Documentatie

Domotica is een programma die het beheers van energie in een gebouw controleert. Elke kamer heeft een steward die verschillende devices beheert, en de central-unit controleert de verschillende stewards. Deze stewards kunnen berichten zenden naar hun devices. Een device bestaat uit sensoren en actuatoren die diverse elementen (temperatuur, licht-intensiteit...) kunnen meten en aanpassen.
Het programma is gesplitst in verschillende modules :

internal : Bevat de interne werking van het domotica programma. Hierin zijn onder andere Steward, Device, Sensor gedefinieerd.

- *bijvoorbeeld* : Devices toevoegen aan Stewards.

communication : Laat toe om boodschappen te sturen naar de hardware.

- *bijvoorbeeld* : Zend de instructie om de temperatuur op te vragen.

physical : Simuleert het gedrag van de hardware.

- *bijvoorbeeld* : Meet de temperatuur van de kamer op.

gui : Bevat alles omtremt de Graphical User Interface. Hiermee kan de gebruiker interageren met het programma.

- *bijvoorbeeld* : Een venster dat toelaat om nieuwe Devices toe te voegen.

db : Beheerst de gegevensbank van het systeem.

- *bijvoorbeeld* : Het bewaren van aanpassingen op de harde schijf.

unit-test : Bevat een aantal tests om te zien of de functionaliteiten van het programma goed zijn geÃŕmplementeerd.

- *bijvoorbeeld* : Test of een boodschap naar de hardware uitgevoerd wordt en het antwoord correct is.

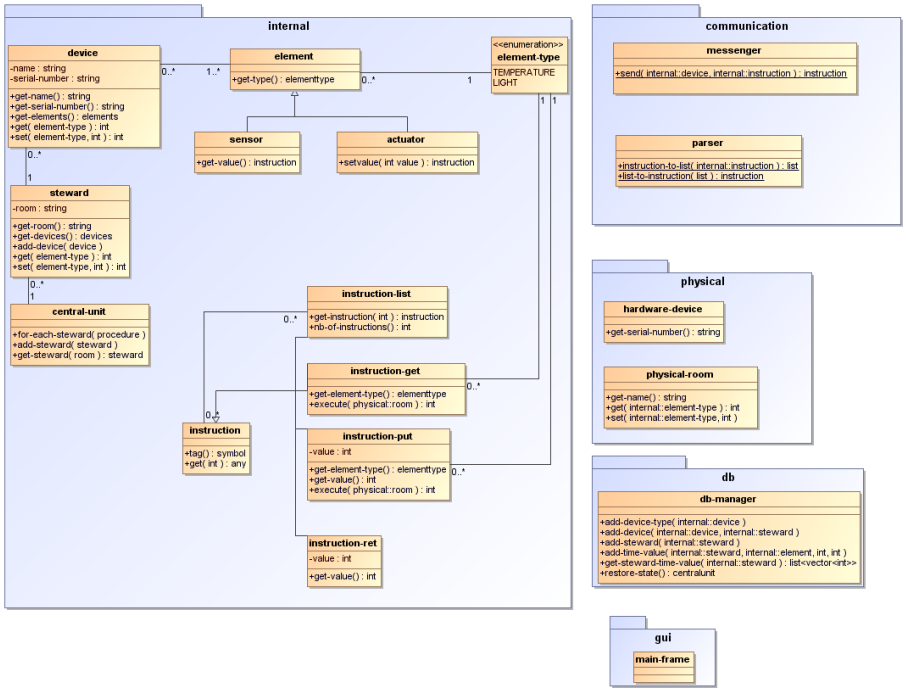


Figure 1: Het class-diagram

1 internal

Het internal module bevat de kern van het domotica systeem. De central-unit houdt de verschillende stewards bij, die instructies kunnen zenden aan de hardware, zoals het aanpassen van de licht-intensiteit of het opvragen van de temperatuur van de kamer.

