Clinical Decision Support for OpenMRS

Bierman, Robert, *Group Lead*
bierman@mail.sfsu.edu

Woeltjen, Victor, *Group Lead*
woeltjen@mail.sfsu.edu

Choi, Kay   Gimeno, Steven   Lum, Jason   Ng, Ying Kit
Uy, Bianca

May 14, 2013

Group 1: Final Project for CSC 668-868 Spring 2013
https://code.google.com/p/sp2013-csc668-868-group1/

Contents

List of Figures

List of Tables

1 Contributions

1.1 Contributions by Robert Bierman

Implemented types to represent values in DSS1. Authored Section 2 of this document. Prepared diagrams, including Figures 22 and 1.

1.2 Contributions by Kay Choi

Added HTML form entry for viral load, CD4, hemoglobin, and patient data entry. Added patient summary. Implemented DSS intrinsic read functions.

1.3 Contributions by Steven Gimeno

Wrote test cases to demonstrate the correctness of the team's implementation of the DSS1 language, as seen in Appendix B.1.

1.4 Contributions by Jason Lum

Implemented DSS intrinsic miscellaneous functions (within, length). Authored documentation for intrinsic functions, Section 5.3.4. Authored use cases shown in Section 4.

1.5 Contributions by Ying Kit Ng

Studied the Spring web MVC (Model View Controller) framework to integrate the compiler. Implemented the DSSDate intrinsic functions. Prepared screen shots for user guide, Section 3.

1.6 Contributions by Bianca Uy

Helped integrate compiler onto OpenMRS. Authored web pages for creating, modifying and uploading DSS rules. Authored text of user guide, Section 3. Wrote test cases to demonstrate the correctness of the team's implementation of the DSS1 intrinsics, as seen in Appendix B.2.

1.7 Contributions by Victor Woeltjen

Implemented DSS Rule Service, including rule storage and conversion to and from XML (Extensible Markup Language). Implemented flow control, execution context, and integrated value types into DSS Interpreter. Authored sections 5 and 7 of this document, except for subsections otherwise noted.

2 Introduction

2.1 Problem statement

It is one thing to have information readily available, it is another to understand and make the best use of that information. The OpenMRS system provides a repository of data on patients but it is still up to the physician to decide the course of treatment, tests that need to be run, and medications to be administered. The Decision Support System (DSS) is designed to assist the physician by providing alerts based on correlation of data and programmatic rules. Because of the vast amounts of data on patients, the number of medications and tests available and the variability of patient behavior, the DSS is designed to provide rules to avoid mistakes, speed patient care and optimize resources.

DSS allows doctors to create rules to alert them if they prescribe a medication that may interact with other medications that the patient is taking or may be allergic to. Or, it can suggest running tests that may be due or alert to the fact that prior tests need to be redone. Decision support is about correlating the data is a manner useful to the physician.

To create a DSS, a method must exist to specify these rules, and should be of a nature to allow non-technical individuals to create them. Once created a system to store and interpret those rules needs to be devised and finally the results of the rules need to be displayed back to the physician in an intuitive and meaningful way.

The task here is given the presented grammar for a simplified decision support language, DSS1 (described in Appendix A), create the interpreter that can store and process rules and display the results on the patient summary and dashboards in OpenMRS.

2.2 Software and platform used

This solution has been implemented as a module for OpenMRS. The DSS Interpreter has been written in Java. Client pages have been written in JavaScript, JSP, and HTML, with OpenMRS extension points indicated using XML. Interactions between the web-based client and server-side rules are supported by the Spring framework and DWR (Direct Web Remoting).

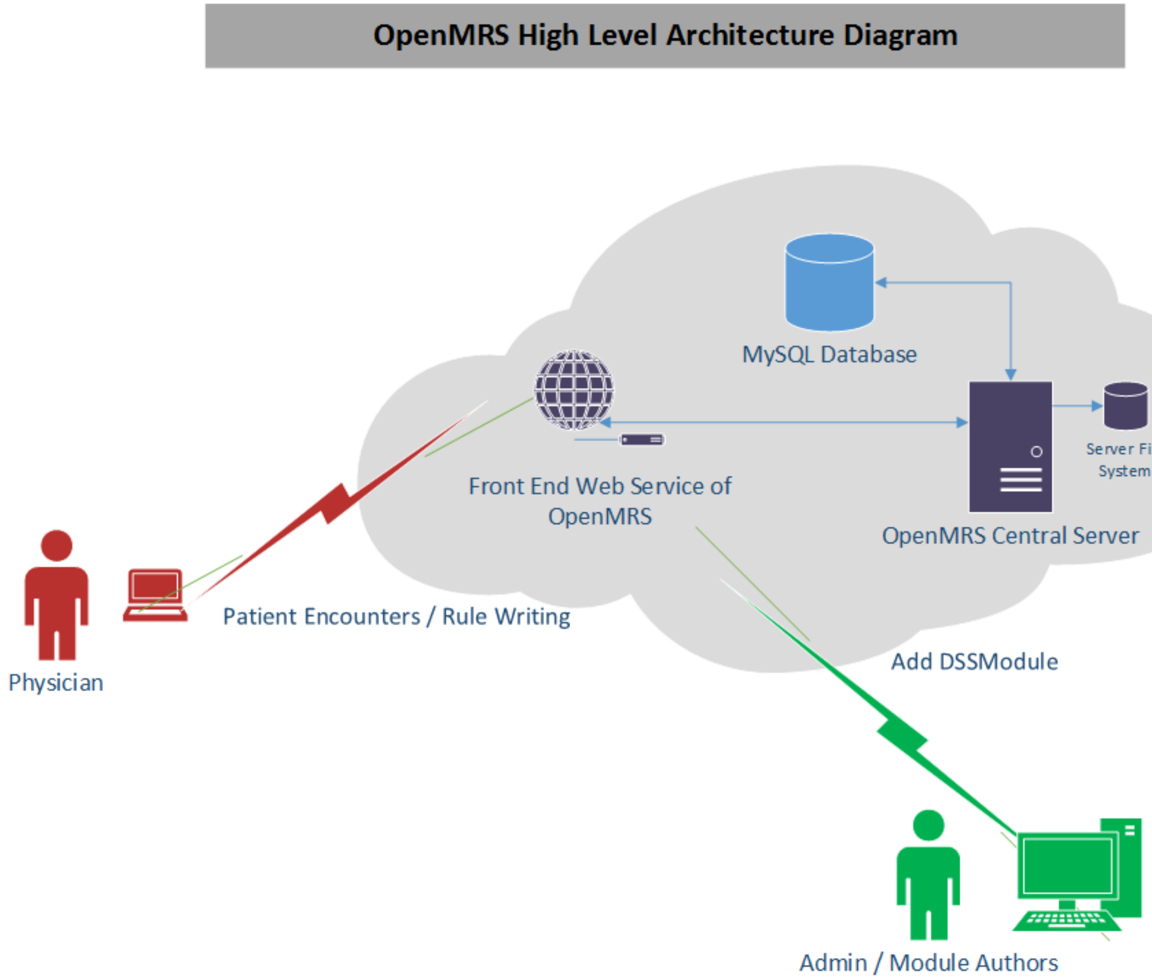

OpenMRS High Level Architecture Diagram
MySQL Database
Front End Web Service of OpenMRS
Server File System
OpenMRS Central Server
Patient Encounters / Rule Writing
Physician
Add DSSModule
Admin / Module Authors

Figure 1: Architecture Diagram

3 Decision support user guide

3.1 Installation

In order to utilize the presented Decision Support System, and administrator must first install the module into OpenMRS. This can be done through the OpenMRS web interface by navigating to the "Administration" menu and following the link "Manage Modules." From there, click "Add or Upgrade Module"; under "Add Module", click browse. Then, locate and select the DSS Module file `dssmodule.omod`. Once this has been selected, click "Upload." OpenMRS should then install and start the module, and notify you of any issues with the installation process.

3.2 First steps

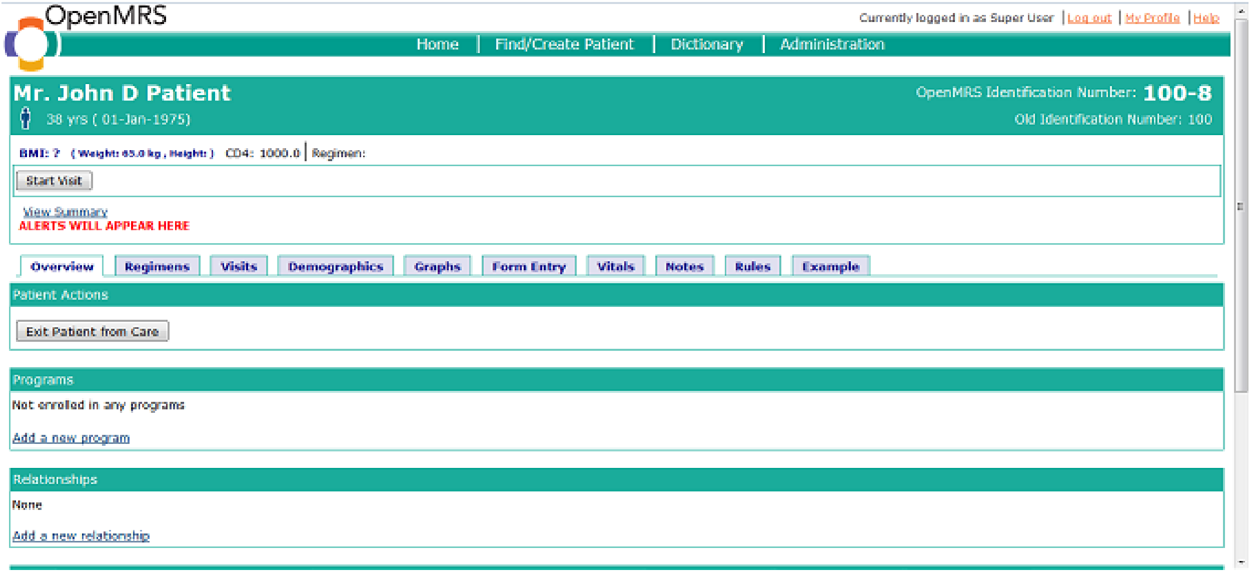
Log onto OpenMRS with your corresponding username and password. Passwords are case-sensitive. Once logged in, you can search or create a patient using the tab 'Find/Create Patient'. Once you have selected the patient, their profile will appear.

3.3 Patient profile

Patient profile has several tabs Overview, Regimens, Visits, Demographics, Graphs, Form Entry, Example, Vitals, and Notes. There is also a patient summary link and underneath are any alerts that a physician may need to be aware about the patient, as shown in Figure 2.

3.3.1 Form entry

There are four different forms available Vitals, Viral, Hemoglobin, and CD4 (shown in Figure 3, 4, 5, and 6). Viral, Hemoglobin, and CD4 are for lab results. The encounter details must be filled along with the data entry. The Vitals form is an assessment of the current condition of the patient. Once the form is completed, press Enter Form and all information will be saved. The forms from the last three encounters are available for viewing; for a more complete listing, check Visits tab.



OpenMRS
Currently logged in as Super User | Log out | My Profile | Help
Home | Find/Create Patient | Dictionary | Administration

Mr. John D Patient

38 yrs (01-Jan-1975)

OpenMRS Identification Number: **100-8**

Old Identification Number: 100

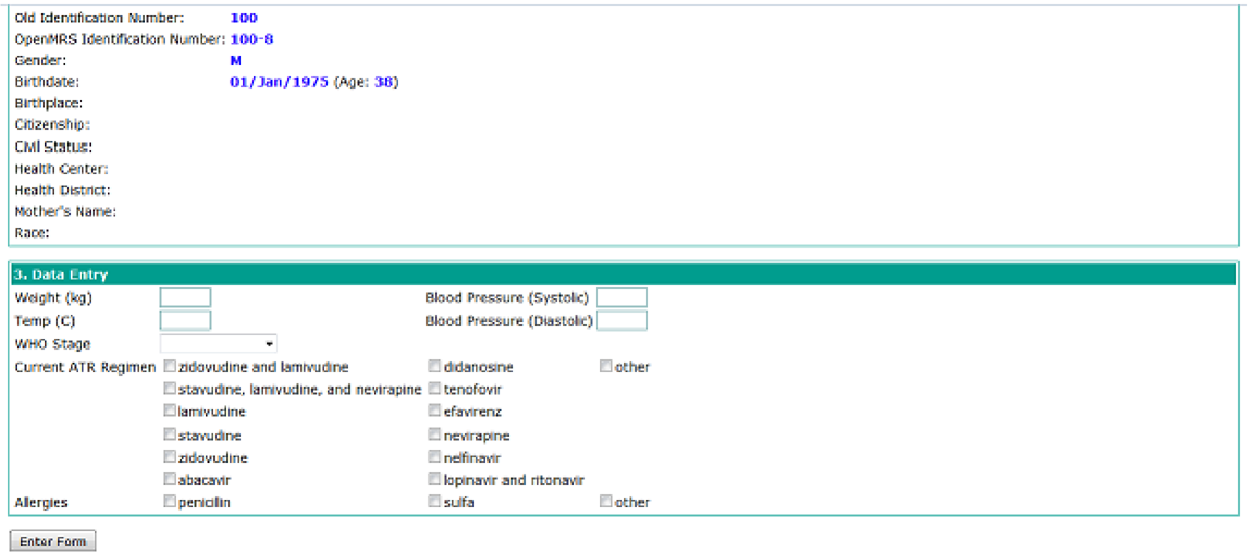
BMI: ? (Weight: 65.0 kg , Height:) CD4: 1000.0 | Regimen:

ALERTS WILL APPEAR HERE

Not enrolled in any programs

None

Figure 2: Patient profile



Old Identification Number:	100
OpenMRS Identification Number:	100-8
Gender:	M
Birthdate:	01/Jan/1975 (Age: 38)
Birthplace:	
Citizenship:	
CMI Status:	
Health Center:	
Health District:	
Mother's Name:	
Race:	

3. Data Entry

Weight (kg) ☐ Blood Pressure (Systolic) ☐

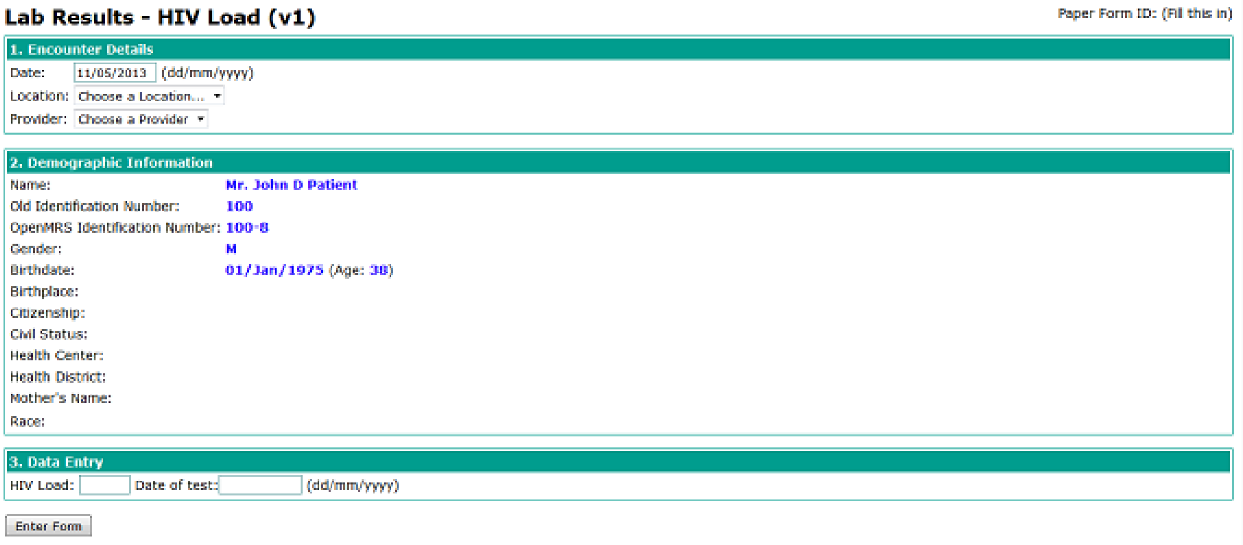
Temp (C) ☐ Blood Pressure (Diastolic) ☐

WHO Stage ▾

Current ATR Regimen ☐ zidovudine and lamivudine ☐ didanosine ☐ other
☐ stavudine, lamivudine, and nevirapine ☐ tenofovir
☐ lamivudine ☐ efavirenz
☐ stavudine ☐ nevirapine
☐ zidovudine ☐ nelfinavir
☐ abacavir ☐ lopinavir and ritonavir

Allergies ☐ penicillin ☐ sulfa ☐ other

Figure 3: Vitals form



Lab Results - HIV Load (v1)

Paper Form ID: (Fill this in)

1. Encounter Details

Date: 11/05/2013 (dd/mm/yyyy)

Location: Choose a Location... ▾

Provider: Choose a Provider ▾

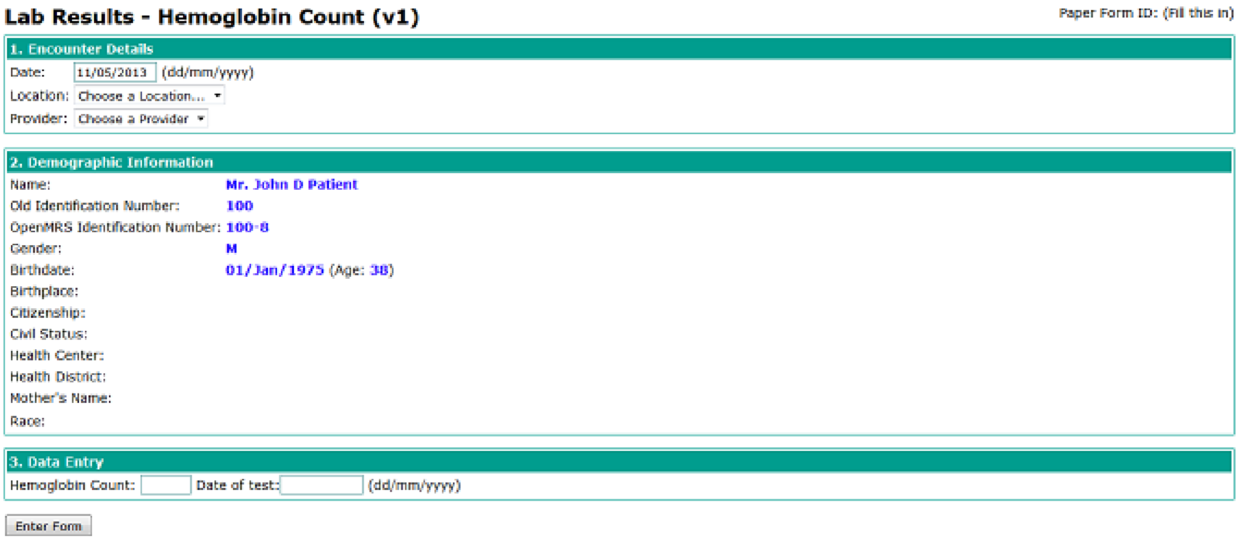
2. Demographic Information

Name:	Mr. John D Patient
Old Identification Number:	100
OpenMRS Identification Number:	100-8
Gender:	M
Birthdate:	01/Jan/1975 (Age: 38)
Birthplace:	
Citizenship:	
Civil Status:	
Health Center:	
Health District:	
Mother's Name:	
Race:	

3. Data Entry

HIV Load: ____ Date of test: ____ (dd/mm/yyyy)

Figure 4: Viral form



Lab Results - Hemoglobin Count (v1)

Paper Form ID: (Fill this in)

1. Encounter Details

Date: 11/05/2013 (dd/mm/yyyy)

Location: Choose a Location... ▾

Provider: Choose a Provider ▾

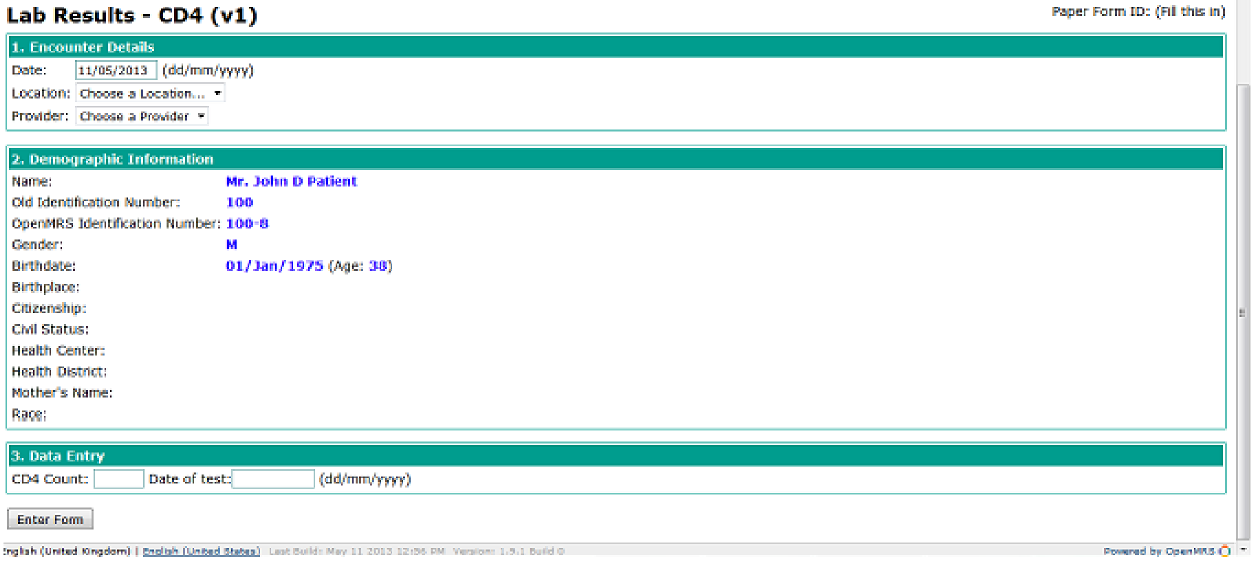
2. Demographic Information

Name:	Mr. John D Patient
Old Identification Number:	100
OpenMRS Identification Number:	100-8
Gender:	M
Birthdate:	01/Jan/1975 (Age: 38)
Birthplace:	
Citizenship:	
Civil Status:	
Health Center:	
Health District:	
Mother's Name:	
Race:	

