanel)
RecorderEastController
+ RecorderEastController(model : Recorder, recorderPanel : RecorderEastPanel)
RecorderCenterController
+ RecorderCenterController(model : Recorder, recorderView : RecorderCenterView)
RecorderNorthView
- channelNames : String[]
+ RecorderNorthView()
+ addPixelPerValueListener(listener : ChangeListener) : void
+ addMinValueListener(listener : ChangeListener) : void
+ addMaxValueListener(listener : ChangeListener) : void
+ addChannelChooserListener(listener : ItemListener) : void
+ update(o : Observable, arg1 : Object) : void
DebugController
+ DebugController(model : Recorder, debugView : DebugCenterView)
RecorderCenterView
- kanaal : String
- pixelPerValue : boolean
- minValue : int
- maxValue : int
- visibleA : boolean
- visibleB : boolean
- visibleC : boolean
- visibleD : boolean
- velocityA : int
- velocityB : int
- velocityC : int
- velocityD : int
- scaleA : double
- scaleB : double
- scaleC : double
- scaleD : double
+ RecorderCenterView()
+ paintComponent(g : Graphics) : void
+ drawRaster(g : Graphics) : void
+ drawExtremeValues(g : Graphics) : void
+ drawGraph(g : Graphics) : void
+ update(o : Observable, arg : Object) : void
RecorderEastPanel
+ RecorderEastPanel()
+ addChannelAVelocityListener(listener : ChangeListener) : void
+ addChannelAScaleListener(listener : ActionListener) : void
+ addChannelAColorChooserButtonListener(listener : ActionListener) : void
+ addChannelBVelocityListener(listener : ChangeListener) : void
+ addChannelBScaleListener(listener : ActionListener) : void
+ addChannelBColorChooserButtonListener(listener : ActionListener) : void
+ addChannelCVelocityListener(listener : ChangeListener) : void
+ addChannelCScaleListener(listener : ActionListener) : void
+ addChannelCColorChooserButtonListener(listener : ActionListener) : void
+ addChannelDScaleListener(listener : ActionListener) : void
+ addChannelDColorChooserButtonListener(listener : ActionListener) : void
DebugCenterView
+ DebugCenterView()
+ update(o : Observable, arg : Object) : void
DebugPanel
+ DebugPanel(view : JPanel)
+ addActionListener(listener : ActionListener) : void
RecorderPanel
+ RecorderPanel(graphView : RecorderCenterView, chooserPanel : RecorderNorthView, controlPanel : RecorderEastPanel, visiblePanel : RecorderSouthPanel)
DebugTextPanel
+ DebugTextPanel(kanaal : String)
+ getText() : String
+ setText(input : String) : void
DebugSouthPanel
+ DebugSouthPanel()
+ addActionListener(listener : ActionListener) : void
RecorderSouthPanel
+ RecorderSouthPanel()
+ addChannelAVisibilityListener(listener : ActionListener) : void
+ addChannelBVisibilityListener(listener : ActionListener) : void
+ addChannelCVisibilityListener(listener : ActionListener) : void
+ addChannelDVisibilityListener(listener : ActionListener) : void
RecorderEastCardsPanel
- comboBoxItems : Double[]
+ RecorderEastCardsPanel()
+ addVelocityListener(listener : ChangeListener) : void
+ addScaleListener(listener : ActionListener) : void
+ addColorChooserButtonListener(listener : ActionListener) : void
RecorderSouthController
+ RecorderSouthController(model : Recorder, recorderPanel : RecorderSouthPanel)