1.1 Overview

1.1.1 Class

element : Een abstacte classe die een bepaalde element-type bijhoud.

sensor : Vraagt informatie over zijn element-type op.

actuator : Laat toe om de informatie van zijn element-type aan te passen.

device : Een toestel die een aantal sensoren en activatoren bevat.

steward : Bevat en beheerst een aantal devices.

central-unit : Bevat een aantal stewards.

instruction : Een instructie die naar de hardware verzonden kan worden.

instruction-get : Instructie om een element-type te krijgen.

instruction-put : Instructie om een element-type aan te passen.

instruction-list : Een lijst van instructions.

instruction-ret : Een return instructie die een waarde teruggeeft.

1.1.2 Enum

element-type : Alle mogelijke types elementen die door de applicatie toegelaten zijn.

1.2 element-type

Alle mogelijke types elementen die door de applicatie toegelaten zijn. element-type representeert de verschillende attributen die meetbaar zijn door sensoren en aanpasbaar door actuatoren.

1.2.1 Value

TEMPERATURE : Temperatuur.

LIGHT : Licht-intensiteit.

1.2.2 Method

(to-string element-type)→ string? : Zet een element-type om naar een string.

(for-each-element-type proc)→ void? : Past de procedure proc toe voor elk element-type.

(to-string element-type)→ string?
Zet een element-type om naar een string.

- *element-type* : het type dat omgezet moet worden.
- *return* : de resulterende string.

(for-each-element-type proc)→ void?
Past de procedure proc toe voor elk element-type.

- *proc* : de procedure die op elk element-type moet toegepast worden.

1.3 element

Een abstacte classe die een bepaalde element-type bijhoud.
Known Implementations : sensor, actuator.

1.3.1 Constructor

(new-element element-type)→ procedure? : Maakt een nieuw element aan.

1.3.2 Method

(class)→ symbol? : Geeft de naam van de classe terug.

(get-type)→ element-type? : Geeft het type element van dit element.

(new-element element-type)→ procedure?
Maakt een nieuw element aan van een bepaalde element-type.

- *element-type* : Het type van dit element.

(class)→ symbol?
Geeft de naam van de classe terug, in dit geval **Element** .

- *return* : De naam van de classe.

(get-type)→ element-type?
Geeft het type element van dit element.

- *return* : Het element-type.

1.4 sensor

Vraagt informatie over zijn element-type op. Een sensor genereert een instruction die de waarde van zijn element-type opvraagt. Deze instruction kan daarna via een messenger verzonden worden naar hardware zodat die het element-type meet.
Inherit : element.

1.4.1 Constructor

(new-sensor element-type)→ procedure? : Maakt een nieuwe sensor aan.

1.4.2 Method

(class)→ symbol? : Geeft de naam van de classe terug.

(super)→ element? : Geeft de instantie van de superclasse van het sensor-object.

(get-value)→ instruction? : Vraagt de waarde van deze sensor's element-type.

(new-sensor element-type)→ procedure?
Maakt een nieuwe sensor aan.

- *element-type* : Het type van deze sensor.

(class)→ symbol?
Geeft de naam van de classe terug, in dit geval **Sensor** .

- *return* : De naam van de classe.

(super)→ element?
Geeft de instantie van de superclasse van het sensor-object.

- *return* : Het superclasse object.

(get-value)→ instruction?
Deze procedure genereert een **instruction-get** die verzonden kan worden naar hardware zodat deze de waarde opvraagt van deze sensor's element-type in de kamer waarin die zich bevindt.

- *return* : De antwoord-instructie.

1.5 actuator

Laat toe om de informatie van zijn element-type aan te passen. Een actuator genereert een instruction die later verzonden kan worden naar de hardware. Deze instruction past de waarde van element-type naar een nieuwe waarde toe.
Inherit : element.

1.5.1 Constructor

(new-actuator element-type)→ procedure? : Maakt een nieuwe actuator aan.

1.5.2 Method

(class)→ symbol? : Geeft de naam van de classe terug.

(super)→ element? : Geeft de instantie van de superclasse van het actuator-object.

(set-value value)→ instruction? : Past de waarde van deze actuator aan.