3. Data Entry

Hemoglobin Count: ____ Date of test: ____ (dd/mm/yyyy)

Figure 5: Hemoglobin form



Lab Results - CD4 (v1)

Paper Form ID: (Fill this in)

1. Encounter Details

Date:	11/05/2013 (dd/mm/yyyy)
Location:	Choose a Location... ▾
Provider:	Choose a Provider ▾

2. Demographic Information

Name:	Mr. John D Patient
Old Identification Number:	100
OpenMRS Identification Number:	100-8
Gender:	M
Birthdate:	01/Jan/1975 (Age: 38)
Birthplace:	
Citizenship:	
Civil Status:	
Health Center:	
Health District:	
Mother's Name:	
Race:	

3. Data Entry

CD4 Count: ☐ Date of test: ☐ (dd/mm/yyyy)

Enter Form
nglish (United Kingdom) | English (United States) Last Build: May 11 2013 12:56 PM Version: 1.9.1 Build 0
Powered by OpenMRS

Figure 6: CD4 form

3.3.2 Vitals tab

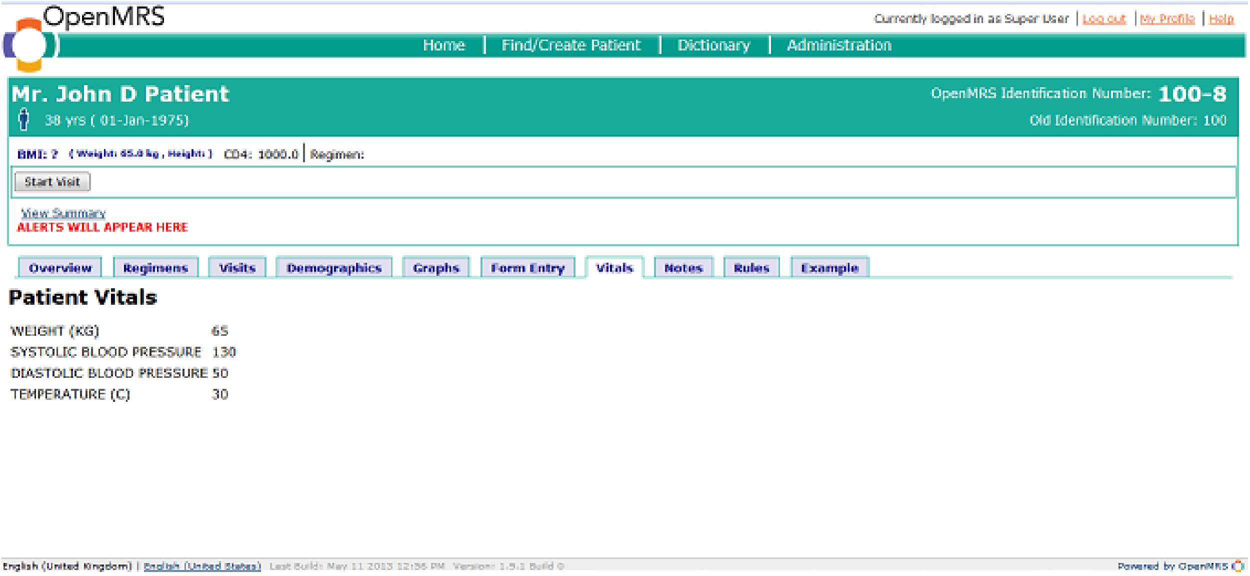
Displays the weight, systolic blood pressure, diastolic blood pressure, and temperature of the patient based the latest encounter without having to pull out the form. This tab is shown in Figure 7.

3.3.3 Patient summary

Includes allergies, all ART regimen drugs, who stage, and TB status; it also includes the vitals of the patient from the first encounter and the last encounter; and all lab results viral, hemoglobin, and cd4. Any alerts that a physician may need to be aware of are displayed in the bottom of the page in large bold letters, as shown in Figure 8. These information are all based on patient encounters.

3.3.4 Alerts

Alerts are generated based on DSS rules on OpenMRS which utilizes information from patient encounters. They can be found in two locations patient dashboard and patient summary. Figure 2 and 8 show where the alerts will appear on the patient dashboard and patient


OpenMRS
Currently logged in as Super User | Log out | My Profile | Help
Home | Find/Create Patient | Dictionary | Administration
Start Visit
View Summary
Overview
Regimens
Visits
Demographics
Graphs
Form Entry
Vitals
Notes
Rules
Example
English (United Kingdom) | English (United States) Last Build: May 11 2013 12:56 PM Version: 1.9.1 Build 0
Powered by OpenMRS

Mr. John D Patient

38 yrs (01-Jan-1975)

OpenMRS Identification Number: **100-8**

Old Identification Number: 100

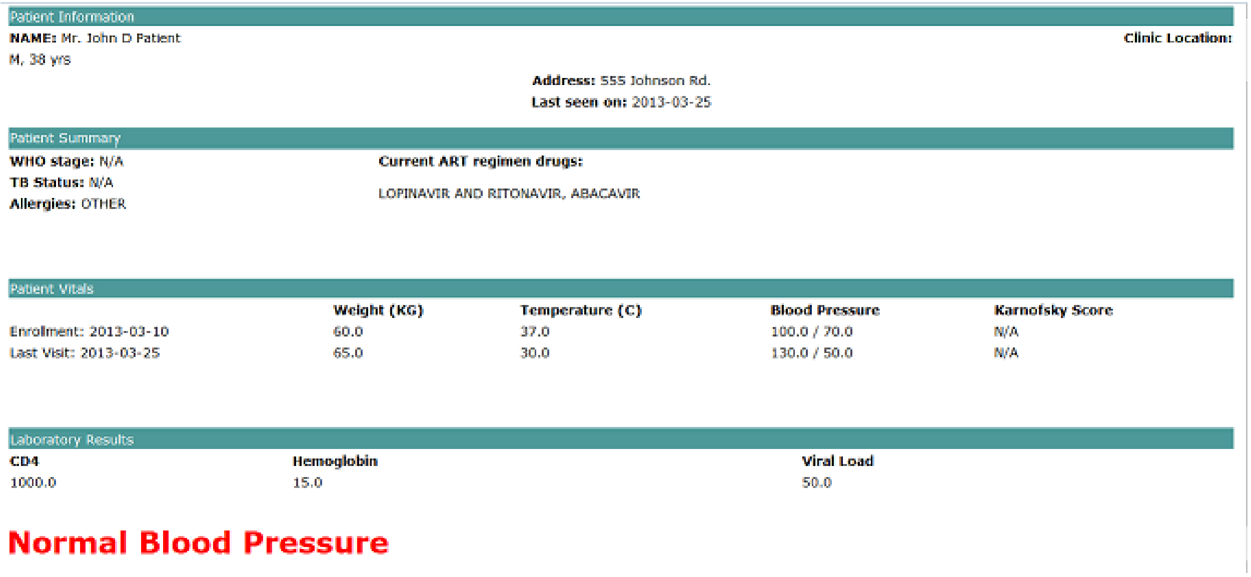
BMI: ? (Weight: 65.0 kg , Height:) CD4: 1000.0 | Regimen:

ALERTS WILL APPEAR HERE

Patient Vitals

WEIGHT (KG)	65
SYSTOLIC BLOOD PRESSURE	130
DIASTOLIC BLOOD PRESSURE	50
TEMPERATURE (C)	30

Figure 7: Vitals tab, displaying weight, blood pressure, and temperature based on latest encounter.



Patient Information

NAME: Mr. John D Patient

M, 38 yrs

Clinic Location:

Address: 555 Johnson Rd.

Last seen on: 2013-03-25

Patient Summary

WHO stage: N/A

TB Status: N/A

Allergies: OTHER

Current ART regimen drugs:

LOPINAVIR AND RITONAVIR, ABACAVIR

Patient Vitals

	Weight (KG)	Temperature (C)	Blood Pressure	Karnofsky Score
Enrollment: 2013-03-10	60.0	37.0	100.0 / 70.0	N/A
Last Visit: 2013-03-25	65.0	30.0	130.0 / 50.0	N/A

Laboratory Results

CD4	Hemoglobin	Viral Load
1000.0	15.0	50.0

Normal Blood Pressure

Figure 8: Patient summary

summary, they appear in bold red letters.

3.4 Rule administration

Under Administration and under the heading DSS Compiler there are two links that provides you a way to create new rules onto OpenMRS, shown in Figure 9. Once rules are loaded onto the system, they cannot be deleted. A loading image may appear while the rules are rendering prohibiting you from using the page, it will disappear once it is done.

There are two options for managing rules: The link ”Upload a DSS File” takes you to pages described in Section 3.4.1, and ”Create DSS File” takes you to the page shown in Figure 10 and described in Sections 3.4.2, 3.4.3, and 3.4.4.

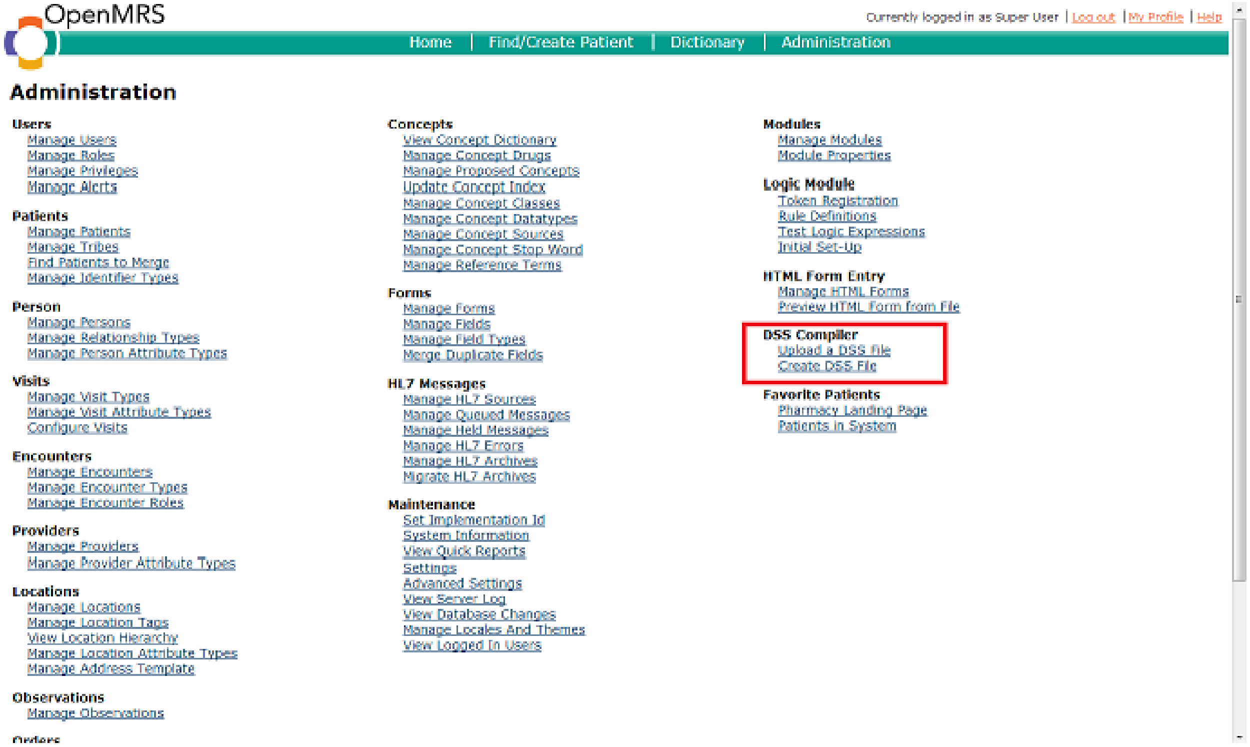


Figure 9: Administration page

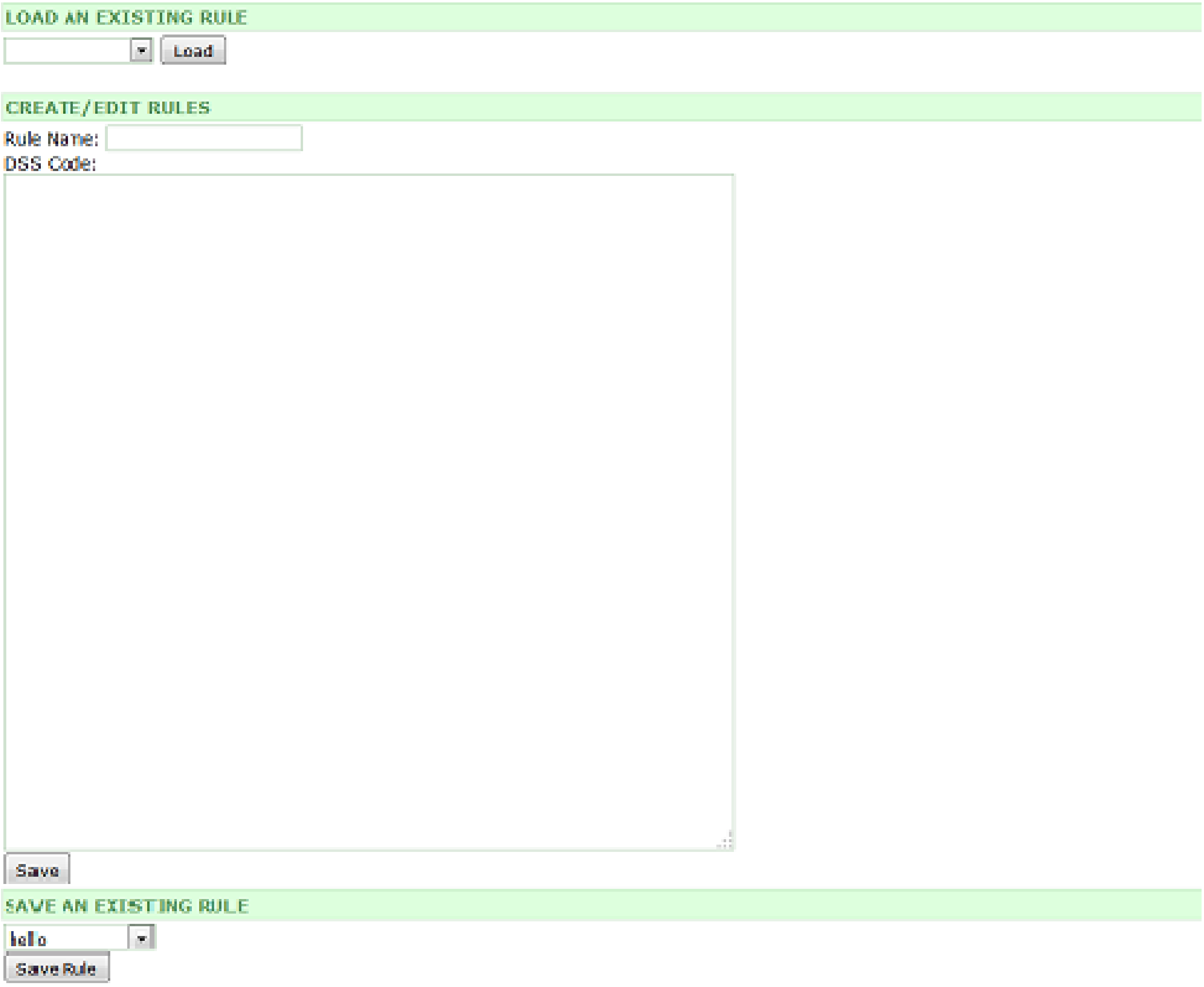
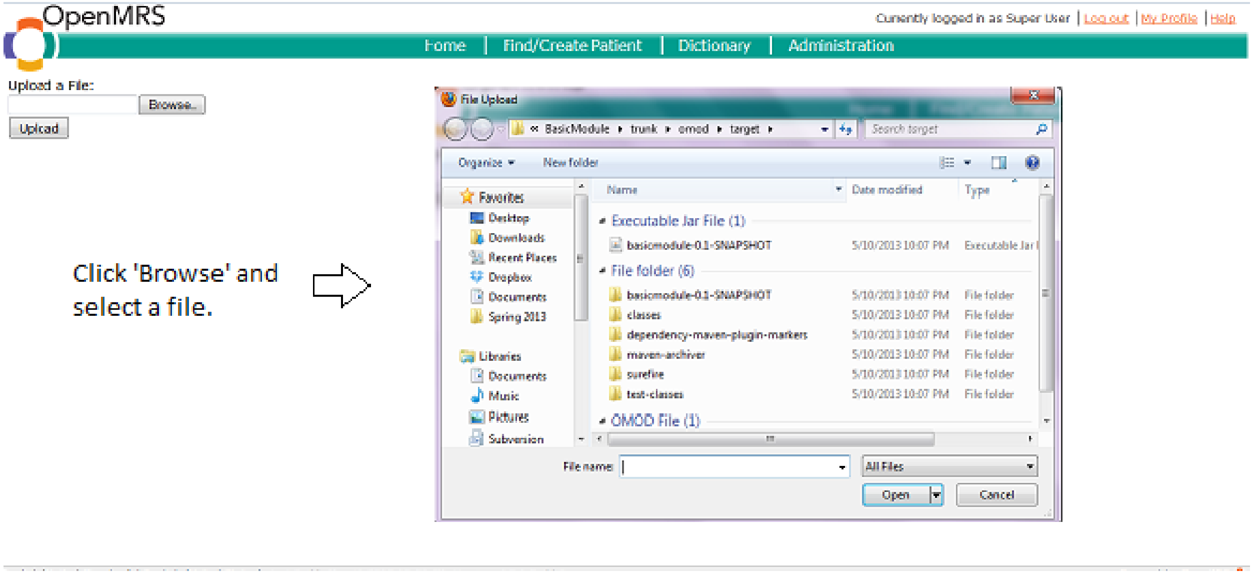


Figure 10: Create/Modify DSS Rule

3.4.1 Upload a DSS rule

You can upload an existing .DSS file onto OpenMRS. Simply click Browse and locate the file you want to upload, select it, and click upload, as seen in Figure 11.

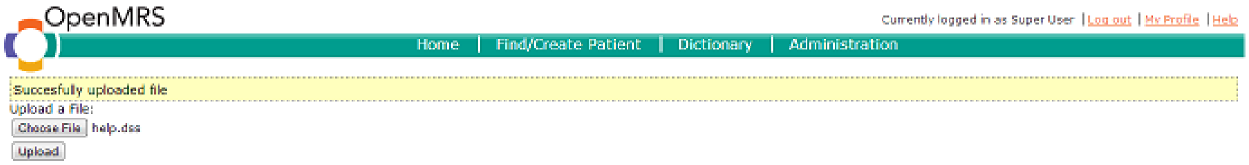
If there are any errors, it will be reported back and the file would not have been saved. Otherwise, the rule will be executing, as indicated in Figure 12.



Name	Date modified	Type
Executable Jar File (1)		
basicmodule-0.1-SNAPSHOT	5/10/2013 10:07 PM	Executable Jar I
File folder (6)		
basicmodule-0.1-SNAPSHOT	5/10/2013 10:07 PM	File folder
classes	5/10/2013 10:07 PM	File folder
dependency-maven-plugin-markers	5/10/2013 10:07 PM	File folder
maven-archiver	5/10/2013 10:07 PM	File folder
surefire	5/10/2013 10:07 PM	File folder
test-classes	5/10/2013 10:07 PM	File folder
OMOD File (1)		

Click 'Browse' and select a file.

Figure 11: DSS rule upload


OpenMRS
Currently logged in as Super User | Log out | My Profile | Help
Home | Find/Create Patient | Dictionary | Administration
Choose File
Upload

Succesfully uploaded file

Upload a File: help.dss

Figure 12: DSS rule upload confirmation

3.4.2 Load or modify an existing rule

Select an existing rule from the drop-down menu that you would like to modify then click 'Load', as seen in Figure 13. It will populate the textboxes under the Create/Edit Rule section, as shown in Figure 14.

