1 Functionaliteit

Alle features die we wilden implementeren zijn geïmplementeerd. Alle noden van de opdrachtgever (Sectie **??**) zijn geïmplementeerd. Er zijn verschillende uitbreidingen die we toch graag geïmplementeerd hadden, echter zijn deze features geen core functionaliteit. Deze features zijn niet geïmplementeerd omwille van zowel tijdsbeperkings als de voorkeur om het programma zorgvoldig af te werken. Deze features worden besproken in Sectie **??**.

Eerst worden de verschillende features opgesomd, hierna volgt een meer gedetailleerde beschrijving van de features.

1.1 Features:

- professioneel uiterlijk (neutrale kleuring)
- beschikbaar in meerdere talen
- debug modus: stappen doorheen het programma
- inladen en opslaan van programma's in een leesbaar formaat
- events aanmaken
- klassen aanmaken
- programma opbouwen via blokken
- blokken verwijderen
- instanties maken
- programma uitvoeren

1.2 Extra Features:

- selectie plaatsing blokken
- breakpoints plaatsen
- typechecking
- sleep blok
- data editor
- kostuums opslaan
- verplaatsen aanknopingspunt wires
- naam van een klasse aanpassen

- doorzichtigheid blokken
- hulp menu
- event filter

1.3 Uitleg eatures:

1.3.1 Professionele look

De IDE heeft een **professioneel** uiterlijk zoals te zien in Figuur **??**. De verschillende blokken die de gebruiker kan plaatsen hebben neutrale kleuren.

1.3.2 Meerdere talen

De IDE is beschikbaar in zowel het **Nederlands als het Engels**. Hoe het wisselen tussen de verschillende talen gebeurt is te zien in Figuur **??**

1.3.3 Inladen en opslaan

De gebruiker kan programma's **opslaan en inladen** in een leesbaar formaat, dit gebeurt door middel van het XML-formaat. Het inladen zal de IDE niet doen crashen op een verkeerde layout, er kan wel een eigen error message uitgeprint worden naar de console zodanig dat de gebruiker weet wat er fout gaat. Er is ook de mogelijkheid om een nieuw project te starten, dit verwijdert alle hudige niet opgeslagen progressie.

1.3.4 Debug modus

Er is een **debug modus** aanwezig. Deze laat toe om stap voor stap doorheen het programma te lopen. Er zijn verschillende extra features geïmplementeerd met betrekking tot de debug modus, deze staan uitgelegd in Sectie 1.2.

1.3.5 Events

De gebruiker kan nieuwe **events aanmaken en verwijderen**. De inhoud van deze events kan de gebruiker aanpassen. Hij kan nieuwe informatie toevoegen, informatie verwijderen of het type van de informatie aanpassen.

1.3.6 Klassen

Klassen kunnen aangemaakt en verwijderd worden.

1.3.7 Blokken plaatsen

Het Klasse-view laat toe om **blokken** te plaatsen. Deze blokken kunnen aan de linkerkant geselecteerd worden. In het dropdown menu staan verschillende categorieëen met verschillende blokken. Blokken kunnen vervolgens volgens

hun regels genest worden in elkaar. Het blokje “Input Events” toont alle input events van een bepaalde klasse. Via het dropdown menu kunnen input events toegevoegd worden, via het kruisje kan dat input-event verwijderd worden. Globale variabelen zijn niet toegelaten in de IDE. De gebruiker kan nieuwe member **variabelen** aanmaken. Een member variabele kan overal in een klasse gebruikt worden. Elke instantie heeft zijn eigen instanties van de member variabelen. Er kunnen verschillende **kostuums** toegevoegd worden aan een klasse. Deze kan als primair gezet worden, zodanig dat elke instantie van een klasse begint met dat kostuum.

1.3.8 Blokken verwijderen

De gebruiker kan een **blok verwijderen** door middel van een optie die tevoorschijn komt via een popup. Deze popup kan getoond worden als de gebruiker de rechtmuisknop indrukt. De gebruiker kan geen geneste blokken apart verwijderen. Het verwijderen van een blok zal altijd van toepassing zijn op de meest externe blok, deze verwijdert dan ook alle geneste blokken. De reden dat geneste blokken niet verwijderd kunnen worden is duidelijkheid voor de gebruiker. Blokken die genest zijn vormen een geheel. Een actie zoals verwijderen heeft dan ook betrekking op het geheel.

1.3.9 Instanties

In het Frame-view kan de gebruiker **instanties** aanmaken van een Klasse. De gebruiker kan input-events verbinden met output-events van hetzelfde type. Het Canvas-view toont alle instanties en hun geselecteerde uiterlijk. De gebruiker kan deze instanties en verbindingen ook weer verwijderen.

1.3.10 Uitvoering

De gebruiker kan een programma **compileren, uitvoeren, stoppen of stappen** (een stap uitvoeren). Deze actie is mogelijk voor zowel de debug modus als de gewone modus.

1.4 Uitleg extra features:

We hebben verschillende extra features geïmplementeerd. Sommige features hiervan zijn slechts kleine toevoegingen, andere zijn grotere toevoegingen.