(new-actuator element-type)→ procedure?
Maakt een nieuwe actuator aan.

- *element-type* : Het type van deze actuator.

(class)→ symbol?
Geeft de naam van de classe terug, in dit geval **Actuator** .

- *return* : De naam van de classe.

(super)→ element?
Geeft de instantie van de superclasse van het actuator-object.

- *return* : Het superclasse object.

(set-value value)→ instruction?
Deze procedure genereert een **instruction-put** die vraagt om de waarde van deze actuator's element-type aan te passen naar value.

- *value* : De nieuwe waarde.
- *return* : De antwoord-instructie.

1.6 device

Een toestel die een aantal sensoren en actuatoren bevat. Een device kan de element-types van de kamer waarin hij zich bevindt opvragen en aanpassen, vermits hij bestaat uit de corresponderende sensoren en actuatoren.

1.6.1 Constructor

(new-device name serial-number)→ procedure? : Maakt een nieuwe device aan.

1.6.2 Method

(class)→ symbol? : Geeft de naam van de classe terug.

(get-name)→ string? : De naam van deze device.

(get-serial-number)→ string? : Het serial-nummer van deze device.

(get-elements)→ (listof element?) : Een lijst van de elements die de device bevat.

(add-element element)→ void? : Voeg een element toe aan de device.

(get element-type)→ (or/c #f number?) : Vraagt informatie over het meegegeefde element-type.

(set element-type value)→ (or/c #f number?) : Past element-type naar value aan.

(new-device name serial-number)→ procedure?
Maakt een nieuwe device aan. Deze device heeft een naam en een uniek serial-number. Alle devices met dezelfde naam hebben ook dezelfde sensoren en actuatoren (ze hebben wel een verschillend serial-number).
Stel we hebben een device "thermostat" die een **TEMPERATURE** sensor en actuator heeft, dan hebben nieuwe devices met de naam "thermostat" ook deze sensor en actuator.

- *name* : De naam van de device.
- *serial-number* : Het unieke serial-nummer van de device.

(class)→ symbol?
Geeft de naam van de classe terug, in dit geval **Device** .

- *return* : De naam van de classe.

(get-name)→ string?
De naam van deze device.

- *return* : De naam.

(get-serial-number)→ string?
Het unieke serial-nummer van deze device.

- *return* : Het serial-nummer.

(get-elements)→ (listof element?)
Een lijst van de verschillende sensoren en actuatoren waaruit deze device bestaat.

- *return* : De lijst van elements.

(add-element element)→ void?
Voegt een nieuw element toe aan deze device.

- *element* : Het element dat toegevoegd moet worden.

(get element-type)→ (or/c #f number?)
Vraagt informatie over het meegegeefde element-type. De device zoekt of het een sensor heeft met het gevraagde element-type. Het gebruikt dan die sensor om een instruction te genereren die hij gaat zenden naar zijn corresponderen hardware via de messenger. De hardware voert de instructie uit en zendt het antwoord terug naar deze device, die daaruit het resultaat ophaalt en teruggeeft.

- *element-type* : Het type dat gevraagd wordt.
- *return* : **#f** wanneer de informatie niet gevraagd kon worden (wegens het mankeren van de sensor voor dit soort element-type).
  number? De waarde van het element-type.

(set element-type value)→ (or/c #f number?)

Past element-type naar value aan. De device zoekt naar een actuator met het gevraagde element-type. Deze actuator wordt dan gebruikt om een instruction te genereren. Die instruction wordt via de messenger verzonden naar deze device's corresponderende hardware. Na het uitvoeren van de instructie wordt het antwoord terugverzonden.

- *element-type* : Het type dat aangepast moet worden.
- *value* : De nieuwe waarde van het type.
- *return* : **#f** wanneer de informatie niet aangepast kon worden (wegens het mankeren van een actuator voor dit soort element-type). **number?** Een bewijs dat de informatie aangepast is.

1.7 steward

Bevat en beheert een aantal devices. De steward kan aan devices vragen om de element-types van de room waarin hij zich bevindt op te vragen en aanpassen (vermits hij de nodige device heeft).