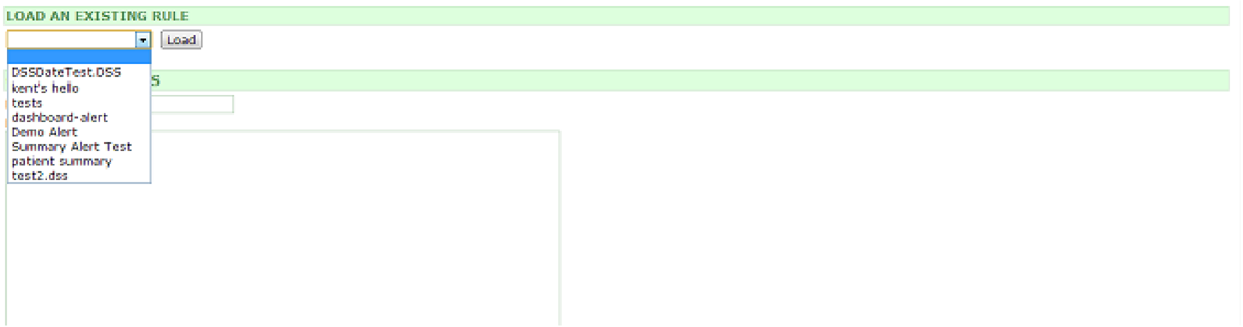
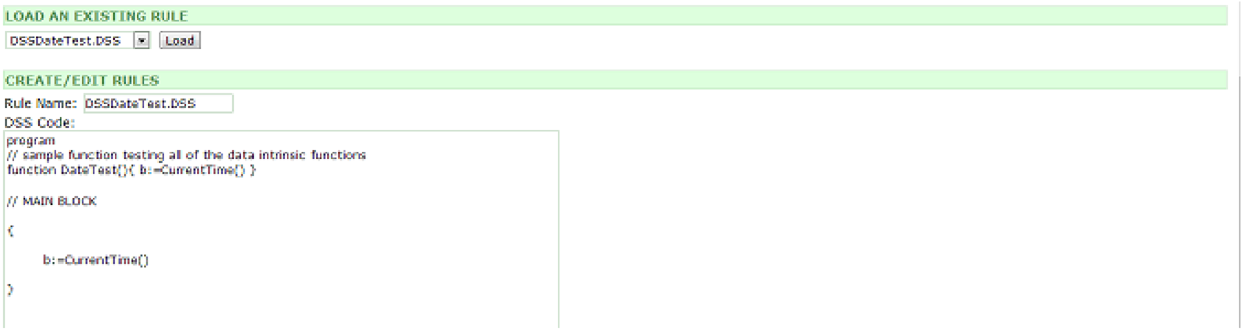


Figure 13: Loading an existing rule using the drop down menu



Rule Name: DSSDateTest.DSS

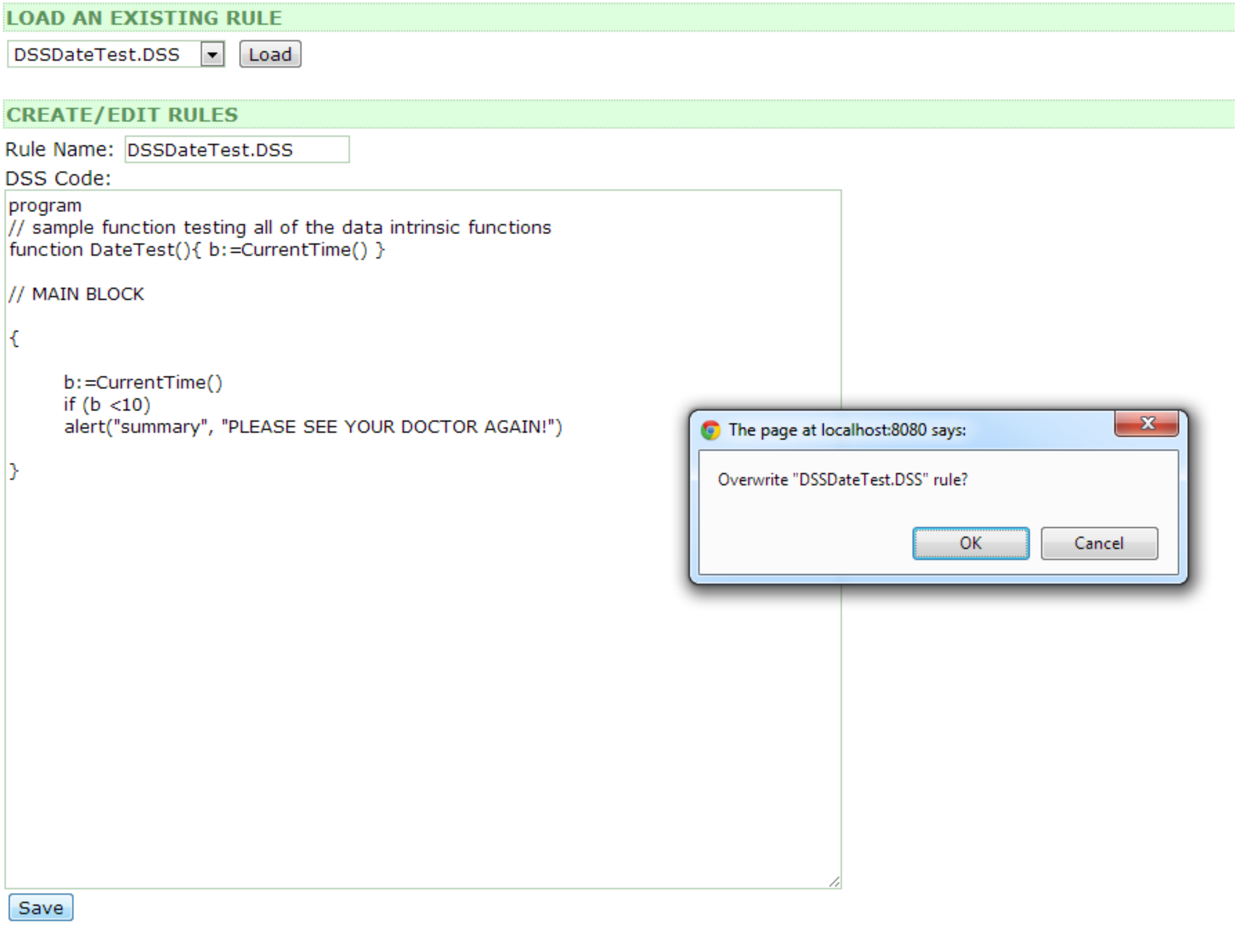
DSS Code:

```
program
// sample function testing all of the data intrinsic functions
function DateTest(){ b:=CurrentTime() }

// MAIN BLOCK

{

      b:=CurrentTime()

}
```

Figure 14: Loaded rule displayed in text area

Edit the file. Once complete, click 'Save'. A confirmation message, shown in Figure 15, will appear to ensure that you would like to overwrite an existing rule. Click yes to save the changes.

A 'Successfully uploaded file' message, as shown in Figure 16, will appear if there were no errors in the program. If there are any errors, it will be reported back at the top of the page, as in Figure 17, and it would not have saved the changes. The code will still be available for you to work on it.



LOAD AN EXISTING RULE

DSSDateTest.DSS

CREATE/EDIT RULES

Rule Name: DSSDateTest.DSS

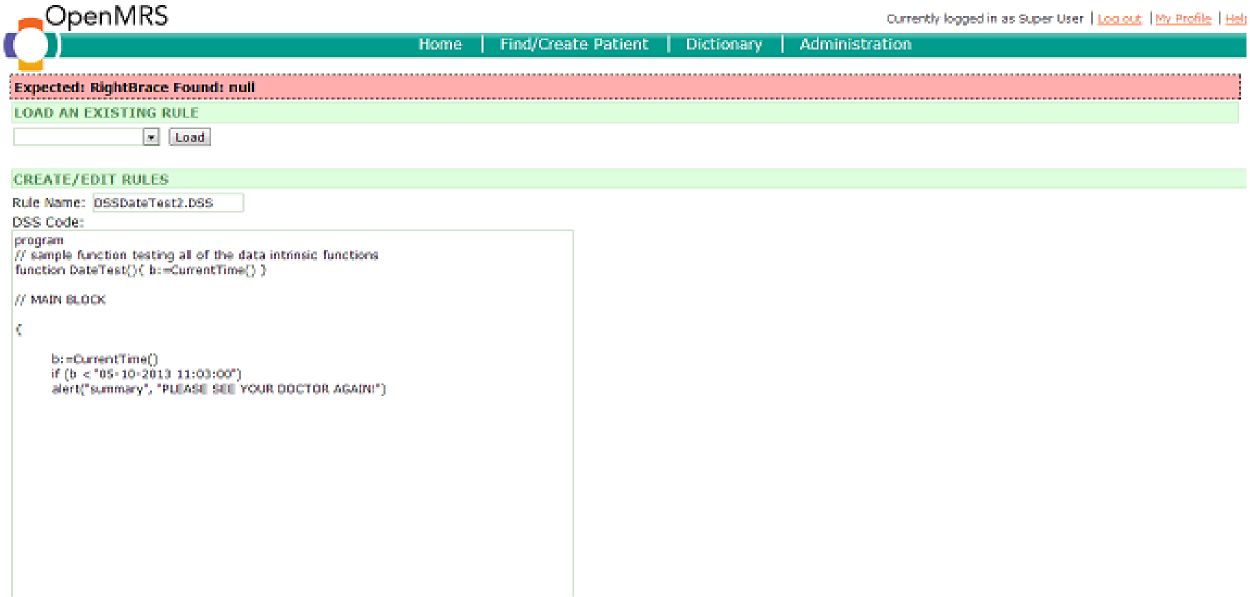
DSS Code:

```
program
// sample function testing all of the data intrinsic functions
function DateTest(){ b:=CurrentTime() }

// MAIN BLOCK

{

        b:=CurrentTime()
        if (b <10)
        alert("summary", "PLEASE SEE YOUR DOCTOR AGAIN!")

}
```

The page at localhost:8080 says:

Overwrite "DSSDateTest.DSS" rule?

Figure 15: Confirmation message, shown before over-writing an existing rule


OpenMRS
Currently logged in as Super User | Log out | My Profile | Hel
Home | Find/Create Patient | Dictionary | Administration
LOAD AN EXISTING RULE
Load
CREATE/EDIT RULES

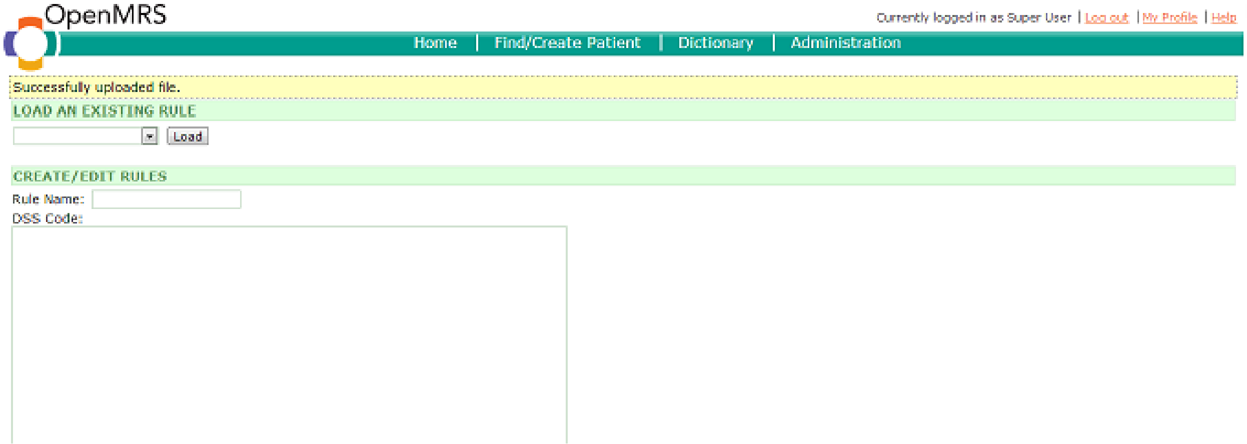
Expected: RightBrace Found: null

Rule Name: DSSDateTest2.DSS

DSS Code:

```
program
// sample function testing all of the data intrinsic functions
function DateTest(){ b:=CurrentTime() }

// MAIN BLOCK

{

        b:=CurrentTime()
        if (b < "05-10-2013 11:03:00")
        alert("summary", "PLEASE SEE YOUR DOCTOR AGAIN!")
```

Figure 16: Error message shown when DSS code cannot be compiled


OpenMRS
Currently logged in as Super User | Log out | My Profile | Help
Home | Find/Create Patient | Dictionary | Administration
LOAD AN EXISTING RULE
Load
CREATE/EDIT RULES

Successfully uploaded file.

Rule Name:

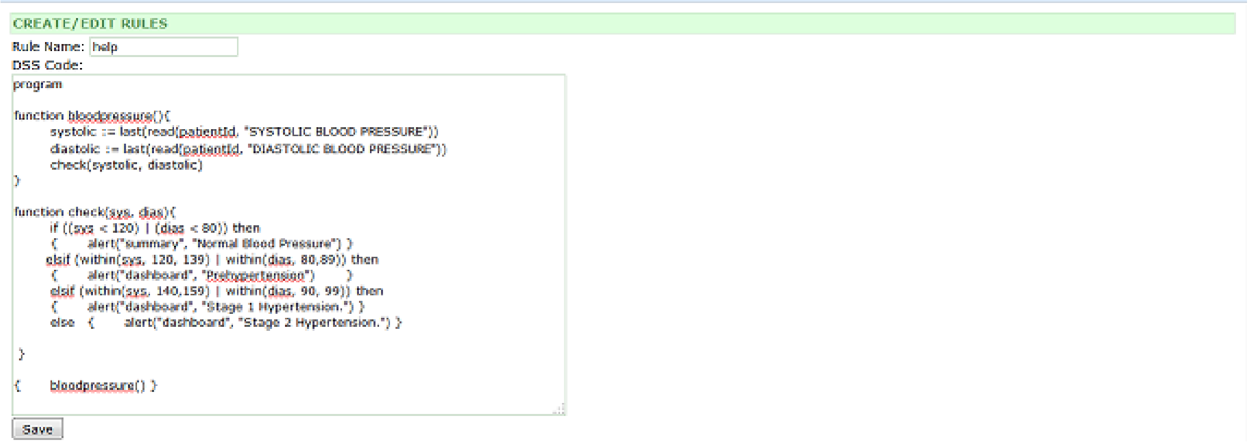
DSS Code:

Figure 17: Confirmation message shown when DSS code has compiled successfully

3.4.3 Create a rule

You can create your own rule, it should not match an existing rule. Enter a rule name for the rule and enter the program under the textbox 'DSS Code' Once finished, click 'Save', as shown in Figure 18.

A 'Successfully uploaded file' message will appear if there were no errors in the program and the rule name will become available in the drop down menu, as shown in Figure 19. If there are any errors, it will be reported back at the top of the page and it would not have saved the changes. The code will still be available for you to work on it.



Rule Name: help

DSS Code:

```
program

function bloodpressure(){
        systolic := last(read(patientId, "SYSTOLIC BLOOD PRESSURE"))
        diastolic := last(read(patientId, "DIASTOLIC BLOOD PRESSURE"))
        check(systolic, diastolic)
}

function check(sys, dias){
        if ((sys < 120) | (dias < 80)) then
        {       alert("summary", "Normal Blood Pressure") }
       elsif (within(sys, 120, 139) | within(dias, 80,89)) then
        {       alert("dashboard", "Prehypertension")       }
        elsif (within(sys, 140,159) | within(dias, 90, 99)) then
        {       alert("dashboard", "Stage 1 Hypertension.") }
        else   {       alert("dashboard", "Stage 2 Hypertension.") }

 }

{       bloodpressure() }
```

Figure 18: Creating a new rule

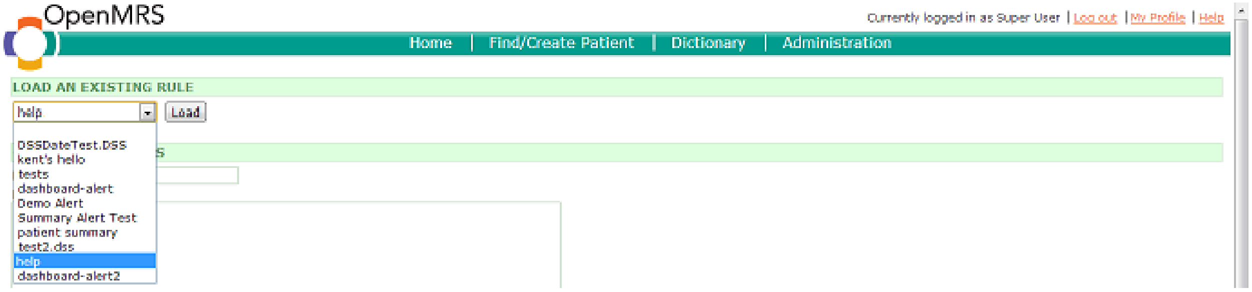


Figure 19: New rule after submission, shown in drop down menu

3.4.4 Save an existing rule

An existing rule can be saved on to your local file system. Choose the rule that you would like to save from the drop-down menu, as shown in Figure 20. Click 'Save Rule' and a download attachment window should appear, as shown in Figure 21. You can choose to either open the file or save the file. The file will be saved to your designated Downloads folder.

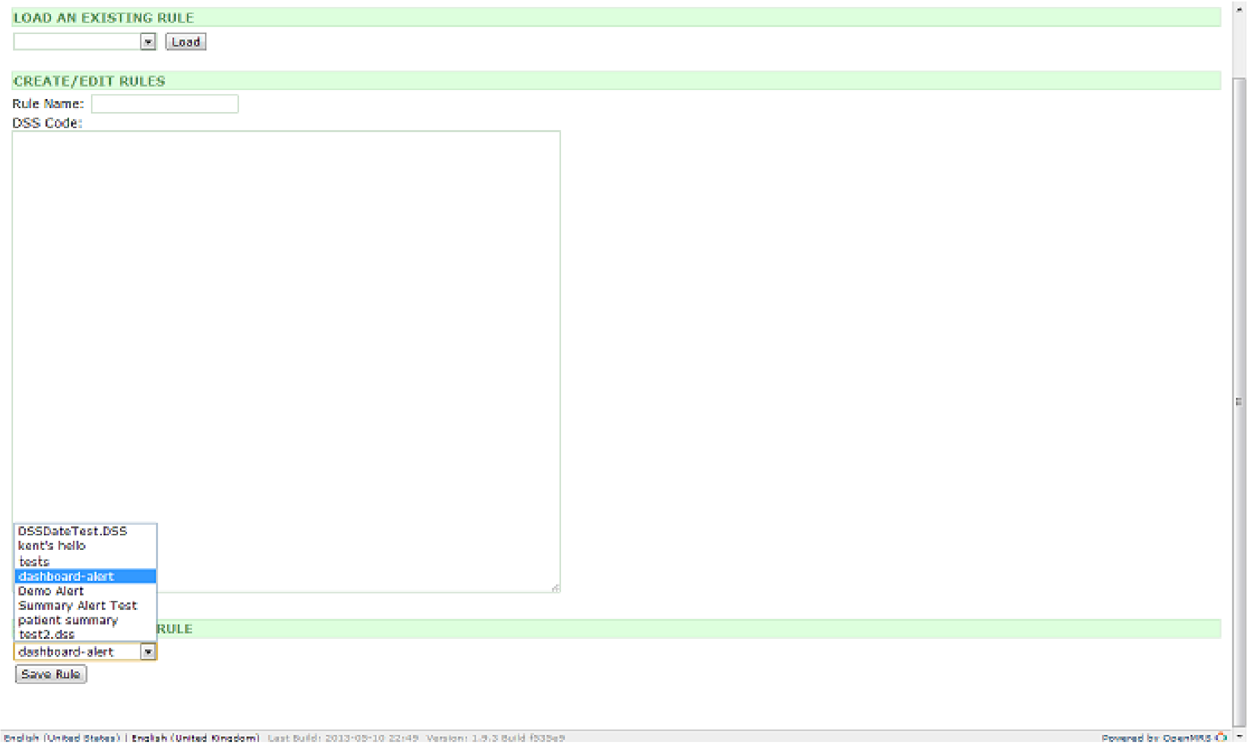
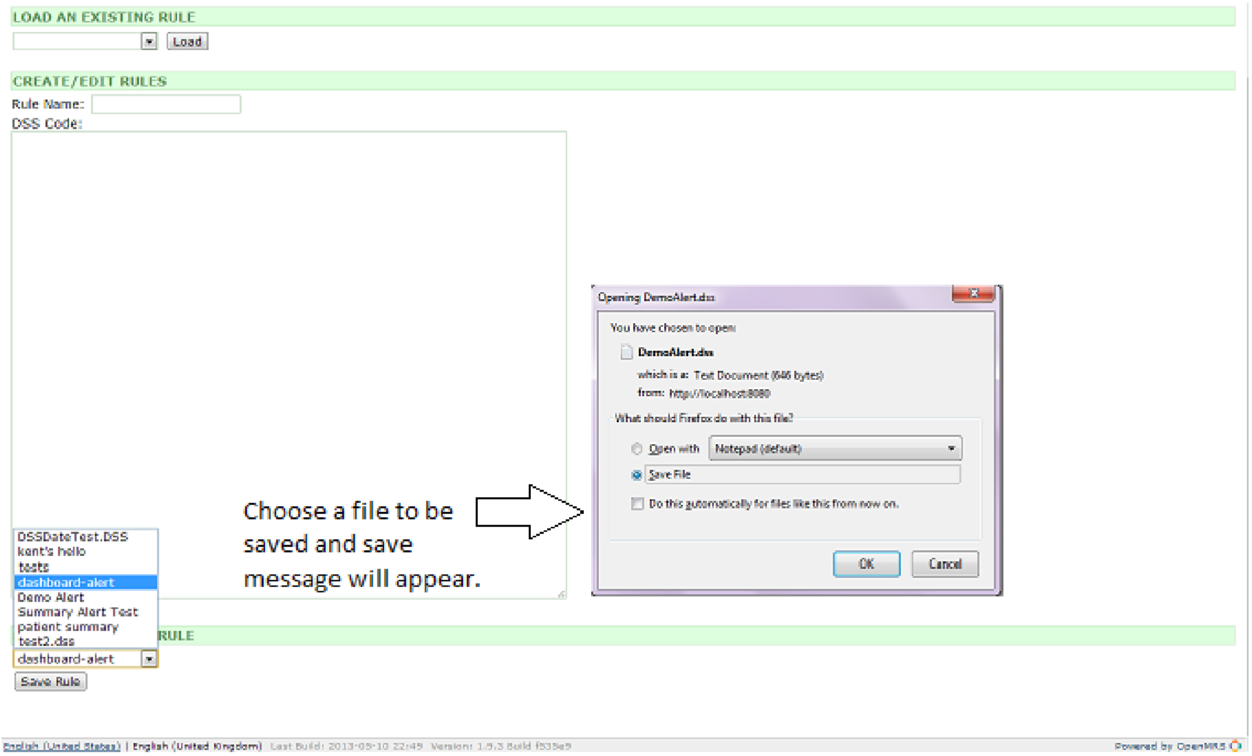


Figure 20: Choosing a rule to save locally



Rule Name:
DSS Code:

DSSDateTest.DSS
kent's hello
tests
dashboard-alert
Demo Alert
Summary Alert Test
patient summary
test2.dss

dashboard-alert

Opening DemoAlert.dss

You have chosen to open:

DemoAlert.dss

which is a: Text Document (646 bytes)
from: http://localhost:8080

What should Firefox do with this file?