1.4.1 Selecteren blokken

In zowel het Klasse view als het Frame view kan de gebruiker in de linkerbalk blokken selecteren. Vervolgens zal elke muisklik op het Klasse venster een nieuwe blok van het geselecteerde type maken op de muispositie. Beide views zijn ook “oneindig” groot (gelimiteerd door de grote van een int). De blauwe lijn stelt de $x - as$ voor, de rode lijn stelt de $y - as$ voor zoals te zien in Figuur **??**

1.4.2 Breakpoints plaatsen

De **debug modus** is uitgebreid met de mogelijkheid om break-points aan te duiden. Deze kunnen gezet worden op een blok. Als de gebruiker doorheen het programma stapt/loopt in debug modus, gaan zwarte omlijningen aanduiden waar de uitvoer zich momenteel bevindt zoals te zien in Figuur **??**. De `break` stopt enkel het proces dat de `break` tegenkomt, de andere processen blijven doorlopen.

1.4.3 Typechecking

Er is ook **typechecking** toegepast. Dit gebeurt op drie niveau's. Eenderzijds gebeurt typechecking visueel. Dit wordt getoond door een rode border (Sectie **??**). Anderszijds zal de compiler stellen dat compilatie faalt indien het programma niet volledig is. Als er toch fouten door deze checks komen, is er nog runtime errorchecking. Dit kan optreden wanneer een functie aanroep gebeurt naar een niet-bestaande functie. Er wordt dan ook gespecifieerd wat de fout exact is.

1.4.4 Sleep blok

Er is een `sleep` blok geïmplementeerd om een proces te laten slapen voor het gespecifieerde aantal milliseconden.

1.4.5 Data editor

We hebben een data editor geïmplementeerd zoals te zien in Figuur **??**. De data editor laat het programma zien in zijn XML-vorm. De gebruiker kan deze dan vrij aanpassen. Als de gebruiker terug wisselt naar een ander view zullen de aanpassingen in de data editor ook zichtbaar zijn in de visuele views. De Data editor heeft ook syntax highlighting. Dit is geïmplementeerd via een externa library [**?**].

1.4.6 Kostuums opslaan

De kostuums van een klasse worden ook opgeslagen in dezelfde locatie als de XML-file. Zodanig kan een project gebruikt worden op verschillende computers zonder dat de XML-file aangepast moet worden.

1.4.7 Aanknopingspunt wire

Het aanknopingspunt voor een wire kan wisselen van links naar rechts (en andersom) zoals te zien in Figuur **??**. Dit verhoogt de leesbaarheid bij grotere programma's.

1.4.8 Naam klasse aanpassen

De naam van een klasse kan aangepast worden. Door de rechtermuisknop te gebruiken op het tab van die klasse, kan zijn naam aangepast worden.

1.4.9 Doorzichtigheid blokken

Als de gebruiker een blok vast neemt met zijn muis, zal de blok doorzichtig kleuren. Dit maakt het duidelijker voor de gebruiker waar de blok terecht zal komen.

1.4.10 Hulp menu

Er is een **hulp menu** toegevoegd. De gebruiker kan via de rechtermuisknop een hulp-dialoog openen over de blok die zich momenteel onder de muis bevindt. Deze dialoog geeft een korte uitleg over de blok zoals zijn functionaliteit.

1.4.11 Event filter

In het Frame view is een filter geïmplementeerd. Hierbij kan de zichtbaarheid van events geregeld worden. Verbindingen tussen instanties kunnen op deze manier zichtbaar/onzichtbaar gemaakt worden. Dit wordt geregeld per event type. Dit staat verder beschreven in Sectie **??**.

1.5 Evaluatiecriteria

In het analyseverslag stonden enkele evaluatiecriteria waaraan de applicatie moest voldoen. Enkele van deze criteria betrof features die geïmplementeerd zijn. Andere criteria stellen iets over de uitvoering van de IDE.

Zo moest een verzonden event gelijktijdig opgevangen worden door de geabboneerde instanties. Na elke primitieve stap zullen alle verzonden events afgehandeld worden. Hierdoor worden die ook gelijktijdig opgevangen. Aangezien de IDE zelf threading simuleert gebeurt de uitvoer ook concurrent, en worden er verschillende processen gesimuleerd. Hierdoor zal een proces met een eewige loop niet de Virtual machine doen crashen.

Er worden geen limitaties (vanuit de IDE) opgelegd op het aantal klassen, events, instanties, processen dat een gebruiker kan maken.

De IDE ondersteund volledige functies, hierdoor is parameter passing en recursiviteit mogelijk. Een lock kan niet gezet worden indien een ander proces al een lock gezet heeft op dezelfde instantie. Hierdoor zijn deadlocks niet mogelijk en indien de gebruiker locks gebruikt, kunnen geen racing conditions optreden.

Bij een runtime fout zal enkel het proces dat de fout genereerde afbreken.

Figure 1: Alle geïmplementeerde blokken.

Figure 2: Selecteer de taal.

Figure 3: Infinite view

Figure 4: Debug view

Figure 5: Data Editor

Figure 6: Switchen aanknopingspunt