1.7.1 Constructor

(new-steward room)→ procedure? : Maakt een nieuwe steward aan.

1.7.2 Method

(class)→ symbol? : Geeft de naam van de classe terug.

(get-room)→ string? : De kamer waarin deze steward ligt.

(get-devices)→ (listof device?) : Een lijst van de devices die deze steward beheert.

(add-device device)→ void? : Voeg een device toe aan deze steward.

(get element-type)→ (or/c #f number?) : Vraagt informatie over het meegegeevde element-type.

(set element-type value)→ (or/c #f number?) : Past het element-type naar value aan.

(new-steward room)→ procedure?
Maakt een nieuwe steward aan, er kan maar een steward per kamer zijn, elke room moet dus uniek zijn.

- *room* : De kamer waarin deze steward ligt.

(class)→ symbol?
Geeft de naam van de classe terug, in dit geval **Steward** .

- *return* : De naam van de classe.

(get-room)→ string?
De kamer waarin deze steward ligt.

- *return* : De kamer.

(get-devices)→ (listof device?)
Een lijst van de devices die deze steward beheert.

- *return* : De lijst van devices.

(add-device device)→ void?
Voeg een device toe aan deze steward. Elke device kan maar van een steward afhangen.

- *device* : De device die toegevoegd wordt.

(get element-type)→ (or/c #f number?)
Vraagt informatie over het meegegeefde element-type. De steward zoekt naar een sensor van het gevraagde type in de lijst van devices die hij beheert. Een instructie wordt dan verzonden naar de hardware om informatie te krijgen over het type in de kamer van de steward.

- *element-type* : Het gevraagde type.
- *return* : **#f** als er geen sensor is voor het gezochte element-type.
  number? De gevraagde informatie.

(set element-type value)→ (or/c #f number?)
Past het element-type naar value aan. De steward zoekt naar een actuator in zijn lijst van devices die element-type zou kunnen aanpassen. Informatie wordt dan verzonden naar de hardware om element-type in de kamer van de steward te veranderen.

- *element-type* : Het type dat aangepast moet worden.
- *value* : De waarde waarop het type gezet moet worden.
- *return* : **#f** als er geen actuator is voor het gezochte element-type.
  number? Een bewijs dat element-type aangepast werd.

1.8 central-unit

Bevat een aantal stewards.

1.8.1 Constructor

(new-central-unit)→ procedure? : Maakt een nieuwe central-unit aan.

1.8.2 Method

(class)→ symbol? : Geeft de naam van de classe terug.

(for-each-steward proc)→ void? : Past de procedure proc toe voor elke steward die de central-unit beheert.

(add-steward steward)→ void? : Voegt een nieuwe steward toe.

(get-steward room)→ steward? : Zoekt voor een bepaalde steward in de lijst van stewards van de central-unit.

(new-central-unit)→ procedure?
Maakt een nieuwe central-unit aan.

(class)→ symbol?
Geeft de naam van de classe terug, in dit geval **CentralUnit** .

- *return* : De naam van de classe.

(for-each-steward proc)→ void?
Past de procedure proc toe voor elke steward die de central-unit beheert.

- *proc* : De procedure die op elke steward toegepast wordt.

(add-steward steward)→ void?
Voegt een nieuwe steward toe.

- *steward* : De steward die toegevoegd moet worden.

(get-steward room)→ steward?
Zoekt voor een bepaalde steward in de lijst van stewards van de central-unit.

- *room* : The kamer waarin de steward is (het room-attribuut van de gezochte steward).
- *return* : De gevonden steward.

1.9 instruction

Een instructie die naar de hardware verzonden kan worden.
Known Implementations : instruction-get, instruction-put, instruction-list, instruction-ret.

1.9.1 Constructor

(new-instruction tag . args)→ procedure? : Maakt een nieuwe instruction aan.

1.9.2 Method

(tag)→ symbol? : Geeft de tag terug van deze instruction.