○ Open with Notepad (default)

● Save File

□ Do this automatically for files like this from now on.

English (United States) | English (United Kingdom) Last Build: 2013-05-10 22:49 Version: 1.9.3 Build f535e9

Powered by OpenMRS

Choose a file to be saved and save message will appear.

Figure 21: Browser dialog shown when attempting to save rule

Use case	1
Use case name	Create Rule
Summary	Administrator or physician can create, load, or edit rules.
Dependency	Knowledge of source code
Actor	Any administrator or physician
Precondition	Default settings
Description	Rule author navigates to Create/Modify DSS Rule page. Rule author enters a rule name and DSS Code. Rule is saved. Rule author is able to edit or modify that rule.
Alternative	If errors are present, rule author is notified. Rule entry form is not cleared, allowing error to be corrected.
Postcondition	Rule is subsequently executed when navigating to patient summary or patient dashboard. Alerts produced by rule are shown where appropriate.

Table 1: Rule creation use case

4 Use cases

Two main actors may interact directly with behaviors exposed in the DSS rule module: Administrators, who are responsible for maintaining the OpenMRS instance, and physicians, who utilize OpenMRS to support their interactions with patients. Either administrators or physicians may be responsible for authoring and maintaining rules, effectively creating a third category of actors. This distinction is relevant to understanding that physicians who see and respond to the alerts produced by rules may or may not be the same individuals who author those rules. Figure 22 summarizes these interactions.

As a precondition to all use cases, an administrator is assumed to have installed the DSS rule module.

4.1 Authoring rules

As discussed, a rule author may be either a physician or some form of administrator. Rule authorship and maintenance can be summarized by two tasks: Creating rules, described in Table 1 and Figure 23, and modifying existing rules, described in Table 2 and Figure 24.

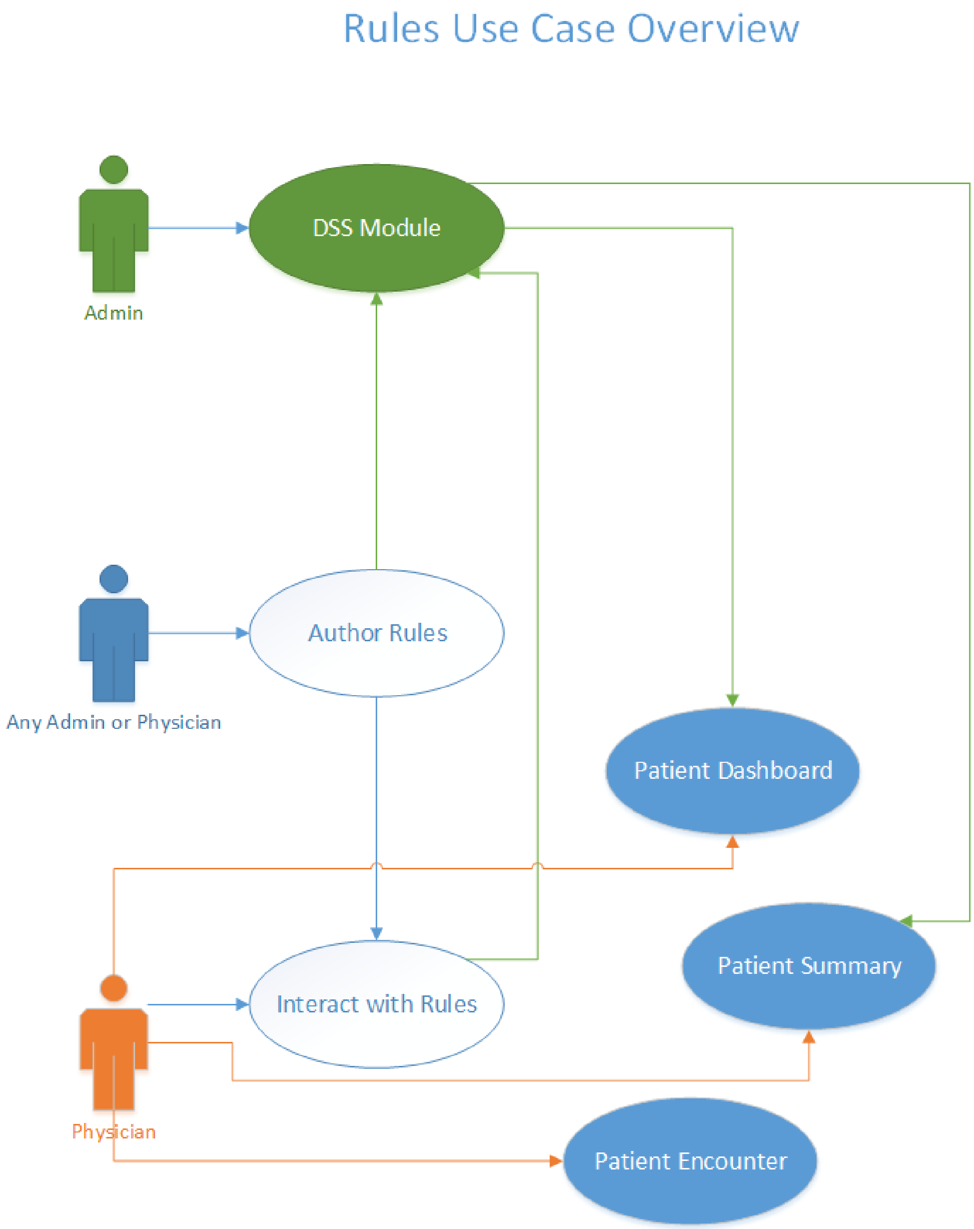


Figure 22: Use case overview

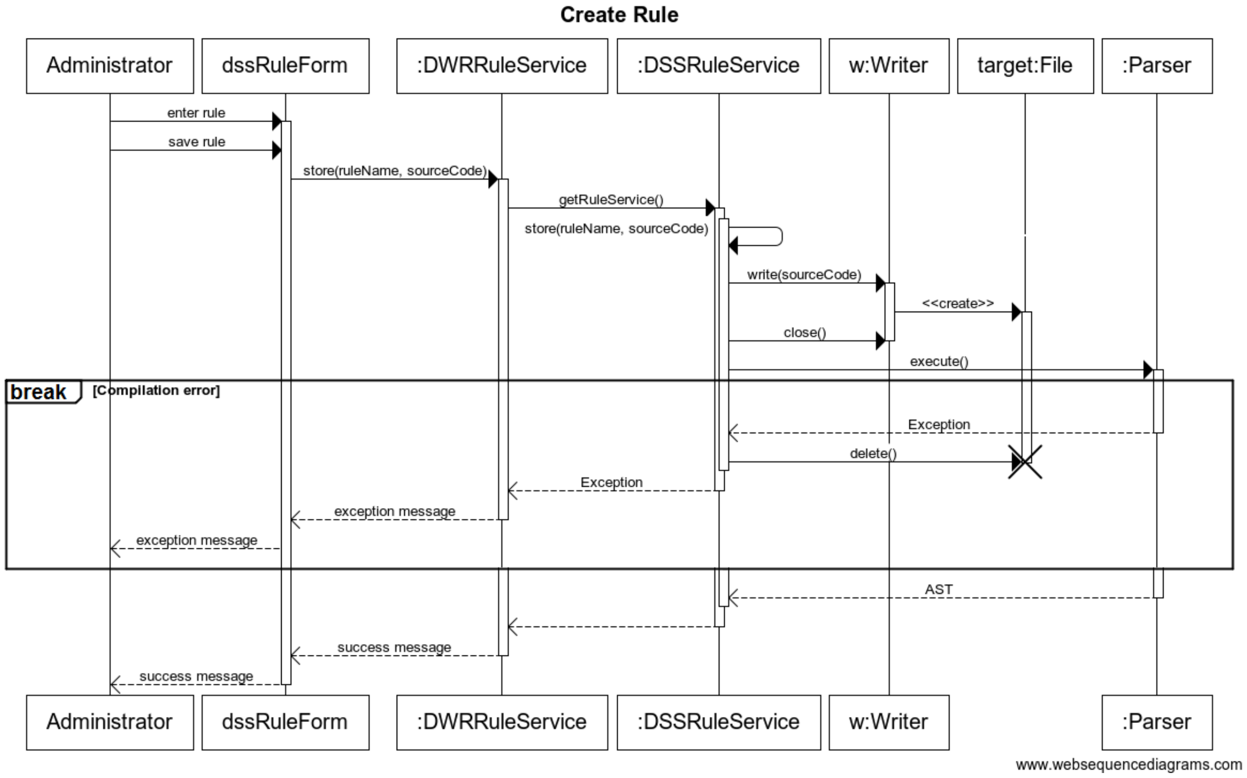


Figure 23: Rule creation sequence diagram

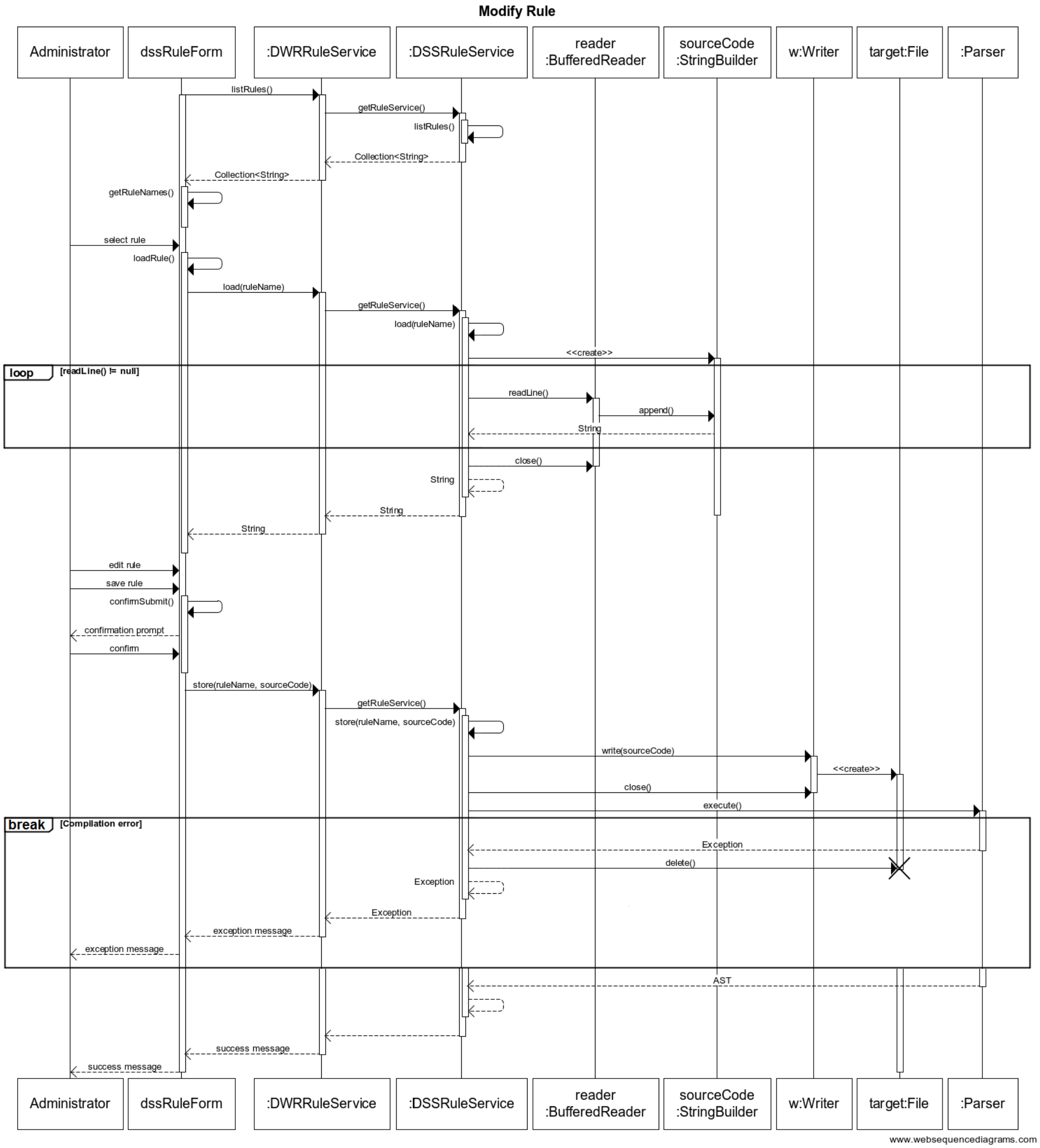


Figure 24: Rule modification sequence diagram

Use case	2
Use case name	Modify Rule
Summary	Administrator or physician can make changes to a rule.
Dependency	Knowledge of source code
Actor	Any adminstrator or physician
Precondition	An existing rule previously saved
Description	Rule author navigates to Create/Modify DSS Rule page. Rule author selects name of rule from drop down and hits ”Load.” Rule author makes necessary changes to a rule within the source code. Rule author hits save and confirms the modification when prompted.
Alternative	If errors are present, rule author is notified. Rule entry form is not cleared, allowing error to be corrected.
Postcondition	Subsequent rule executions and alerts will exhibit the behavior specified in the modified rule.

Table 2: Rule modification use case

4.2 Alerts

The useful product of DSS rules is the alerts which they may generate, which can inform the actions a physician may subsequently take. Alerts are shown in two contexts: On the patient dashboard, as described in Table 3 and Figure 25, and within the patient summary, as described in Table 4 and Figure 26.

Use case	3
Use case name	Patient Dashboard
Summary	Encounter information and relevant alerts are made available for a specific patient.
Dependency	Log in to OpenMRS.
Actor	Physician
Precondition	An existing patient has been created.
Description	Physician searches for a patient and navigates to patient dashboard. Medical information recorded on the patient is accessible through tabs for greater detail. Rules for patient are executed and any alerts for the ”dashboard” target are displayed near the top of the page. A link to the Patient Summary page is provided near the top of the page.
Alternative	
Postcondition	Physician may navigate among patient dashboard tabs containing patient information. Physician may navigate to patient summary. Physician has been notified with relevant alerts produced by defined DSS rules.

Table 3: Patient dashboard use case

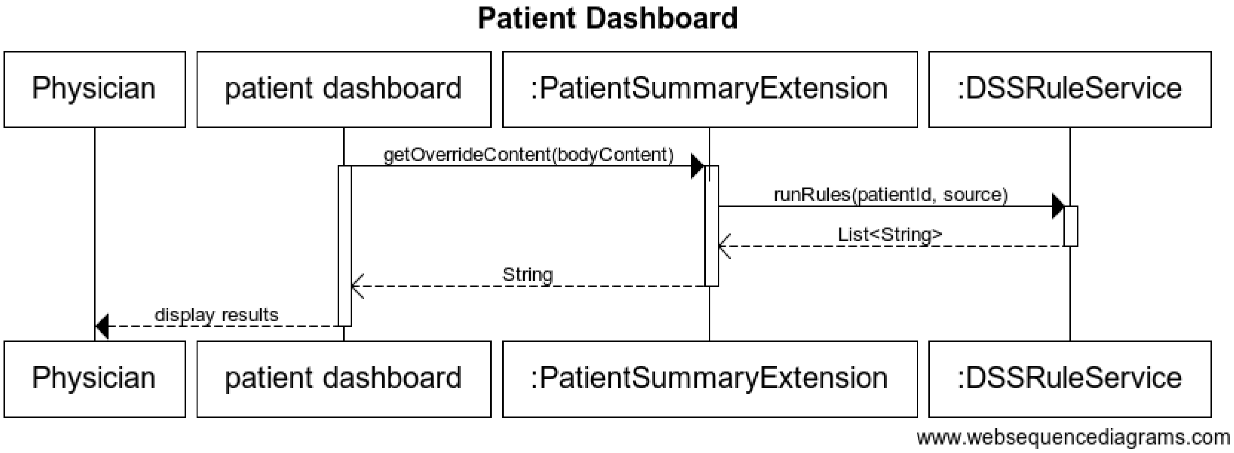


Figure 25: Patient dashboard sequence diagram

Use case	4
Use case name	Patient Summary
Summary	Major information of a patient is displayed.
Dependency	Patient Dashboard
Actor	Physician
Precondition	Relevant encounter data for the patient.
Description	Physician clicks ”Patient Summary” link from Patient Dashboard. Patient summary displays the main medical information including the WHO stage, TB Status, Allergies, and current drugs. Rules are executed, and any alerts for the ”summary” target are displayed at the bottom of the page.
Alternative	
Postcondition	Physician has summary information about the patient. Physician has been notified with relevant alerts produced by defined DSS rules.

Table 4: Patient summary use case

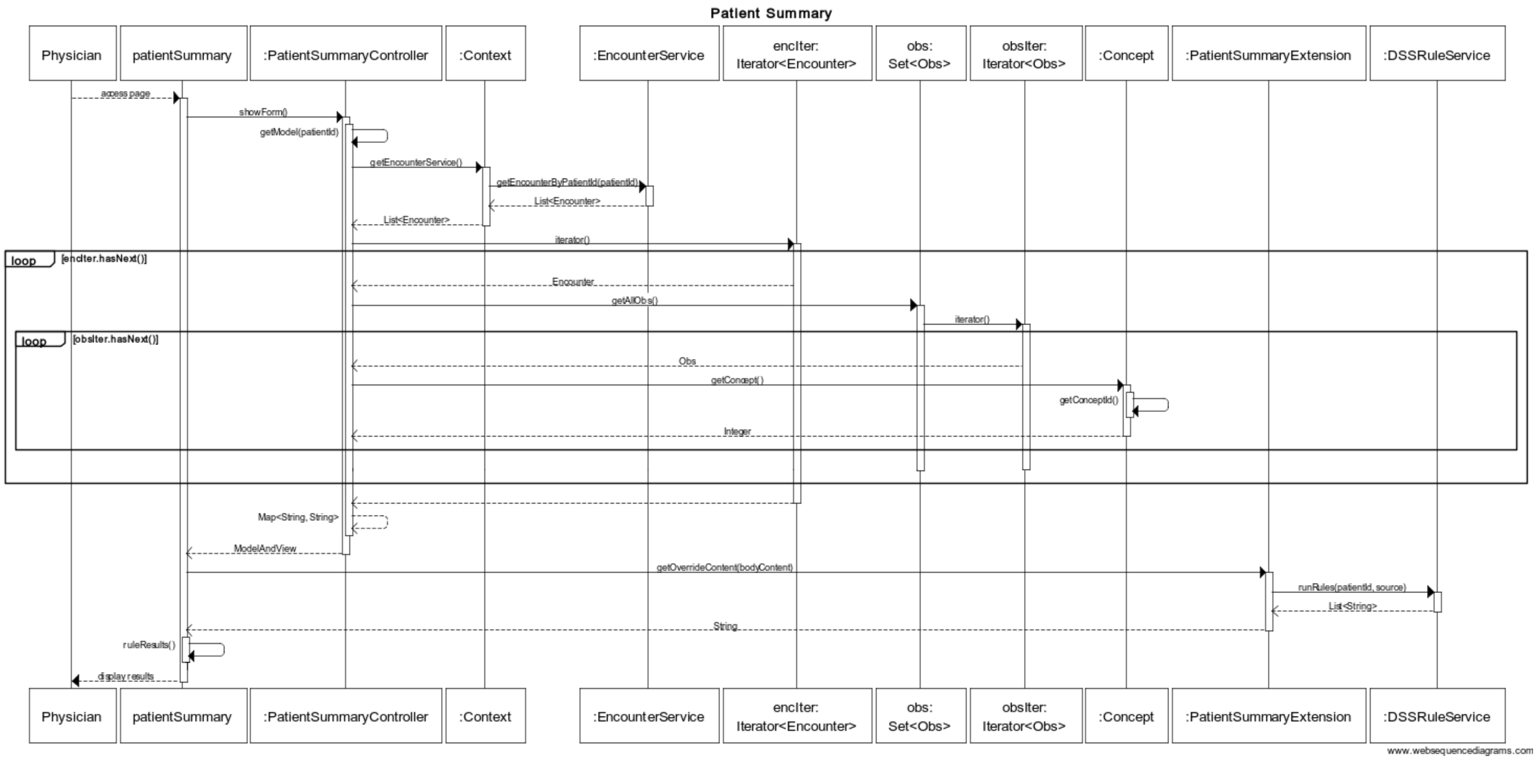


Figure 26: Patient summary sequence diagram

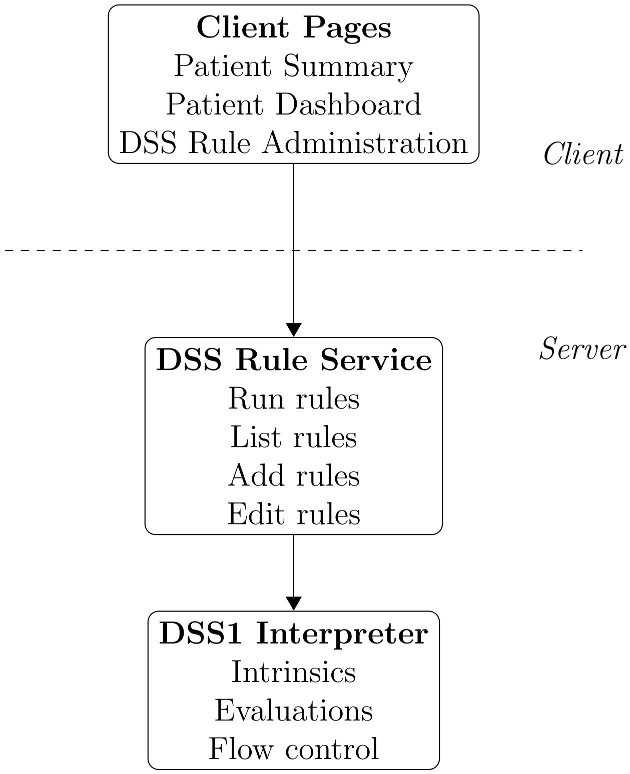


Figure 27: Usage relationship of major tiers.