(get i)→ any? : Geeft het i-de argument van de instruction terug.

(new-instruction tag . args)→ procedure?
Maakt een nieuwe instruction aan.

- *tag* : De tag dat deze instruction definieerd.
- *args* : de verschillende argumenten dat aan deze instructie meegegeven worden.

(tag)→ symbol?
Geeft de tag terug van deze instruction.

- *return* : De tag dat deze instruction definieerd.

(get i)→ any?
Geeft het i-de argument van de instruction terug.

- *i* : De positie van het gezochte argument.
- *return* : Het i-de argument.

1.10 instruction-get

Instructie om een element-type te krijgen. Wanneer die verzonden wordt naar de hardware gaat deze instructie de waarde van het element-type opvragen (in de kamer waar de hardware zich bevindt).
Inherit : instruction.

1.10.1 Constructor

(new-instruction-get element-type)→ procedure? : Maakt een nieuwe instruction-get aan.

1.10.2 Method

(tag)→ symbol? : Geeft de tag **TAG_GET** terug.

(get-element-type)→ element-type? : Geeft het element-type van deze instruction terug.

(execute room)→ number? : Geeft de waarde van de instruction-get's element-type voor deze room.

(new-instruction-get element-type)→ procedure?
Maakt een nieuwe instruction-get aan.

- *element-type* : De type van deze instruction-get.

(tag)→ symbol?
Geeft de tag **TAG_GET** terug.

- *return* : **TAG_GET**

(get-element-type)→ element-type?
Geeft het element-type van deze instruction terug.

- *return* : Het element-type van deze instruction-get.

(execute room)→ number?
Geeft de waarde van de instruction-get's element-type voor deze room.

- *room* : De kamer waarvan we de element-type willen weten.

- *return* : de waarde van element-type in room.

1.11 instruction-put

Instructie om een element-type aan te passen. Wanneer die verzonden wordt naar hardware gaat die de waarde van zijn element-type aanpassen (in de room waar de hardware zich bevindt).

1.11.1 Constructor

(new-instruction-put element-type value)→ procedure? : Maakt een nieuwe instruction-put aan.

1.11.2 Method

(tag)→ symbol? : Geeft de tag **TAG_PUT** terug.

(get-element-type)→ element-type? : Geeft het element-type van deze instruction terug.

(get-value)→ number? : Geeft de waarde van deze instruction-put terug.

(execute room)→ number? : Past de waarde van deze instruction's element-type aan naar een nieuwe waarde in room.

(new-instruction-put element-type value)→ procedure?
Maakt een nieuwe instruction-put aan.

- *element-type* : De element-type van deze instruction.
- *value* : De waarde die deze instruction gaat zetten bij het aanpassen van element-type.

(tag)→ symbol?
Geeft de tag **TAG_PUT** terug.

- *return* : **TAG_PUT** .

(get-element-type)→ element-type?
Geeft het element-type van deze instruction terug.

- *return* : Het element-type van deze instruction-put.

(get-value)→ number?
Geeft de waarde van deze instruction-put terug.

- *return* : De waarde van deze instruction.

(execute room)→ number?
Past de waarde van deze instruction's element-type aan naar een nieuwe waarde in room.

- *room* : De kamer waarin de element-type aangepast zal worden.
- *return* : De aangepaste waarde van de kamer.

1.12 instruction-list

Een lijst van instructions.
Inherit : instruction

1.12.1 Constructor

(new-instruction-list . instructions)→ procedure? : Maakt een nieuwe instruction-list aan.

1.12.2 Method

(tag)→ symbol? : Geeft de tag **TAG_LIST** terug.

(get-instruction i)→ instruction? : Geeft de i-de instruction terug.

(nb-of-instructions)→ number? : Geeft het aantal instructions die deze instruction-list bevat.

(new-instruction-list . instructions)→ procedure?
Maakt een nieuwe instruction-list aan.

- *instructions* : De instructions die deze lijst bevat.

(tag)→ symbol?
Geeft de tag **TAG_LIST** terug.

- *return* : **TAG_LIST** .