5 Design overview

The DSS1 rule subsystem is incorporated into OpenMRS in a simple Client-Server fashion. The target implementation will feature client-side web pages which interact with the DSS1 rule subsystem on the server by way of DSSRuleService. Figure 27 illustrates this interaction.

The DSS Rule Service, in turn, utilizing the DSS1 Interpreter subsystem to run rules and report results.

While the DSS Rule Service runs on the server, its interface is exposed to client-side JavaScript code via DWR (Direct Web Remoting).

5.1 Client pages

Multiple client web pages interact with the DSS Rule Service.

5.1.1 Patient summary

The Patient Summary (`patientsummary.jsp`) is stand-alone page, reachable from a link on the Patient Dashboard. It is used primarily to contain major information about a patient (gender, age, WHO stage, etc.) The Patient Summary invokes the rule service via DWR to retrieve all alerts for the named target `summary` and displays them below other patient information.

5.1.2 Patient Dashboard

The Patient Dashboard is the primary landing point for viewing patient information, and contains multiple tabs for this purpose. This extension inserts a link to the Patient Summary on the Patient Dashboard, and accompanies this with relevant alerts by invoking the rule service for the `dashboard` target.

See `org.openmrs.module.basicmodule.extension.html.PatientSummaryExtension`

5.1.3 DSS Rule Administration

Create DSS Rule (`dssRules.form`) provides a form where DSS source code can be entered and submitted to the rule service with a specific rule name. Consolidates the ability to create new rules, load existing rules, and edit rules in one form. Made accessible through an extension to the Administration menu.

5.2 Rule service

The DSSRuleService follows the facade design pattern to expose important functionality to clients. The high-level tasks that are relevant to client code are defined using a few simple methods which hide the details of compiling, interpreting, and managing the storage of rules. Specific functionality is detailed in table 5.

5.2.1 Rule storage

On upload, rules are compiled to an Abstract Syntax Tree (AST) form using the provided Parser class. Once compiled successfully, the rule is stored to the OpenMRS application data directory.

Each rule is stored in two formats: Plain text source code, and an XML (Extensible Markup Language) representation of the compiled AST. The source code is subsequently used only to support user interactions (for instance, if an administrator wants to load or

Method	Description	Details
`getRuleService()`	Static method to retrieve an instance of the rule service.	Constructs a new DSSRuleService object, if necessary. During constructor call any existing rules are loaded from the file system, using the DSSXMLConvertor to convert from XML to DOM to AST.
`store(rule, code)`	Stores a rule (either as a new rule, or replacing an existing rule) with the given source code.	Invokes the Parser to convert source code to AST; Invokes the DSSXMLConvertor to convert AST to DOM and save; Saves the original source to file system for subsequent retrieval; Stores the AST in memory for subsequent running.
`load(rule)`	Load the source code for an existing rule.	Reads stored source code from the file system.
`listRules()`	List all existing rules.	Returns a list of all stored rule names.
`runRules(patientId, target)`	Get all alerts for the given target (summary or dashboard) as appropriate to the given patient.	For each rule: Construct interpreter; Install intrinsics, including alert function which stores to a map; Pre-define patientId for DSS1 program; Run the interpreter on the rule. Thereafter, pull all alerts appropriate to the target from the map.

Table 5: Methods exposed by the DSS Rule Service

modify source for an existing rule). The XML form is used when the DSS Rule Service is first initialized to load any existing rules from the file system. After initialization, rules are stored in memory as AST objects.

The utility class DSSXMLConvertor is used for conversion between AST and XML. Internally, the class maintains a Document Object Model (DOM) representation of the AST. This can be either as loaded from an XML file, or as formed by traversing an AST. Likewise, DSSXMLConvertor provides methods for both producing AST objects or writing XML files.

5.3 Interpreter

The Interpreter is implemented with four distinct sub systems, as depicted in Figure 28. At the top level, *flow control* is provided by the InterpreterVisitor, which is responsible for traversing the Abstract Syntax Tree. An *execution context* is maintained to describe the running state of the system, including defined variables and functions. While tree traversal coordinates complex expressions, the actual *evaluation* of expressions is itself implemented in a distinct set of classes representing the types available under DSS1. Finally, a library of *intrinsic functions* is provided in order to mediate interactions with OpenMRS from running DSS1, as well as to provide certain convenience functions to DSS1 rule programmers.

5.3.1 Flow control

Flow control in the interpreter is implemented using the Visitor design pattern, traversing the Abstract Syntax Tree (AST) produced by the existing Compiler using an implementation of the provided ASTVisitor interface, performing computation as appropriate at every given node in the tree.

The Visitor design pattern leverages double dispatch to decouple a data structure from the operations which can be performed while traversing this data structure. The Visitor calls an `accept` method on a node within the data structure, which is itself overloaded to call a more specific method on the Visitor itself; `visitBlockTree`, for example. This permits the external object the Visitor to implement behavior using the data structure's type hierarchy, without adding that specific behavior to those types directly.

In the case of the Interpreter, the data structure is the AST, which describes a DSS1 program as a tree of elements block (BlockTree), if statements (IfTree), et cetera. The Visitor is the IntepreterVisitor, which manages and performs the computation described by this program. This is done with the support of other underlying subsystems to describe variable state and perform type-specific evaluations, as described in the Architecture section.

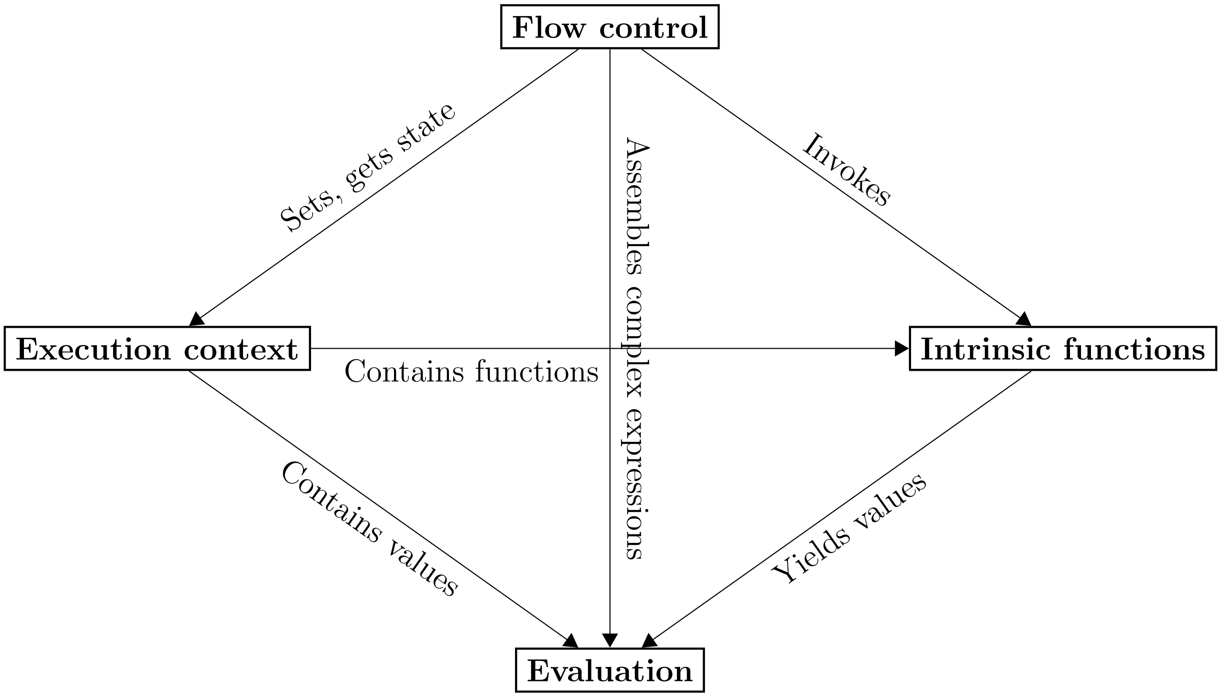


Figure 28: High-level overview of the DSS1 Interpreter.

The class InterpreterVisitor acts as the center of a subsystem responsible for high-level interpretation of the program, including flow control, and coordinating complex expressions.

5.3.2 Execution context

The ExecutionContext class provides a means to store and retrieve return values, variable states, and named functions. It also handles rules of scope to hide variables during function calls, and exposes the Evaluator. Note that this may be populated with functions or even variables before being given to the InterpreterVisitor, allowing the definition of intrinsics and constants (such as `patientId`).

Functions stored in the ExecutionContext are of type DSSFunction, which is an interface used to describe any function called from DSS1 (either intrinsic or user-defined). This permits function calling to be implemented identically for both categories of function. Additionally includes a method for testing if a given argument should be passed as a raw identifier instead of evaluated directly (as used by some intrinsics.).

Similarly, variables and return values are stored as DSSValue objects, with their specific implementation defined within the evaluation subsystem.

5.3.3 Evaluation of expressions

The abstract class DSSValue describes a set of operations which can be performed on values in DSS1 as methods, as well as the common state (the potential to store time stamps). Its concrete sub-classes, such as DSSValueInt, DSSValueFloat, et cetera, provide specific implementations of these operations in order to define the behavior of their DSS1 type. Additionally, concrete subclasses of DSSValue typically are defined with some field to maintain their specific value (for instance, DSSValueBool has an underlying Java `boolean` field to describe its value.)

The Evaluator interface and DSSEvaluator implementation exposes methods to perform operations upon DSS1 values, to interpret literals, allocate DSS objects, and perform conversions between DSS1 values and similar Java objects. The Evaluator serves as intermediary between flow control and the specific semantics implemented in DSSValue types; this facilitates separation of concerns, allowing the gradual introduction of new DSS1 data types while avoiding changes to flow control.

Finally, a DSSValueFactory class is provided to aid in the instantiation of DSSValue objects. This class utilizes the Factory design pattern to allow new values to be created (for instance, as the return values of intrinsic functions) without requiring users of those values to have specific knowledge of the DSSValue subclasses actually used.

5.3.4 Intrinsic functions

DSSLibrary defines an interface for delivering or generating intrinsic functions in related groupings. Each function is returned in a map where the name should be used to call the function from a DSS program, and the function object is used as a Java object that extends the DSSFunction. These functions may then be easily installed into the ExecutionContext used by the Intepreter before running rules. A list of libraries used in this implementation is presented in Table 6.

This approach supports extensibility of the DSS rule module. Rather than being built into the DSS1 Interpreter at the language level, intrinsic functions can be contained and communicated as DSSLibrary objects. Adding intrinsics is then as simple as defining a new DSSLibrary and installing it to the execution context before running rules.

The ReadLibrary serves as an interesting example case, at it illustrates interaction with the OpenMRS platform without requiring specific knowledge of this platform from other elements of the interpreter. The read functions retrieve a list of observations associated with a patient. The first parameter of the functions, patientId, is a numeric identifier unique to each patient. The second parameter, conceptName, is the word or phrase used by the OpenMRS dictionary to refer to a concept.

The three functions are nearly identical, save for one difference: while read() returns a list containing all observations that match the function parameters, readInitialEncounter() and readLatestEncounter() filter out results based on the timestamp of the observations. Calling readInitialEncounter() retrieves only the observations from the patient's earliest encounter on record, while readLatestEncounter() retrieves only the observations from the patient's most recent encounter on record.

When the functions are called, they retrieve a list of all encounters associated with patientId from the OpenMRS database. The functions iterate through these lists, and in the case of readInitialEncounter() and readLatestEncounter(), the timestamp for each encounter is checked. If the timestamp does not meet the criteria, the encounter is discarded. Once an encounter has been verified as valid the function shall retrieve all observations associated with the encounter. Each observation shall have its concept name checked against conceptName, and matches are added to the list of observations that each function shall return.

Observations consist of three pieces of data: the value of the observation, the data type of the observation value, and the time of the observation. Internally, observations are represented as DSSValue objects, which store the value of the observation and the time of the observation. The data type of the observation value is stored as part of the DSSValue class type itself. Both the time and data type of the observation can be retrieved using the time() and type check intrinsics, respectively.

Note that the `alert` intrinsic is treated as a special case. Rather than being contained within a library class, it is installed directly by the DSS Rule Service into the Interpreter before running rules. This facilitates retrieval of results issued via `alert` calls.

Library	**Functions implemented**

IsLibrary	`isString(var)`
	`isFloat(var)`
	`isInt(var)`
	`isBoolean(var)`
	`isList(var)`
	`isObject(var)`
	`isDate(var)`

LengthAndWithinLibrary	`length(var)`
	`within(v,a,b)`

ListLibrary	`merge(a,b)`
	`sortTime(list)`
	`sortData(list)`
	`first(list)`
	`last(list)`

ReadLibrary	`read(patientId, concept)`
	`readInitialEncounter(patientId, concept)`
	`readLatestEncounter(patientId, concept)`

DateLibrary	`currenttime()`
	`recentTimeItem(list)`
	`oldestTimeItem(list)`
	`before(a,b)`
	`time(var)`
	`addDays(v,days)`
	`addMonths(v,months)`

Table 6: Libraries of intrinsics

`org.openmrs.module.dssmodule`	Contains classes which define core functionality exposed to front-end classes or to OpenMRS directly, such as the DSSRuleService, as well as classes which directly support these.
`org.openmrs.module.dssmodule.extension.html`	Describes module-introduced behavior at specific extension points within OpenMRS.
`org.openmrs.module.dssmodule.web.controller`	Provides classes to handle web requests which support maintenance and execution of DSS rules.

Table 7: Description of packages which support interaction with OpenMRS front-end

6 Package structure

Figure 29 illustrates the usage relationships between packages. For brevity, only the last relevant tokens of a package name are used; for instance, `org.openmrs.module.dssmodule` has been labeled simply `dssmodule`.

6.1 OpenMRS integration packages

Multiple components support interaction with the DSS module from OpenMRS. These are described in Table 7.

6.2 Interpreter packages

As described in Section 5.3, the Interpreter is composed of four major sub-systems. These are defined in corresponding packages, as documented in Table 8.

6.3 Provided packages

At the start of this project, an existing compiler for the DSS1 language was provided, and has been used without significant modification. The packages which comprise this component are identified in Table 9.

`org.openmrs.module.dssmodule.flowcontrol`	Classes which handle and implement flow control for the DSS1 language.
`org.openmrs.module.dssmodule.intrinsics`	Implementations of DSS1 intrinsic functions, and the libraries used to organize them.
`org.openmrs.module.dssmodule.state`	Describes the execution classes and closely related classes.
`org.openmrs.module.dssmodule.value`	Defines the specific value types available under DSS1 and implements their behavior.

Table 8: Description of interpreter packages

`org.openmrs.module.dssmodule.lexer`	Provides classes which support the conversion of raw DSS1 source code to meaningful lexical units (tokens).
`org.openmrs.module.dssmodule.ast`	Provides classes to describe the syntactic structure of a compiled DSS1 program.
`org.openmrs.module.dssmodule.parser`	Supports parsing of DSS1 source code (using the `lexer` package) into a compiled tree structure (using the `ast` package).
`org.openmrs.module.dssmodule.visitor`	Describes an interface for traversing a compiled AST, following the visitor design pattern.

Table 9: Description of provided packages

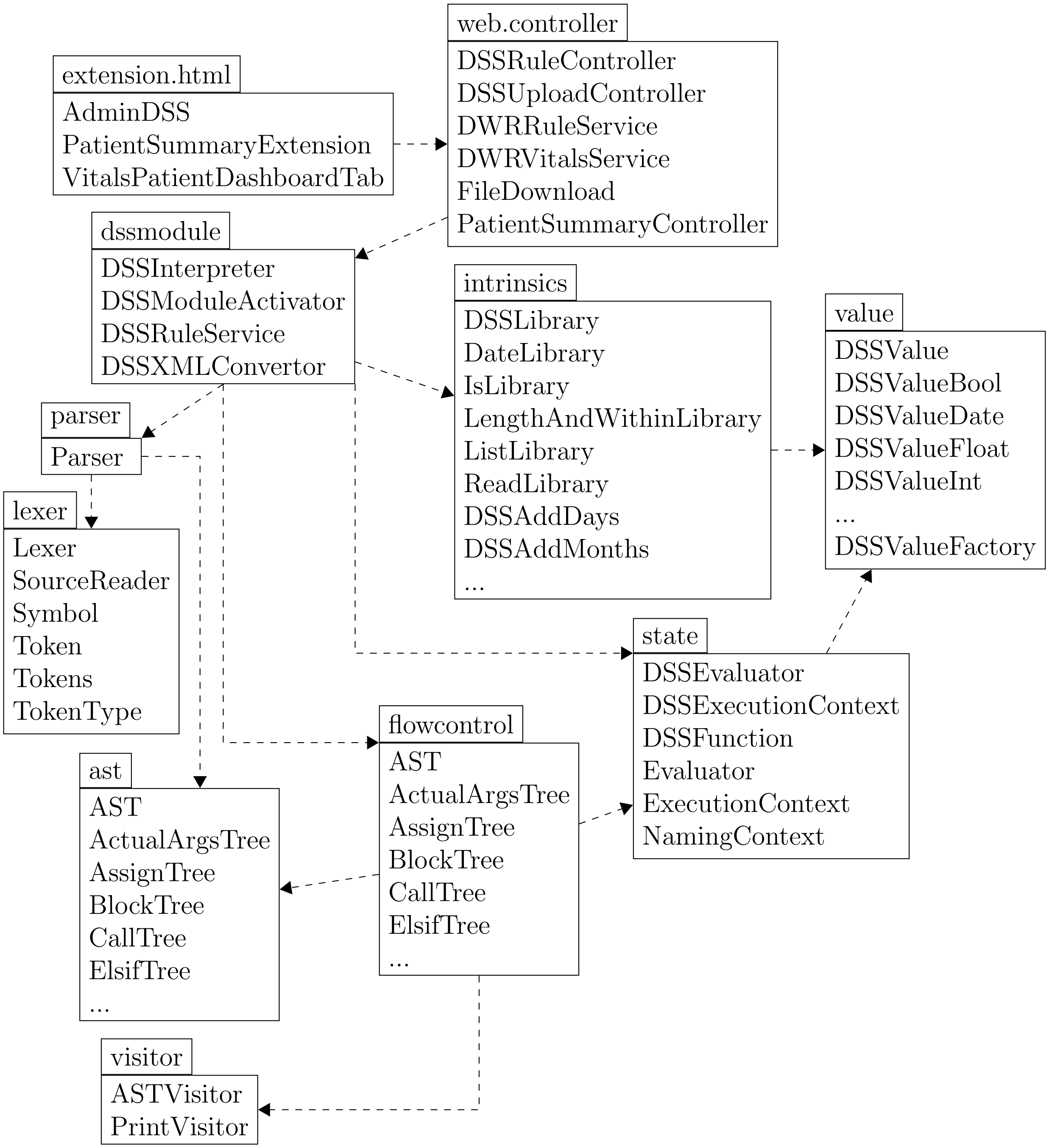


Figure 29: Package diagram

7 Class diagrams

7.1 Classes describing flow control

Figure 30 shows the relationship of classes directly involved in or utilized during the handling of flow control when interpreting a DSS program.

The InterpreterVisitor is used to initiate program interpretation. It delegates the interpretation of specific node types to corresponding ASTInterpreter types. It also maintains an instance of a DSSExecutionContext to support interactions with the running state of the program.

7.2 Classes describing execution context

Figure 31 describes the composition of the execution context. The active state of the program, including currently-defined functions and variable assignments, is maintained in appropriate data structures.

Each ExecutionContext additionally maintains a reference to an Evaluator object, which exposes necessary methods for the interpreter to interact with values in DSS.

7.3 Classes describing values

Figure 32 shows the classes used to describe values in a running DSS program. The abstract class DSSValue describes the operations available under DSS1 as methods; its concrete subclasses provide implementations for these operations. Note that each DSSValue object also maintains a field for the time stamp of a value, which is populated when observations are read from OpenMRS services. For other values this is null.

Not shown is DSSValueFactory, which exposes methods to create DSSValue objects to wrap their corresponding underlying Java types.

7.4 Classes describing intrinsics

Figure 33 describes the relationship of classes which describe specific intrinsic functions to the classes used to categorize them.

Calls made by a running DSS program are resolved by a named DSSFunction object, as stored in the execution context. The DSSLibrary interface provides a useful way to group related functions along with their names to facilitate their installation into the execution context.

Also shown is the DeclaredFunction class. A DeclaredFunction is created for user-defined functions in a DSS program, and contains references to appropriate nodes within the AST to support actual interpretation of the function call, as well as the visitor used to perform interpretation, and the execution context, which is used to handle the change in variable scope associated with the function call.