(get-instruction i)→ instruction?
Geeft de i-de instruction terug.

- *i* : De positie van de gezochte instruction.
- *return* : De instruction op positie i.

(nb-of-instructions)→ number?
Geeft het aantal instructions die deze instruction-list bevat.

- *return* : Het aantal instructions die deze lijst bevat.

1.13 instruction-ret

Een return instructie die een waarde teruggeeft. Deze instructions worden door de hardware als antwoord gegenereerd naar het uitvoeren van een instruction-get of -put. **Inherit** : instruction

1.13.1 Constructor

(new-instruction-ret value)→ procedure? : Maakt een nieuwe instruction-ret aan.

1.13.2 Method

(tag)→ symbol? : Geeft de tag **TAG_RET** terug.

(get-value)→ number? : Geeft de waarde van deze instruction terug.

(new-instruction-ret value)→ procedure?
Maakt een nieuwe instruction-ret aan.

- *value* : De waarde van deze instruction-ret.

(tag)→ symbol?
Geeft de tag **TAG_RET** terug.

- *return* : **TAG_RET** .

(get-value)→ number?
Geeft de waarde van deze instruction terug.

- *return* : De waarde.

2 communication

Laat toe om boodschappen te sturen naar de hardware. Dit module bevat het nodige om te communiceren tussen de steward en de hardware door het zenden en opvangen van instructions.

2.1 Overview

2.1.1 Class

messenger : Kan instructions sturen naar de hardware via ports.

parser : Zet een instruction om naar een lijst van symbols en omgekeerd.

2.2 messenger

Kan instructions sturen naar de hardware via ports. Wanneer een device een instruction wilt zenden zoekt de messenger naar de port waar de corresponderende hardware-device naar luisterd. De instruction wordt dan geparsed en verzonden. Een antwoord wordt door de hardware-device gestuurd.

2.2.1 Method

(send device instruction)→ instruction? : Zendt een instruction naar een bepaalde device.

(send device instruction)→ instruction?
Zendt een instruction naar een bepaalde device.

- *device* : De messenger zoekt naar de port van de corresponderende hardware-device. Daar zal de instruction verzonden worden.
- *instruction* : De instruction die verzonden moet worden.
- *return* : Een antwoord instruction.

2.3 parser

De parser zet een instruction om naar een lijst zodat die via ports gestuurd kan worden. Deze zorgt ook voor het terug-omzetten van lijsten naar geldige instructions.

2.3.1 Method

(instruction-to-list instruction)→ (listof symbol?) : Zet een instruction om naar een list.

(list-to-instruction lst)→ instruction? : Zet een list om naar een instruction.

(instruction-to-list instruction)→ (listof symbol?)
Zet een instruction om naar een list.

- *instruction* : De instruction dat omgezet moet worden.
- *return* : Een lijst die de instruction voorstelt.

(list-to-instruction lst)→ instruction?
Zet een lijst om naar een instruction.

- *lst* : De lijst de naar een instruction omgezet moet worden.
- *return* : De instruction die de lijst representeerde.

3 physical

Omdat we nog niet met hardware omgaan wordt deze gesimuleerd.

3.1 Overview

3.1.1 Class

hardware-device : Simuleert het gedrag van hardware devices (zoals een thermostaat).

physical-room : De gesimuleerde hardware bevindt zich in een kamer, dit wordt gesimuleert door physical-room.

3.2 hardware-device

Simuleert het gedrag van hardware devices (zoals een thermostaat). Een hardware-device luistert naar een bepaalde port en voert instructions uit die door die port verzonden worden (zoals het meten van de temperatuur in de kamer).

3.2.1 Constructor

(new-hardware-device serial-number room)→ procedure? : Maakt een nieuwe hardware-device aan.

3.2.2 Method

(get-serial-number)→ string? : Geeft het serial-nummer van deze hardware-device.

(new-hardware-device serial-number room)→ procedure?
Maakt een nieuwe hardware-device aan. Er wordt automatisch een port een deze hardware toegewezen. Alle ports worden in een map gehouden (key:serial-number, value:port) zodat de messenger instructions kan zenden op de toegewezen port.

- *serial-number* : Het serial-nummer van deze hardware-device. Dit nummer moet hetzelfde zijn als een device, zodanig dat, wanneer er een instruction verzonden wordt naar een device, de messenger de port van de corresponderende hardware-device kan vinden en naar de hardware een instructie kan zenden.
- *room* : De physical-room waarin deze hardware-device zich bevindt.

(get-serial-number)→ string?
Geeft het serial-nummer van deze hardware-device.

- *return* : het serial-nummer.

3.3 physical-room

De gesimuleerde hardware bevindt zich in een kamer, dit wordt gesimuleert door physical-room. Een physical-room bestaat uit verschillenden attributen (zoals de licht-intensiteit). Elk attribuut wordt gerepresenteerd door zijn corresponderende element-type. Die attributen kunnen opgevragen en aangepast worden door de devices die zich in die room bevinden.

3.3.1 Constructor

(new-physical-room name)→ procedure? : Maakt een nieuwe physical-room aan.

3.3.2 Method

(get-name)→ string? : Geeft de naam van deze physical-room terug.

(get element-type)→ number? : Geeft het attribuut van type element-type van deze kamer terug.

(set element-type value)→ void? : Past het attribuut van type element-type van deze kamer aan.

(new-physical-room name)→ procedure?
Geeft de naam van deze physical-room terug.

- *name* : De naam van deze physical-room.

(get-name)→ string?
Geeft de naam van deze physical-room terug.

- *return* : De naam van deze physical-room.

(get element-type)→ number?
Geeft het attribuut van type element-type van deze kamer terug.

- *element-type* : Het type van attribuut dat gegeven zal worden.
- *return* : De waarde van de element-type van deze physical-room.

(set element-type value)→ void?

Past het attribuut van type element-type van deze kamer aan.

- *element-type* : Het type van het attribuut dat aangepast zal worden.
- *value* : De waarde waarop het attribuut gezet zal worden.

4 gui

Bevat alles omtremd de Graphical User Interface. Hiermee kan de gebruiker interageren met het programma. De GUI laat toe om nieuwe type devices toe te voegen, nieuwe stewards toevoegen die bestaande type devices beheersen. De GUI laat ook toe om een overzicht te zien van wat er door elke steward gemeten wordt.

4.1 Overview

4.1.1 Class

panel-main : Het panel waarmee de user kan interageren.

4.2 panel-main

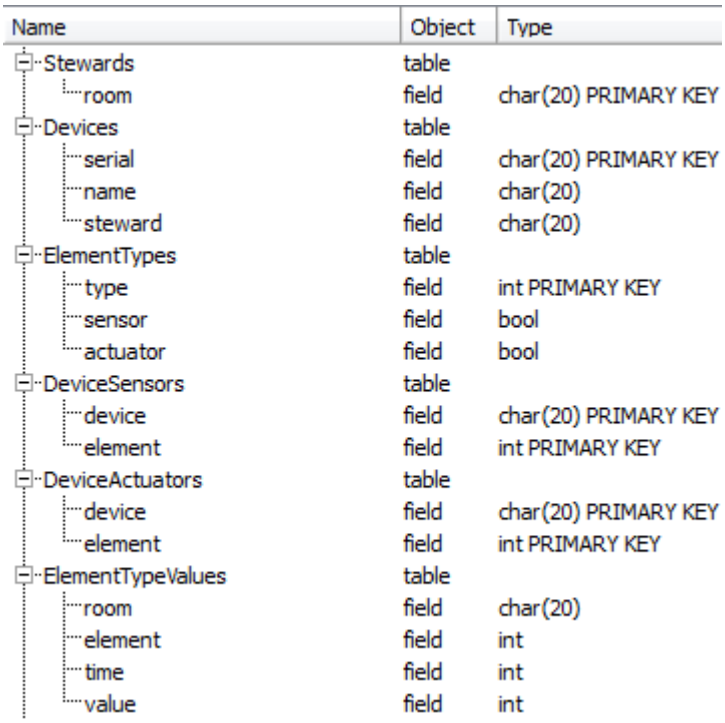
Het panel waarmee de user kan interageren. Deze panel heeft een overzicht, een menu voor het toevoegen van nieuwe devices en een menu voor het toevoegen van nieuwe stewards.
Inherit : vertical-panel%.