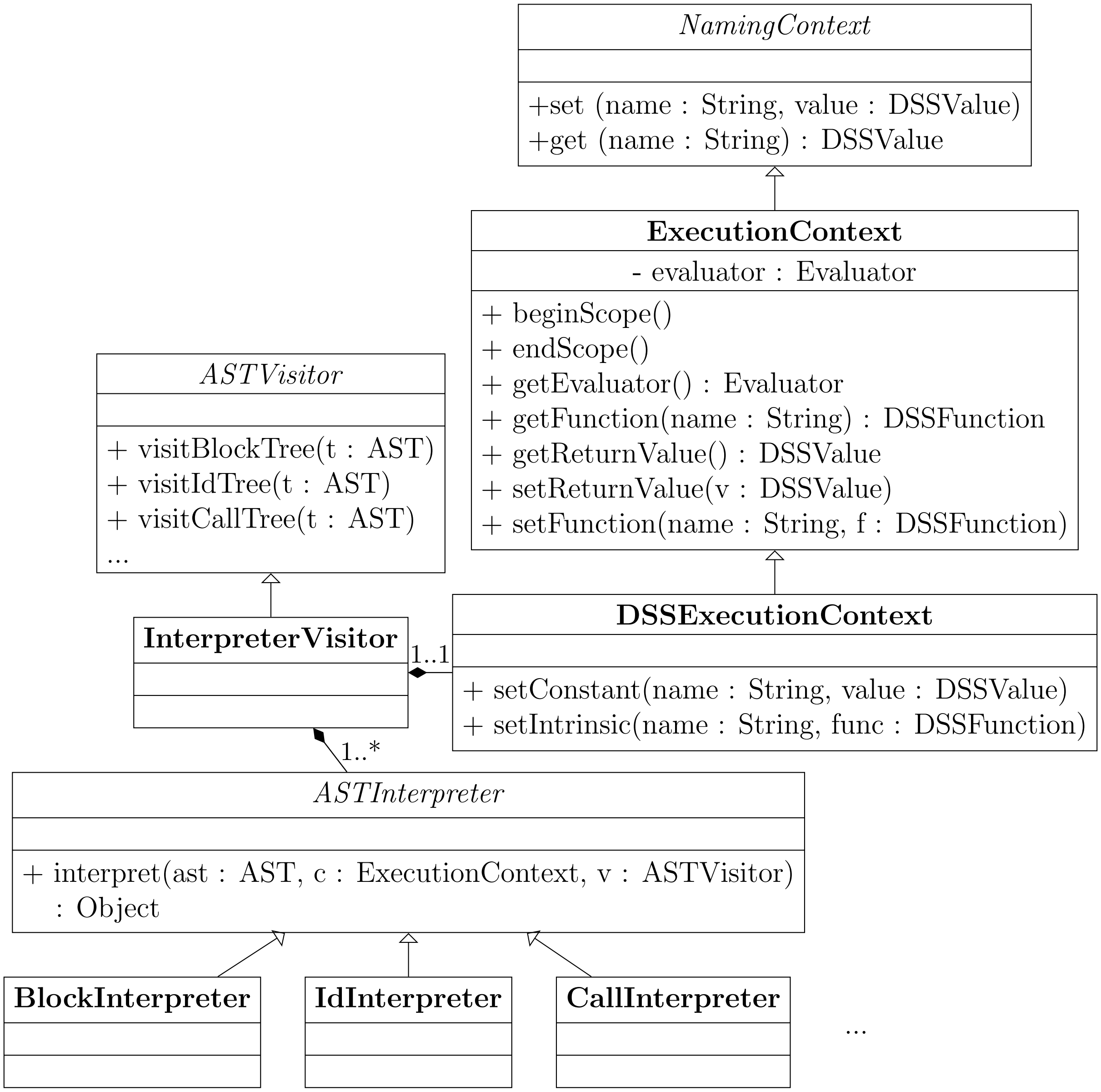

NamingContext
+set (name : String, value : DSSValue)
+get (name : String) : DSSValue
ExecutionContext
- evaluator : Evaluator
+ beginScope()
+ endScope()
+ getEvaluator() : Evaluator
+ getFunction(name : String) : DSSFunction
+ getReturnValue() : DSSValue
+ setReturnValue(v : DSSValue)
+ setFunction(name : String, f : DSSFunction)
ASTVisitor
+ visitBlockTree(t : AST)
+ visitIdTree(t : AST)
+ visitCallTree(t : AST)
...
InterpreterVisitor
1..1
DSSExecutionContext
+ setConstant(name : String, value : DSSValue)
+ setIntrinsic(name : String, func : DSSFunction)
1..*
ASTInterpreter
+ interpret(ast : AST, c : ExecutionContext, v : ASTVisitor) : Object
BlockInterpreter
IdInterpreter
CallInterpreter
...

Figure 30: Interpreter visitor

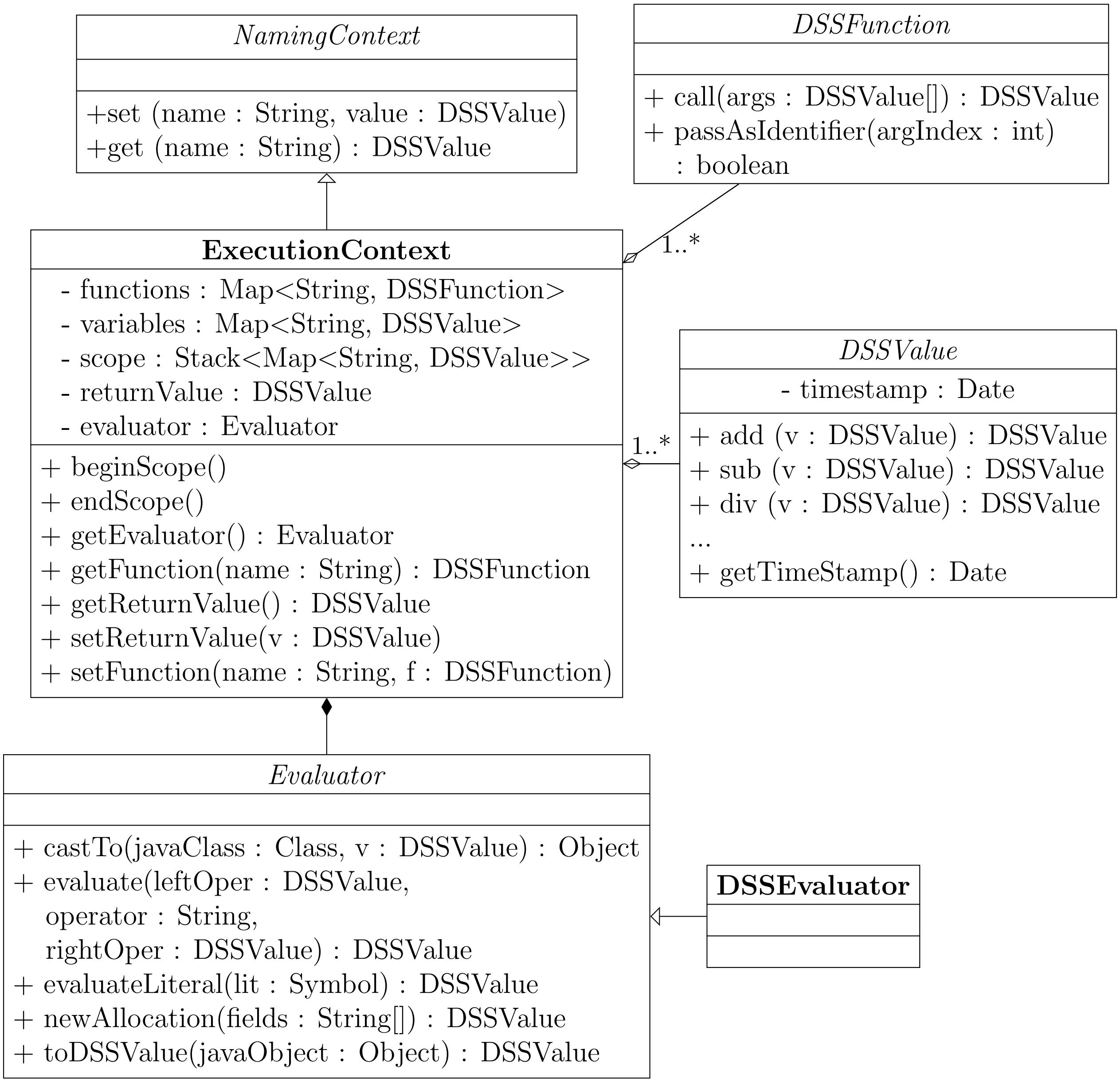

NamingContext
+set (name : String, value : DSSValue)
+get (name : String) : DSSValue
DSSFunction
+ call(args : DSSValue[]) : DSSValue
+ passAsIdentifier(argIndex : int)
: boolean
1..*
ExecutionContext
- functions : Map<String, DSSFunction>
- variables : Map<String, DSSValue>
- scope : Stack<Map<String, DSSValue>>
- returnValue : DSSValue
- evaluator : Evaluator
+ beginScope()
+ endScope()
+ getEvaluator() : Evaluator
+ getFunction(name : String) : DSSFunction
+ getReturnValue() : DSSValue
+ setReturnValue(v : DSSValue)
+ setFunction(name : String, f : DSSFunction)
DSSValue
- timestamp : Date
1..*
+ add (v : DSSValue) : DSSValue
+ sub (v : DSSValue) : DSSValue
+ div (v : DSSValue) : DSSValue
...
+ getTimeStamp() : Date
Evaluator
+ castTo(javaClass : Class, v : DSSValue) : Object
+ evaluate(leftOper : DSSValue,
operator : String,
rightOper : DSSValue) : DSSValue
+ evaluateLiteral(lit : Symbol) : DSSValue
+ newAllocation(fields : String[]) : DSSValue
+ toDSSValue(javaObject : Object) : DSSValue
DSSEvaluator

Figure 31: Execution context class diagram

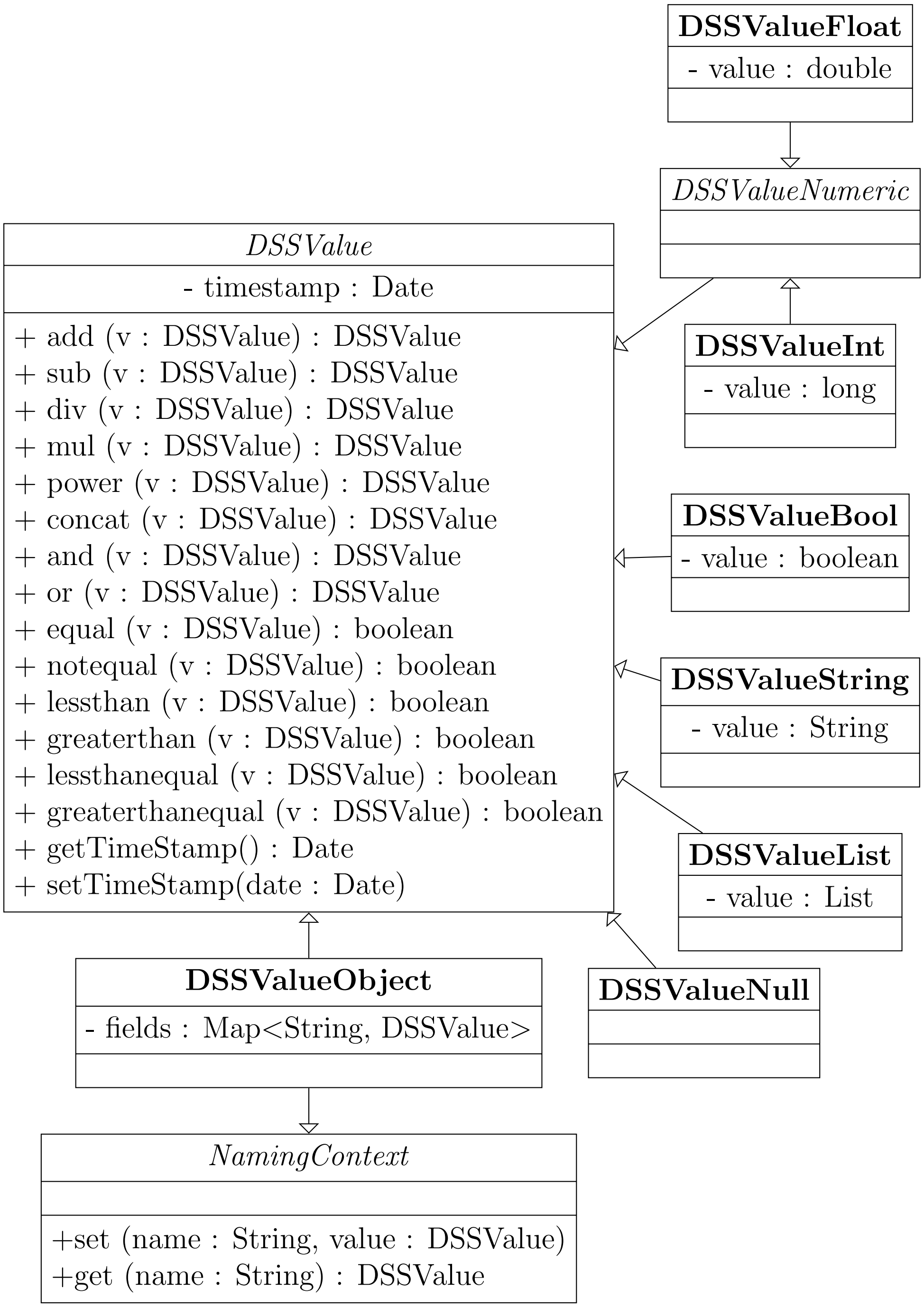

DSSValueFloat
- value : double
DSSValueNumeric
DSSValue
- timestamp : Date
+ add (v : DSSValue) : DSSValue
+ sub (v : DSSValue) : DSSValue
+ div (v : DSSValue) : DSSValue
+ mul (v : DSSValue) : DSSValue
+ power (v : DSSValue) : DSSValue
+ concat (v : DSSValue) : DSSValue
+ and (v : DSSValue) : DSSValue
+ or (v : DSSValue) : DSSValue
+ equal (v : DSSValue) : boolean
+ notequal (v : DSSValue) : boolean
+ lessthan (v : DSSValue) : boolean
+ greaterthan (v : DSSValue) : boolean
+ lessthanequal (v : DSSValue) : boolean
+ greaterthanequal (v : DSSValue) : boolean
+ getTimeStamp() : Date
+ setTimeStamp(date : Date)
DSSValueInt
- value : long
DSSValueBool
- value : boolean
DSSValueString
- value : String
DSSValueList
- value : List
DSSValueObject
- fields : Map<String, DSSValue>
DSSValueNull
NamingContext
+set (name : String, value : DSSValue)
+get (name : String) : DSSValue

Figure 32: Value class diagram

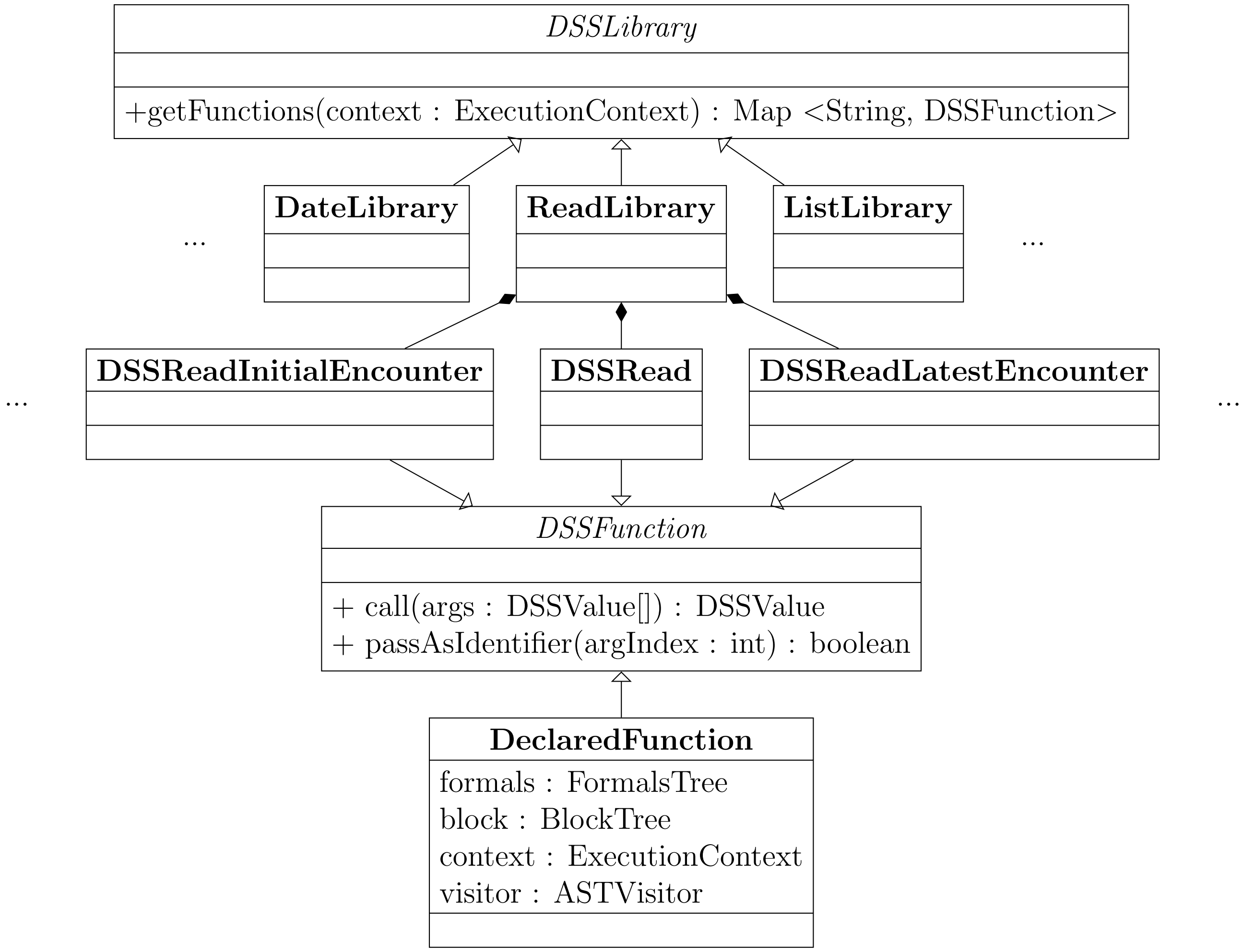


Figure 33: Intrinsic class diagram

A DSS1 language specification

A.1 Grammar

```
 PROGRAM -> program D* BLOCK ==> program
 BLOCK -> { S* }  ==> block
 D -> 'function' NAME FUNHEAD BLOCK       ==> functionDecl
 FUNHEAD  -> '(' (NAME list ',')? ')'   ==> formals
 S -> if EE then BLOCK ('else' BLOCK)? ==> if
   -> if EE then BLOCK Elif ==> if
   -> while EE BLOCK                ==> while
   -> 'for' NAME in NLIST BLOCK      ==> FOR
   -> return EE                     ==> return
   -> BLOCK
   -> IdMod:= EE                     ==> assign
   -> NAME '(' (EE list ',')? ')'  ==> call

  Elif -> elsif EE 'then' BLOCK Elif ==> elsif
-> elsif EE 'then' BLOCK ('else' BLOCK)? ==> elsif

  EE  -> E
   -> EE '||' E

  E  -> SE
   -> SE == SE    ==> =
   -> SE != SE    ==> !=
   -> SE <  SE    ==> <
   -> SE <= SE    ==> <=

  SE  ->  T
   ->  SE + T   ==> +
   ->  SE - T   ==> -
   ->  SE | T   ==> or

  T   -> TT
   -> T * F   ==> *
   -> T / F   ==> /
   -> T & F   ==> and

  TT  -> F
   -> TT ** F     ==> **

  F   -> ( EE )
   -> IdMod
   -> <literal>
   -> NAME '(' (EE list ',')? ')'  ==> call
   -> Object '(' (NAME list ',')? ')'    ==> ObjectDecl
   -> new NAME           ==> Object
   -> LIST
  IdMod -> NAME
        -> NAME '.' NAME     ==> fieldRef

  NLIST -> NAME
        -> LIST

  LIST -> '{' (E list ',')? '}'    ==> list

  NAME -> <id>
```

A.2 Intrinsic functions

Several built-in functions may be utilized when writing a DSS program, as listed in Table 10. Function names are case sensitive.

`isString(v)`	Check if `v` is of type string; returns a boolean.
`isInt(v)`	Check if `v` is of type integer; returns a boolean.
`isFloat(v)`	Check if `v` is of type float; returns a boolean.
`isBoolean(v)`	Check if `v` is of type boolean; returns a boolean.
`isDate(v)`	Check if `v` is of type date; returns a boolean.
`isObject(v)`	Check if `v` is of type object; returns a boolean.
`isList(v)`	Check if `v` is of type list; returns a boolean.
`currenttime()`	Returns current system time, as a date.
`recentTimeItem(list)`	Returns the item in the list with the most recent time.
`oldestTimeItem(list)`	Returns the item with the oldest time.
`before(t1, t2)`	Returns true if `t1` is before `t2`
`time(v)`	Returns the time associated with `v`
`addDays(time, n)`	Returns a new date, `n` days after `time`.
`addMonths(time, n)`	Returns a new date, `n` months after `time`.
`merge(list1, list2)`	Returns a new list containing all items from `list1` and `list2`, sorted chronologically.
`sortTime(list)`	Returns a new list sorted based on time of items
`sortData(list)`	Returns a new list sorted based on values of items
`last(list)`	Returns the last item in the list
`first(list)`	Returns the first in the list
`read(p, c)`	Returns a list of observations of concept `c` for the specified patient `p` from database. `c` should match the name given in the Concept Dictionary.
`readInitialEncounter(p,c)`	Returns a list of observations for patient `p` and concept `c` from database, only from the initial encounter.
`readLatestEncounter(p,c)`	Returns a list of observations for patient `p` and concept `c` from database, only from the latest encounter.
`alert(target, message)`	Send an alert to a target; `target` can either be `summary` (for Patient Summary) or `dashboard` (for Patient Dashboard)
`within(v, a, b)`	Returns true if `v` is between `a` and `b`, inclusive.
`length(v)`	Returns the length of a list or a string.