5 db

Beheerst de gegevensbank van het systeem. Elke steward en device die door de user aangemaakt wordt wordt opgeslagen in de database.

De gegevensbank :

- De *Stewards* table laat toe om stewards op te slaan.
- De *Devices* table laat toe om devices op te slaan.
- De *ElementTypes* table toont welke types meetbaar en/of beinvloedbaar kunnen zijn door sensoren en actuatoren.
- De *DeviceSensors* table toont, voor elk type device (devices zijn van hetzelfde type wanneer ze dezelfde naam hebben), welke sensoren het heeft.
- De *DeviceActuators* table toont, vool elk type device (devices zijn van hetzelfde type wanneer ze dezelfde naam hebben), welke actuatoren het heeft.
- De *ElementTypeValues* table toont de waarde van een bepaalde elementtype in een bepaalde kamer op een bepaalde tijd.



Name	Object	Type
Stewards	table	
room	field	char(20) PRIMARY KEY
Devices	table	
serial	field	char(20) PRIMARY KEY
name	field	char(20)
steward	field	char(20)
ElementTypes	table	
type	field	int PRIMARY KEY
sensor	field	bool
actuator	field	bool
DeviceSensors	table	
device	field	char(20) PRIMARY KEY
element	field	int PRIMARY KEY
DeviceActuators	table	
device	field	char(20) PRIMARY KEY
element	field	int PRIMARY KEY
ElementTypeValues	table	
room	field	char(20)
element	field	int
time	field	int
value	field	int

Figure 2: Het database schema

5.1 Overview

5.1.1 Class

manager : Een database manager.

5.2 manager

Een database manager. Deze laat toe om devices en stewards gemakkelijk op te slagen, en een central-unit te krijgen gebaseerd op de gegevens van de database.

5.2.1 Constructor

(new-db-manager)→ procedure? : Maakt een nieuwe manager aan.

5.2.2 Method

(add-device device)→ void? : Voegt een device toe aan de database.

(add-steward steward)→ void? : Voegt een steward toe aan de database.

(restore-state)→ central-unit? : Geeft een central-unit terug die stewards beheert die opgeslagen waren in de database.

(new-db-manager)→ procedure?
Maakt een nieuwe manager aan.

(add-device device)→ void?
Voegt een device toe aan de database. Pas op, als de device al door een steward beheert is zal de gegevensbank proberen om die nogmaals op te slagen bij het saven van de steward.

- *device* : De device dat toegevoegd wordt aan de database.

(add-steward steward)→ void?
Voegt een steward toe aan de database. Alle devices die deze steward beheert worden ook aan de gegevensbank toegevoegd.

- *steward* : De steward dat toegevoegd wordt aand de database.

(restore-state)→ central-unit?
Geeft een central-unit terug die stewards beheert die opgeslagen waren in de database.

- *return* : Een central-unit die de stewards en devices beheert die in de gegevensbank opgeslagen waren.

6 unit-test

Bevat een aantal tests om te zien of de functionaliteiten van het programma goed zijn geimplementeerd. Meeste van de interne werking en communicatie wordt getest, voor de GUI is dit niet het geval.

6.1 Overview

6.1.1 Class

test-element : Een reeks tests op element.

test-sensor : Een reeks tests op sensor.

test-actuator : Een reeks tests op actuator.

test-device : Een reeks tests op device.

test-steward : Een reeks tests op steward.

test-instruction : Een reeks tests op instruction.

test-messenger : Een reeks tests op messenger.

test-parser : Een reeks tests op parser.

test-hardware-device : Een reeks tests op hardware-device.

test-physical-room : Een reeks tests op physical-room.