Table 10: DSS1 intrinsic functions

B Test cases

B.1 Language testing

B.1.1 Source code for language test

```
program                 //program keyword

function show(value) {
  // When run from console, ”alert” intrinsic uses no target
  alert(value)
}

function space(){
  show(” ”)
}

function expect(testName, expected, actual) {
  if expected == actual
    then { result := ”PASSED” }
    else { result := ”FAILED” }
  show(””               + result +
       ” test for ’”    + testName +
       ”’. Expected: ” + expected +
       ”, actual: ”     + actual)
}

// Comments work

function returnTest(){
   return ”return value”
}

function argumentTest(a,b){
   return b + a
}
```

```
{
  show("====STARTING TESTS====")
  space()

  // Assignment tests
  strings := "Hello"    //Assignments work
  ints := 4
  bools := true
  expect("string assignment", "Hello", strings)
  expect("numeric assignment", 4, ints)
  expect("boolean assignment", true, bools)
  space()

  // If tests
  value := ""
  if ints == 4 then { value := "thenBlock" }
  expect("if-then", "thenBlock", value)

  if bools == false then { value := "thenBlock" }
                    else { value := "elseBlock" }
  expect("if-then-else", "elseBlock", value)

  if bools == false then{ value := "thenBlock"  }
  elsif ints == 4   then{ value := "elsifBlock" }
                    else{ value := "elseBlock"  }
  expect("if-then-elsif-else", "elsifBlock", value)
  space()

  // Arithmetic test
  expect("addition", 10, 5 + 5)
  expect("subtraction", 11, 12 - 1)
  expect("multiplication", 32, 8 * 4)
  expect("division", 0.5, 1.0 / 2.0)
  expect("exponentiation", 4, 2 ** 2)
  expect("square root", 2, 4 ** 0.5)
  expect("multi-part expression", 32, 6 * 5 + 6 / 3)
  space()
```

```
// Comparison test
expect("less than",          true , 5  < 10)
expect("less than",          false, 15 < 10)
expect("less than",          false, 5  < 5)
expect("less than or equal", true,  5 <= 10)
expect("less than or equal", false, 15<= 10)
expect("less than or equal", true , 5 <= 5)
space()

// Equality test
expect("integer equality",   true, 1 == 1)
expect("string equality" ,   true, "a string"=="a string")
expect("float equality",     true, 2.0 == 2.0)
expect("list equality",      true, {1,2,3} == {1,2,3})
expect("integer equality",   false, 1 == 2)
expect("string equality",    false, "a string"=="string a")
expect("float equality",     false, 2.0 == 2.5)
expect("list equality",      false, {1,2,3} == {1,2,8})
expect("integer inequality", false, 1 != 1)
expect("string inequality",  false, "a string"!="a string")
expect("float inequality",   false, 2.0 != 2.0)
expect("list inequality",    false, {1,2,3} != {1,2,3})
expect("integer inequality", true,  1 != 2)
expect("string equality" ,   true, "a string"!="string a")
expect("float inequality",   true,  2.0 != 2.5)
expect("list inequality",    true,  {1,2,3} != {1,2,8})
space()

// Logic test
expect("t-or-f" , true,  true  | false)
expect("t-or-t" , true,  true  | true)
expect("f-or-t" , true,  false | true)
expect("f-or-f" , false, false | false)
expect("t-and-f", false, true  & false)
expect("t-and-t", true,  true  & true)
expect("f-and-t", false, false & true)
```

```
  expect("f-and-f", false, false & false)
  space()

  // Function call test
  expect("call/return", "return value", returnTest() )
  expect("argument passing", "a bee", argumentTest("bee","a ") )
  space()

  // Loops
  power := 1     i := 0
  while (i < 8) { power := power * 2    i := i + 1 }
  expect("while loop", 256, power)
  sum := 0       list := { 1 , 2 , 3 , 4 }
  for x in list { sum := sum + x }
  expect("for loop", 10, sum)
  space()

  // Objects
  obj := Object(x, y, z)
  obj.x := 0
  obj.y := "a field"
  obj.z := 3.0
  expect("field reference", 0, obj.x)
  expect("field reference", "a field", obj.y)
  other := new obj
  other.x := 1
  expect("new object", 1, other.x)
  space()

  show("=====ENDING TESTS=====")
}
```

B.1.2 Results for language test

```
=====STARTING TESTS=====

PASSED test for 'string assignment'. Expected: Hello, actual: Hello
PASSED test for 'numeric assignment'. Expected: 4, actual: 4
PASSED test for 'boolean assignment'. Expected: True, actual: True

PASSED test for 'if-then'. Expected: thenBlock, actual: thenBlock
PASSED test for 'if-then-else'. Expected: elseBlock, actual: elseBlock
PASSED test for 'if-then-elsif-else'. Expected: elsifBlock, actual: elsifBlock

PASSED test for 'addition'. Expected: 10, actual: 10
PASSED test for 'subtraction'. Expected: 11, actual: 11
PASSED test for 'multiplication'. Expected: 32, actual: 32
PASSED test for 'division'. Expected: 0.5, actual: 0.5
PASSED test for 'exponentiation'. Expected: 4, actual: 4
PASSED test for 'square root'. Expected: 2, actual: 2.0
PASSED test for 'multi-part expression'. Expected: 32, actual: 32

PASSED test for 'less than'. Expected: True, actual: True
PASSED test for 'less than'. Expected: False, actual: False
PASSED test for 'less than'. Expected: False, actual: False
PASSED test for 'less than or equal'. Expected: True, actual: True
PASSED test for 'less than or equal'. Expected: False, actual: False
PASSED test for 'less than or equal'. Expected: True, actual: True

PASSED test for 'integer equality'. Expected: True, actual: True
PASSED test for 'string equality'. Expected: True, actual: True
PASSED test for 'float equality'. Expected: True, actual: True
PASSED test for 'list equality'. Expected: True, actual: True
PASSED test for 'integer equality'. Expected: False, actual: False
PASSED test for 'string equality'. Expected: False, actual: False
PASSED test for 'float equality'. Expected: False, actual: False
PASSED test for 'list equality'. Expected: False, actual: False
PASSED test for 'integer inequality'. Expected: False, actual: False
PASSED test for 'string inequality'. Expected: False, actual: False
PASSED test for 'float inequality'. Expected: False, actual: False
PASSED test for 'list inequality'. Expected: False, actual: False
PASSED test for 'integer inequality'. Expected: True, actual: True
PASSED test for 'string equality'. Expected: True, actual: True
PASSED test for 'float inequality'. Expected: True, actual: True
PASSED test for 'list inequality'. Expected: True, actual: True

PASSED test for 't-or-f'. Expected: True, actual: True
PASSED test for 't-or-t'. Expected: True, actual: True
PASSED test for 'f-or-t'. Expected: True, actual: True
PASSED test for 'f-or-f'. Expected: False, actual: False
PASSED test for 't-and-f'. Expected: False, actual: False
PASSED test for 't-and-t'. Expected: True, actual: True
PASSED test for 'f-and-t'. Expected: False, actual: False
PASSED test for 'f-and-f'. Expected: False, actual: False

PASSED test for 'call/return'. Expected: return value, actual: return value
PASSED test for 'argument passing'. Expected: a bee, actual: a bee

PASSED test for 'while loop'. Expected: 256, actual: 256
PASSED test for 'for loop'. Expected: 10, actual: 10

PASSED test for 'field reference'. Expected: 0, actual: 0
PASSED test for 'field reference'. Expected: a field, actual: a field
PASSED test for 'new object'. Expected: 1, actual: 1

=====ENDING TESTS=====
```

B.2 Intrinsic function testing

B.2.1 Source code for comprehensive intrinsics test

```
program
// BASIC TESTING OF INTRINSICS
{
        b       := true
        lst     := {1,2,3,4,5}
        time    := currenttime()
        time2 := addDays(time, 7)

        // Utilizing DSSRead functions
        wt      := read(patientId, "WEIGHT (KG)")

        // isLibrary
        alert(summary, isInt(1234))
        alert(summary, isString("hello"))
        alert(summary, isFloat(12.345))
        alert(summary, isBoolean(b))
        alert(summary, isList(lst))
        alert(summary, isDate(time))

        // DSSListLibrary
        alert(summary, merge(wt, wt))
        alert(summary, sortTime(wt))
        alert(summary, sortData({1,10,4,15}))
        alert(summary, last(wt))
        alert(summary, first(wt))

        // DSSDateLibrary
        alert(summary, addDays(time, 14))
        alert(summary, addMonths(time, 2))
        alert(summary, before(time, time2))
        alert(summary, recentTimeItem(wt))
        alert(summary, oldestTimeItem(wt))
        alert(summary, time(recentTimeItem(wt)))


        // Misc
        alert(summary, within(40,20,60))
        alert(summary, length({10,20,30,40,50}))

}
```

B.2.2 Results for comprehensive intrinsics test

```
True
True
True
True
True
True
{60.0, 60.0, 30.0, 30.0, 65.0, 65.0}
{60.0, 30.0, 65.0}
{1, 4, 10, 15}
60.0
65.0
Fri May 10 19:26:55 PDT 2013
Wed Jun 26 19:26:55 PDT 2013
True
65.0
60.0
2013-03-25 00:00:00.0
True
5
```

B.2.3 Source code for example test (blood pressure)

```
program

function bloodpressure()
{
        systolic := last(read(patientId, "SYSTOLIC BLOOD PRESSURE"))
        diastolic := last(read(patientId, "DIASTOLIC BLOOD PRESSURE"))
        alert(summary, "Systolic: " + systolic)
        alert(summary, "Diastolic: " + diastolic)
        check(systolic, diastolic)
}

function check(sys, dias)
{
        if ((sys < 120) | (dias < 80)) then
        {
            alert(summary,
               "Normal Blood Pressure")
      }
        elsif (within(sys, 120, 139) |
              within(dias,     80,89)) then
        {
            alert(summary,
               "Prehypertension")
      }
        elsif (within(sys, 140,159) |
              within(dias, 90, 99)) then
        {
            alert(summary,
               "Stage 1 Hypertension.")
      }
        else
        {
            alert(summary,
               "Stage 2 Hypertension.")
      }
}

{
          bloodpressure()
}
```

B.2.4 Results for example test (blood pressure)

```
Systolic: 100.0
Diastolic: 70.0
Normal Blood Pressure
```

B.2.5 Source code for example test (temperature)

```
program
{
          count := 0
          date  := currenttime()
          month := currenttime()

          while(count < 12)
          {
                    if (count == 10) then
                    {
                         alert(summary,
                       addMonths(date, count))
                    }
                    count := count + 10
          }

          list := read(patientId, "TEMPERATURE (C)")
          wt := readLatestEncounter(patientId, "WEIGHT (KG)")

          if (before(time(oldestTimeItem(list)),
                time(first(wt))))then
          {
                    alert(summary,
                  "Number of observation of temperature
                  on record = " + length(list))
```

```
        }


        if (isList(wt)) then
        { alert(summary, merge(list , wt)) }


}
```

B.2.6 Results for example test (temperature)

```
Wed Feb 26 19:26:55 PST 2014
Number of observation of temperature on record = 3
{37.0, 40.0, 30.0, 65.0}
```

C API Documentation

C.1 Class Hierarchy

Classes

- java.lang.Object
    - AdministrationSectionExt
        - org.openmrs.module.dssmodule.extension.html.AdminDSS (in C.4.1, page 83)
    - Extension
        - org.openmrs.module.dssmodule.extension.html.PatientSummaryExtension (in C.4.2, page 84)
    - PatientDashboardTabExt
        - org.openmrs.module.dssmodule.extension.html.RulesPatientDashboardTab (in C.4.3, page 85)
        - org.openmrs.module.dssmodule.extension.html.VitalsPatientDashboardTab (in C.4.4, page 86)
    - java.lang.Enum
        - org.openmrs.module.dssmodule.lexer.Tokens (in C.7.5, page 127)
    - org.openmrs.module.dssmodule.DSSInterpreter (in C.3.1, page 74)
    - org.openmrs.module.dssmodule.DSSModuleActivator (in C.3.2, page 77)
    - org.openmrs.module.dssmodule.DSSRuleService (in C.3.3, page 78)
    - org.openmrs.module.dssmodule.DSSXMLConvertor (in C.3.5, page 80)
    - org.openmrs.module.dssmodule.ast.AST (in C.10.3, page 160)
        - org.openmrs.module.dssmodule.ast.ActualArgsTree (in C.10.1, page 158)
        - org.openmrs.module.dssmodule.ast.AssignTree (in C.10.2, page 159)
        - org.openmrs.module.dssmodule.ast.BlockTree (in C.10.4, page 163)
        - org.openmrs.module.dssmodule.ast.CallTree (in C.10.5, page 163)
        - org.openmrs.module.dssmodule.ast.ElsifTree (in C.10.6, page 164)
        - org.openmrs.module.dssmodule.ast.FieldRefTree (in C.10.7, page 165)
        - org.openmrs.module.dssmodule.ast.ForTree (in C.10.9, page 167)
        - org.openmrs.module.dssmodule.ast.FormalsTree (in C.10.8, page 166)
        - org.openmrs.module.dssmodule.ast.FunctionDeclTree (in C.10.10, page 168)
        - org.openmrs.module.dssmodule.ast.IdTree (in C.10.11, page 169)
        - org.openmrs.module.dssmodule.ast.IfTree (in C.10.12, page 171)
        - org.openmrs.module.dssmodule.ast.ListTree (in C.10.13, page 171)
        - org.openmrs.module.dssmodule.ast.LiteralTree (in C.10.14, page 172)
        - org.openmrs.module.dssmodule.ast.ObjectDeclTree (in C.10.15, page 173)

  - org.openmrs.module.dssmodule.ast.ObjectTree (in C.10.16, page 174)
  - org.openmrs.module.dssmodule.ast.OpTree (in C.10.17, page 175)
  - org.openmrs.module.dssmodule.ast.ProgramTree (in C.10.18, page 176)
  - org.openmrs.module.dssmodule.ast.ReturnTree (in C.10.19, page 177)
  - org.openmrs.module.dssmodule.ast.WhileTree (in C.10.20, page 178)
- org.openmrs.module.dssmodule.flowcontrol.AssignInterpreter (in C.8.2, page 134)
- org.openmrs.module.dssmodule.flowcontrol.BlockInterpreter (in C.8.3, page 135)
- org.openmrs.module.dssmodule.flowcontrol.CallInterpreter (in C.8.4, page 136)
- org.openmrs.module.dssmodule.flowcontrol.FieldRefInterpreter (in C.8.5, page 136)
- org.openmrs.module.dssmodule.flowcontrol.ForInterpreter (in C.8.6, page 137)
- org.openmrs.module.dssmodule.flowcontrol.FormalsInterpreter (in C.8.7, page 138)
- org.openmrs.module.dssmodule.flowcontrol.FunctionDeclInterpreter (in C.8.8, page 139)
- org.openmrs.module.dssmodule.flowcontrol.IdInterpreter (in C.8.9, page 139)
- org.openmrs.module.dssmodule.flowcontrol.IfInterpreter (in C.8.10, page 140)
- org.openmrs.module.dssmodule.flowcontrol.ListInterpreter (in C.8.12, page 144)
- org.openmrs.module.dssmodule.flowcontrol.LiteralInterpreter (in C.8.13, page 145)
- org.openmrs.module.dssmodule.flowcontrol.ObjectDeclInterpreter (in C.8.14, page 145)
- org.openmrs.module.dssmodule.flowcontrol.ObjectInterpreter (in C.8.15, page 146)
- org.openmrs.module.dssmodule.flowcontrol.OpInterpreter (in C.8.16, page 147)
- org.openmrs.module.dssmodule.flowcontrol.ProgramInterpreter (in C.8.17, page 147)
- org.openmrs.module.dssmodule.flowcontrol.ReturnInterpreter (in C.8.18, page 148)
- org.openmrs.module.dssmodule.flowcontrol.WhileInterpreter (in C.8.19, page 149)
- org.openmrs.module.dssmodule.intrinsics.AnnotatedDSSLibrary (in C.6.4, page 101)
  - org.openmrs.module.dssmodule.DSSRuleService.DSSAlertMap (in C.3.4, page 80)
  - org.openmrs.module.dssmodule.intrinsics.DemoLibrary (in C.6.6, page 103)
  - org.openmrs.module.dssmodule.intrinsics.ListLibrary (in C.6.20, page 118)
- org.openmrs.module.dssmodule.intrinsics.DateLibrary (in C.6.5, page 102)
- org.openmrs.module.dssmodule.intrinsics.IsLibrary (in C.6.18, page 116)
- org.openmrs.module.dssmodule.intrinsics.LengthAndWithinLibrary (in C.6.19, page 117)
- org.openmrs.module.dssmodule.intrinsics.ReadLibrary (in C.6.21, page 120)
- org.openmrs.module.dssmodule.lexer.Lexer (in C.7.1, page 121)
- org.openmrs.module.dssmodule.lexer.SourceReader (in C.7.2, page 124)
- org.openmrs.module.dssmodule.lexer.Symbol (in C.7.3, page 125)
- org.openmrs.module.dssmodule.lexer.Token (in C.7.4, page 126)
- org.openmrs.module.dssmodule.lexer.TokenType (in C.7.6, page 131)
- org.openmrs.module.dssmodule.parser.Parser (in C.5.1, page 87)
- org.openmrs.module.dssmodule.state.DSSEvaluator (in C.2.3, page 65)

- org.openmrs.module.dssmodule.state.DSSFunction (in C.2.5, page 69)
    - org.openmrs.module.dssmodule.intrinsics.DSSAddDays (in C.6.7, page 104)
    - org.openmrs.module.dssmodule.intrinsics.DSSAddMonths (in C.6.8, page 106)
    - org.openmrs.module.dssmodule.intrinsics.DSSAlert (in C.6.9, page 107)
    - org.openmrs.module.dssmodule.intrinsics.DSSBefore (in C.6.10, page 108)
    - org.openmrs.module.dssmodule.intrinsics.DSSCurrentTime (in C.6.11, page 109)
    - org.openmrs.module.dssmodule.intrinsics.DSSOldestTimeItem (in C.6.12, page 110)
    - org.openmrs.module.dssmodule.intrinsics.DSSRead (in C.6.13, page 111)
        - org.openmrs.module.dssmodule.intrinsics.DSSReadInitialEncounter (in C.6.14, page 112)
        - org.openmrs.module.dssmodule.intrinsics.DSSReadLatestEncounter (in C.6.15, page 113)
    - org.openmrs.module.dssmodule.intrinsics.DSSRecentTimeItem (in C.6.16, page 114)
    - org.openmrs.module.dssmodule.intrinsics.DSSTime (in C.6.17, page 115)
- org.openmrs.module.dssmodule.state.ExecutionContext (in C.2.6, page 70)
    - org.openmrs.module.dssmodule.state.DSSExecutionContext (in C.2.4, page 67)
- org.openmrs.module.dssmodule.value.DSSValue (in C.11.1, page 180)
    - org.openmrs.module.dssmodule.value.DSSValueBool (in C.11.2, page 184)
    - org.openmrs.module.dssmodule.value.DSSValueDate (in C.11.3, page 186)
    - org.openmrs.module.dssmodule.value.DSSValueList (in C.11.7, page 194)
    - org.openmrs.module.dssmodule.value.DSSValueNull (in C.11.8, page 196)
    - org.openmrs.module.dssmodule.value.DSSValueNumeric
        - org.openmrs.module.dssmodule.value.DSSValueFloat (in C.11.5, page 189)
        - org.openmrs.module.dssmodule.value.DSSValueInt (in C.11.6, page 191)
    - org.openmrs.module.dssmodule.value.DSSValueObject (in C.11.9, page 197)
    - org.openmrs.module.dssmodule.value.DSSValueString (in C.11.10, page 200)
- org.openmrs.module.dssmodule.value.DSSValueFactory (in C.11.4, page 188)
- org.openmrs.module.dssmodule.value.OpenmrsDSSValue (in C.11.11, page 202)
- org.openmrs.module.dssmodule.visitor.ASTVisitor (in C.9.1, page 150)
    - org.openmrs.module.dssmodule.flowcontrol.InterpreterVisitor (in C.8.11, page 141)
    - org.openmrs.module.dssmodule.visitor.PrintVisitor (in C.9.2, page 153)

Interfaces

- java.lang.annotation.Annotation
- org.openmrs.module.dssmodule.intrinsics.DSSIdentifier (in C.6.1, page 99)
- org.openmrs.module.dssmodule.intrinsics.DSSIntrinsic (in C.6.2, page 99)

- org.openmrs.module.dssmodule.flowcontrol.ASTInterpreter (in C.8.1, page 133)
- org.openmrs.module.dssmodule.intrinsics.DSSLibrary (in C.6.3, page 100)
- org.openmrs.module.dssmodule.state.Evaluator (in C.2.1, page 61)
- org.openmrs.module.dssmodule.state.NamingContext (in C.2.2, page 63)

C.2 Package org.openmrs.module.dssmodule.state

C.2.1 Interface Evaluator

Responsible for evaluating simple expressions; the fine-grained semantics of the language.

Declaration public interface Evaluator

All known subinterfaces DSSEvaluator (in C.2.3, page 65)

All classes known to implement interface DSSEvaluator (in C.2.3, page 65)

Method summary

castTo(Class, DSSValue) Try to cast this value to another type (likely a normal Java type).
evaluate(DSSValue, String, DSSValue)
evaluateLiteral(Symbol) Evaluate this literal and convert it to an appropriate DSSValue.
newAllocation(String[]) Allocate a new object (probably DSSObject) containing the specified fields.
toDSSValue(Object) Convert a regular Java object to an analogous DSSValue, if possible.

Methods

- **castTo**
  `java.lang.Object` **castTo**(`java.lang.Class` **type**, `org.openmrs.module.dssmodule.value.DS`
  value)
  - **Description**
    Try to cast this value to another type (likely a normal Java type). Support may vary upon implementation, but should include Boolean and String. Returns null if the cast is invalid.
  - **Parameters**
    - `type` –
    - `value` –
  - **Returns** –
- **evaluate**
  `org.openmrs.module.dssmodule.value.DSSValue` **evaluate**(`org.openmrs.module.dssmodule.value.DSSValue` **leftOperand**, `java.lang.String` **operator**, `org.openmrs.module.dssmodule.value.DSSValue` **rightOperand**)
- **evaluateLiteral**
  `org.openmrs.module.dssmodule.value.DSSValue` **evaluateLiteral**(`org.openmrs.module.dssmodule.lexer.Symbol` **literal**)

  - **Description**

    Evaluate this literal and convert it to an appropriate DSSValue. Note that this is also where raw identifiers are handled (such as in a read(patientId, cd4counts)) so this method needs to recognize those as well.
  - **Parameters**
    - `literal` –
  - **Returns** –

- **newAllocation**
  `org.openmrs.module.dssmodule.value.DSSValue` **newAllocation**`(java.lang.String[]` **fields**`)`
  - **Description**

    Allocate a new object (probably DSSObject) containing the specified fields.
  - **Parameters**
    - `fields` –
  - **Returns** –

- **toDSSValue**
  `org.openmrs.module.dssmodule.value.DSSValue` **toDSSValue**`(java.lang.Object` **javaObject**`)`
  - **Description**

    Convert a regular Java object to an analogous DSSValue, if possible.
  - **Parameters**
    - `javaObject` –
  - **Returns** –

C.2.2 Interface NamingContext

Represents a place where values may be stored and retrieved by name. This may be the ExecutionContext in general, or it may be a DSS object.

Declaration public interface NamingContext

All known subinterfaces DSSExecutionContext (in C.2.4, page 67), ExecutionContext (in C.2.6, page 70), DSSValueObject (in C.11.9, page 197)

All classes known to implement interface ExecutionContext (in C.2.6, page 70), DSSValueObject (in C.11.9, page 197)

Method summary

get(String) Get the value currently associated with the specified name in this context

names() All names used by objects in this context

set(String, DSSValue) Set a name-value association in this context.

Methods

- **get**
  `org.openmrs.module.dssmodule.value.DSSValue` **`get(java.lang.String name)`**

  - **Description**
    Get the value currently associated with the specified name in this context
  - **Parameters**
    - `name` –
  - **Returns** –

- **names**
  `java.lang.String[]` **`names()`**

  - **Description**
    All names used by objects in this context
  - **Returns** –

- **set**
  `void` **`set(java.lang.String name, org.openmrs.module.dssmodule.value.DSSValue value)`**

  - **Description**
    Set a name-value association in this context. This may overwrite any previous association.

- **Parameters**
  - `name` –
  - `value` –

C.2.3 Class DSSEvaluator

Handles the evaluation of basic operations in DSS1, and provides related functionality to interchange values from their DSS1 forms and their plain Java equivalents.

Declaration public class DSSEvaluator
extends java.lang.Object
implements Evaluator

Constructor summary

DSSEvaluator()

Method summary

castTo(Class, DSSValue) Convert a DSSValue to another Java type.
evaluate(DSSValue, String, DSSValue) Evaluate a simple expression
evaluateLiteral(Symbol) Evaluate a literal as a DSS value
newAllocation(String[]) Allocate a new DSS object, with specified fields.
toDSSValue(Object) Convert a plain Java object to an analogous DSS value

Constructors

- **DSSEvaluator**
  `public` **DSSEvaluator**`()`

Methods

- **castTo**
  `public java.lang.Object` **castTo**`(java.lang.Class` **type**`,`
  `org.openmrs.module.dssmodule.value.DSSValue` **value**`)`
  - **Description**
    Convert a DSSValue to another Java type. May return null (Java null) if the conversion does not make sense.
  - **Parameters**

    - `type` –
    - `value` –
  - **Returns** –

- **evaluate**
  `public org.openmrs.module.dssmodule.value.DSSValue` **evaluate**`(org.openmrs.module.dssmodule.value.DSSValue` **leftOperand**`, java.lang.String` **operator**`, org.openmrs.module.dssmodule.value.DSSValue` **rightOperand**`)`
  - **Description**
    Evaluate a simple expression
  - **Parameters**
    - `leftOperand` –
    - `operator` –
    - `rightOperand` –
  - **Returns** – the result of the operation

- **evaluateLiteral**
  `public org.openmrs.module.dssmodule.value.DSSValue` **evaluateLiteral**`(org.openmrs.module.dssmodule.lexer.Symbol` **symbol**`)`
  - **Description**
    Evaluate a literal as a DSS value
  - **Parameters**
    - `symbol` –
  - **Returns** –

- **newAllocation**
  `public org.openmrs.module.dssmodule.value.DSSValue` **newAllocation**`(java.lang.String[]` **fields**`)`
  - **Description**
    Allocate a new DSS object, with specified fields. All fields will be initialized to the DSS version of null.

- **Parameters**
  - `fields` – the fields to allocate
- **Returns** –

- **toDSSValue**
  `public org.openmrs.module.dssmodule.value.DSSValue` **toDSSValue**`(java.lang.Object` **javaObject**`)`
  - **Description**
    Convert a plain Java object to an analogous DSS value
  - **Parameters**
    - `javaObject` –
  - **Returns** –

C.2.4 Class DSSExecutionContext

Extends the base ExecutionContext to allow variables and functions to be defined which cannot be changed by a running DSS program.

Declaration public class DSSExecutionContext
extends org.openmrs.module.dssmodule.state.ExecutionContext (in C.2.6, page 70)

Constructor summary

DSSExecutionContext(Evaluator)

Method summary

get(String)
getFunction(String)
setConstant(String, DSSValue) Associate a constant value with a given name.
setIntrinsic(String, DSSFunction) Associate a constant function with a given name.

Constructors

- **DSSExecutionContext**
  `public` **DSSExecutionContext**`(Evaluator` **evaluator**`)`

Methods

- **get**
  `public org.openmrs.module.dssmodule.value.DSSValue` **get**`(java.lang.String` **name**`)`
  - **Description copied from ExecutionContext** (**in C.2.6, page 70**)
    Get the value of a variable associated with the specified name
  - **Parameters**
    - `name` –
  - **Returns** –
- **getFunction**
  `public DSSFunction` **getFunction**`(java.lang.String` **name**`)`
  - **Description copied from ExecutionContext** (**in C.2.6, page 70**)
    Retrieve a named function.
  - **Parameters**
    - `name` – the name of the function
  - **Returns** – an object which handles calls to the function
- **setConstant**
  `public void` **setConstant**`(java.lang.String` **name**`,`
  `org.openmrs.module.dssmodule.value.DSSValue` **value**`)`
  - **Description**
    Associate a constant value with a given name. This is used to set pre-defined values in OpenMRS, such as patientId
  - **Parameters**
    - `name` –
    - `value` –

- **setIntrinsic**
  `public void` **setIntrinsic**`(java.lang.String` **name**`, DSSFunction` **func**`)`
  - **Description**
    Associate a constant function with a given name. This is used to set intrinsic functions.
  - **Parameters**
    - `name` –
    - `func` –

Members inherited from class ExecutionContext
`org.openmrs.module.dssmodule.state.ExecutionContext` (in C.2.6, page 70)
beginScope, endScope, get, getEvaluator, getFunction, getReturnValue, names, set, setFunction, setReturnValue

C.2.5 Class DSSFunction

Represents a callable function in DSS. This may be either an intrinsic, or one defined in the source code.

Declaration public abstract class DSSFunction
extends java.lang.Object

All known subclasses DSSReadLatestEncounter (in C.6.15, page 113), DSSCurrentTime (in C.6.11, page 109), DSSRead (in C.6.13, page 111), DSSAddMonths (in C.6.8, page 106), DSSBefore (in C.6.10, page 108), DSSRecentTimeItem (in C.6.16, page 114), DSSAddDays (in C.6.7, page 104), DSSReadInitialEncounter (in C.6.14, page 112), DSSTime (in C.6.17, page 115), DSSOldestTimeItem (in C.6.12, page 110), DSSAlert (in C.6.9, page 107)

Constructor summary

DSSFunction()

Method summary

call(DSSValue[]) Call this function.
passAsIdentifier(int) Some DSS intrinsics want raw identifiers passed, straight from the code.

Constructors

- **DSSFunction**
  `public` **DSSFunction**`()`

Methods

- **call**
  `public abstract org.openmrs.module.dssmodule.value.DSSValue`
  call`(org.openmrs.module.dssmodule.value.DSSValue[]` **args**`)`
  - **Description**
    Call this function. The arguments provided are as observed by the interpreter. Note that the actual number of arguments may not match the number of parameters expected; it is ultimately the function's responsibility to handle this situation.
  - **Parameters**
    - `args` – the arguments to the function
  - **Returns** – the return value of the represented function

- **passAsIdentifier**
  `public boolean` **passAsIdentifier**`(int` **argumentIndex**`)`
  - **Description**
    Some DSS intrinsics want raw identifiers passed, straight from the code. They may indicate this through this method. The majority of DSS functions will never need this, so by default this simply returns false.
  - **Parameters**
    - `argumentIndex` –
  - **Returns** –

C.2.6 Class ExecutionContext

Maintains things like variables & known functions.

Declaration public class ExecutionContext
extends java.lang.Object
implements NamingContext

All known subclasses DSSExecutionContext (in C.2.4, page 67)

Constructor summary

ExecutionContext(Evaluator)

Method summary

beginScope() Begin a new variable scope.
endScope() End the current variable scope, and restore the previous one.
get(String) Get the value of a variable associated with the specified name
getEvaluator() Get the evaluator appropriate to this execution context
getFunction(String) Retrieve a named function.
getReturnValue() Get the currently-specified return value.
names() Get a list of all variable names defined in this execution context
set(String, DSSValue) Associate a value with a variable name.
setFunction(String, DSSFunction) Store a named function to this execution context.
setReturnValue(DSSValue) Set the current return value.

Constructors

- **ExecutionContext**
  `public` **ExecutionContext**`(Evaluator` **evaluator**`)`

Methods

- **beginScope**
  `public void` **beginScope**`()`
  - **Description**
    Begin a new variable scope.
- **endScope**
  `public void` **endScope**`()`
  - **Description**
    End the current variable scope, and restore the previous one.
- **get**
  `public org.openmrs.module.dssmodule.value.DSSValue` **get**`(java.lang.String` **name**`)`

- **Description**
  Get the value of a variable associated with the specified name
- **Parameters**
  - `name` –
- **Returns** –

- **getEvaluator**
  `public Evaluator` **getEvaluator**`()`
  - **Description**
    Get the evaluator appropriate to this execution context
  - **Returns** –

- **getFunction**
  `public DSSFunction` **getFunction**`(java.lang.String` **name**`)`
  - **Description**
    Retrieve a named function.
  - **Parameters**
    - `name` – the name of the function
  - **Returns** – an object which handles calls to the function

- **getReturnValue**
  `public org.openmrs.module.dssmodule.value.DSSValue` **getReturnValue**`()`
  - **Description**
    Get the currently-specified return value. This will be Java null whenever no return value has been specified. This is important, as BlockInterpreter polls for changes to return value and stops executing if their are any (in order to ensure that a function call ends immediately upon return.)
  - **Returns** –
  - **See also**
    - BlockInterpreter

- **names**
  `public java.lang.String[]` **names**`()`

- **Description**

  Get a list of all variable names defined in this execution context

- **Returns** –

- **set**

  `public void` **set**`(java.lang.String` **name**`, org.openmrs.module.dssmodule.value.DSSValue` **value**`)`

  - **Description**

    Associate a value with a variable name. Any previously-defined value for that name in the current execution scope will be overwritten.

  - **Parameters**

    * `name` –
    * `value` –

- **setFunction**

  `public void` **setFunction**`(java.lang.String` **name**`, DSSFunction` **f**`)`

  - **Description**

    Store a named function to this execution context.

  - **Parameters**

    * `name` –
    * `f` –

- **setReturnValue**

  `public void` **setReturnValue**`(org.openmrs.module.dssmodule.value.DSSValue` **v**`)`

  - **Description**

    Set the current return value. Setting to null means that any current return value has been processed.

  - **Parameters**

    * `v` – the new return value (or null to clear return value)

C.3 Package org.openmrs.module.dssmodule

Package Contents *Page*

Classes

C.3.1 Class DSSInterpreter

An interpreter is responsible for running parsed DSS1 programs.

Declaration public class DSSInterpreter
extends java.lang.Object

Constructor summary

DSSInterpreter(DSSLibrary[])
DSSInterpreter(Map, DSSLibrary[]) Create a new Interpreter with the supplied elements pre-defined.

Method summary

defineConstant(String, Object) Associate a constant value with the specified name.

install(DSSLibrary) Install the provided library of functions into this interpreter's execution context.

install(Map) Install the provided library of functions into this interpreter's execution context.

install(String, DSSFunction) Install a function into this interpreter's execution context.

interpret(AST) Interpret the DSS1 program described by the provided AST.

main(String[])

Constructors

- **DSSInterpreter**
  `public` **DSSInterpreter**`(intrinsics.DSSLibrary[]` **libraries**`)`
- **DSSInterpreter**
  `public` **DSSInterpreter**`(java.util.Map` **constants**`, intrinsics.DSSLibrary[]` **libraries**`)`
  - **Description**
    Create a new Interpreter with the supplied elements pre-defined. Values in the constants maps will be converted from their original Java types to DSSValues if necessary, and supplied to the running DSS program as pre-defined variables Libraries will be installed as intrinsics.
  - **Parameters**
    * `constants` – a map of variable names to constant values
    * `libraries` – a list of libraries of functions to install

Methods

- **defineConstant**
  `public void` **defineConstant**`(java.lang.String` **name**`, java.lang.Object` **value**`)`
  - **Description**
    Associate a constant value with the specified name. This is used to define constants such as patient id Supplied values may be common Java objects (including Long, Double, String) and will be converted where possible to appropriate types for usage in the DSS subsystem.
  - **Parameters**
    * `name` –

      - `value` –

- **install**
  `public void` **install**`(intrinsics.DSSLibrary` **library**`)`
  - **Description**

    Install the provided library of functions into this interpreter's execution context. This allows users of the interpreter to pre-define or override certain functions, as appropriate to usage.
  - **Parameters**
    - `library` –

- **install**
  `public void` **install**`(java.util.Map` **library**`)`
  - **Description**

    Install the provided library of functions into this interpreter's execution context. This allows users of the interpreter to pre-define or override certain functions, as appropriate to usage. Functions are delivered in a map, where keys give the name of the function, and the value gives the DSSFunction object itself.
  - **Parameters**
    - `library` – map of names to functions

- **install**
  `public void` **install**`(java.lang.String` **name**`, state.DSSFunction` **func**`)`
  - **Description**

    Install a function into this interpreter's execution context. This can be used to introduce intrinsics and to override existing functions.
  - **Parameters**
    - `name` –
    - `func` –

- **interpret**
  `public void` **interpret**`(ast.AST` **ast**`)`

- **Description**
  Interpret the DSS1 program described by the provided AST.
- **Parameters**
  - `ast` –

- **main**
  `public static void` **main**`(java.lang.String[]` **args**`)`

C.3.2 Class DSSModuleActivator

This class contains the logic that is run every time this module is either started or shutdown

Declaration public class DSSModuleActivator
extends java.lang.Object

Constructor summary

DSSModuleActivator()

Method summary

shutdown()
startup()

Constructors

- **DSSModuleActivator**
  `public` **DSSModuleActivator**`()`

Methods

- **shutdown**
  `public void` **shutdown**`()`
  - **See also**
    - org.openmrs.module.Activator#shutdown()

- **startup**
  `public void` **startup**`()`
  - **See also**
    - org.openmrs.module.Activator#startup()

C.3.3 Class DSSRuleService

Provides a facade for loading, saving, and running DSS rules.

Declaration public class DSSRuleService
extends java.lang.Object

Method summary

getRuleService() Get an instance of the DSSRuleService
listRules() Get all currently defined rules.
load(String) Load existing source code for a rule.
main(String[])
runRules(int) Run all known rules for the given patient
runRules(int, String)
store(String, String) Store a rule with the given name and source code into the rule system.

Methods

- **getRuleService**
  `public static DSSRuleService` **getRuleService**`()`
  - **Description**
    Get an instance of the DSSRuleService
  - **Returns** –
- **listRules**
  `public java.util.Collection` **listRules**`()`
  - **Description**
    Get all currently defined rules.
  - **Returns** –
- **load**
  `public java.lang.String` **load**`(java.lang.String` **ruleName**`) throws java.io.IOException`
  - **Description**
    Load existing source code for a rule.

    - **Parameters**
        - `ruleName` –
    - **Returns** –
    - **Throws**
        - `java.io.IOException` –

- **main**
  `public static void` **main**`(java.lang.String[]` **args**`)`

- **runRules**
  `public java.util.Map` **runRules**`(int` **patientId**`)`
    - **Description**
      Run all known rules for the given patient
    - **Parameters**
        - `patientId` – the patient's numeric id, for encounter look ups
    - **Returns** – all alerts (mapped from target name ->list of alert messages)

- **runRules**
  `public java.util.List` **runRules**`(int` **patientId**`, java.lang.String` **target**`)`

- **store**
  `public void` **store**`(java.lang.String` **ruleName**`, java.lang.String` **sourceCode**`) throws java.lang.Exception`
    - **Description**
      Store a rule with the given name and source code into the rule system. The rule may subsequently be retrieved by this name. If there are problems with the choice of name or compilation errors in the source code, an exception will be thrown.
    - **Parameters**
        - `ruleName` – the name to store the rule under
        - `sourceCode` – DSS1 source code to run for this rule
    - **Throws**
        - `java.lang.Exception` – if storage could not be completed

C.3.4 Class DSSRuleService.DSSAlertMap

Provides an implementation of the ”alert” intrinsic which stores alerts to an internal map, allowing these to subsequently be delivered to appropriate targets.

Declaration public class DSSRuleService.DSSAlertMap
extends org.openmrs.module.dssmodule.intrinsics.AnnotatedDSSLibrary (in C.6.4, page 101)

Constructor summary

DSSRuleService.DSSAlertMap()

Method summary

alert(String, String)
results()

Constructors

- **DSSRuleService.DSSAlertMap**
  `public` **DSSRuleService.DSSAlertMap**`()`

Methods

- **alert**
  `public void` **alert**`(java.lang.String` **target**`, java.lang.String` **alertText**`)`

- **results**
  `public java.util.Map` **results**`()`

Members inherited from class AnnotatedDSSLibrary
`org.openmrs.module.dssmodule.intrinsics.AnnotatedDSSLibrary` (in C.6.4, page 101)
getFunctions

C.3.5 Class DSSXMLConvertor

Supports bi-directional conversion between XML and AST representations of DSS1 programs, using DOM as an intermediary.

Declaration public class DSSXMLConvertor
extends java.lang.Object

Field summary

KIND_ATTR
VALUE_ATTR

Constructor summary

DSSXMLConvertor() Create a convertor with no program.
DSSXMLConvertor(AST) Crate a new Convertor, initialized from an existing AST.
DSSXMLConvertor(File) Crate a new Convertor, initialized from an existing XML file.

Method summary

getAST() Retrieve the current program in AST form
write(File) Store this program as an XML file
write(OutputStream) Write this program as an XML file to the specified stream

Fields

- public static final java.lang.String **KIND_ATTR**
- public static final java.lang.String **VALUE_ATTR**

Constructors

- **DSSXMLConvertor**
  `public` **DSSXMLConvertor**`() throws javax.xml.parsers.ParserConfigurationException`
  - **Description**
    Create a convertor with no program.
  - **Throws**
    - `javax.xml.parsers.ParserConfigurationException` –

- **DSSXMLConvertor**
  `public` **DSSXMLConvertor**`(ast.AST` **tree**`) throws javax.xml.parsers.ParserConfigurationException`
  - **Description**
    Crate a new Convertor, initialized from an existing AST.
  - **Parameters**
    - `tree` – the compiled AST which represents the DSS1 program
  - **Throws**
    - `javax.xml.parsers.ParserConfigurationException` –
- **DSSXMLConvertor**
  `public` **DSSXMLConvertor**`(java.io.File` **file**`) throws java.lang.Exception`
  - **Description**
    Crate a new Convertor, initialized from an existing XML file.
  - **Parameters**
    - `file` – an XML file containing a compiled DSS1 program
  - **Throws**
    - `java.lang.Exception` –

Methods

- **getAST**
  `public java.util.List` **getAST**`() throws java.lang.Exception`
  - **Description**
    Retrieve the current program in AST form
  - **Returns** – an AST representing the current program
  - **Throws**
    - `java.lang.Exception` –
- **write**
  `public void` **write**`(java.io.File` **outputFile**`) throws java.lang.Exception`
  - **Description**
    Store this program as an XML file

- **Parameters**
  - `outputFile` – the file to which to store this program
- **Throws**
  - `java.lang.Exception` –

- **write**
  `public void` **write**`(java.io.OutputStream` **output**`) throws java.lang.Exception`
  - **Description**
    Write this program as an XML file to the specified stream
  - **Parameters**
    - `output` –
  - **Throws**
    - `java.lang.Exception` –

C.4 Package org.openmrs.module.dssmodule.extension.html

Package Contents *Page*

Classes

C.4.1 Class AdminDSS

Declaration public class AdminDSS
extends AdministrationSectionExt

Constructor summary

AdminDSS()

Method summary

getLinks()
getMediaType()
getTitle()

Constructors

- **AdminDSS**
  `public` **`AdminDSS`**`()`

Methods

- **getLinks**
  `public java.util.Map` **`getLinks`**`()`
  - **See also**
    - org.openmrs.module.web.extension.AdministrationSectionExt#getLinks()
- **getMediaType**
  `public Extension.MEDIA_TYPE` **`getMediaType`**`()`
  - **See also**
    - org.openmrs.module.web.extension.AdministrationSectionExt#getMediaType()
- **getTitle**
  `public java.lang.String` **`getTitle`**`()`
  - **See also**
    - org.openmrs.module.web.extension.AdministrationSectionExt#getTitle()

C.4.2 Class PatientSummaryExtension

Declaration public class PatientSummaryExtension
extends Extension

Constructor summary

PatientSummaryExtension()

Method summary

getMediaType()
getOverrideContent(String)
initialize(Map)

Constructors

- **PatientSummaryExtension**
  `public` **PatientSummaryExtension**`()`

Methods

- **getMediaType**
  `public Extension.MEDIA_TYPE` **getMediaType**`()`
- **getOverrideContent**
  `public java.lang.String` **getOverrideContent**`(java.lang.String` **bodyContent**`)`
- **initialize**
  `public void` **initialize**`(java.util.Map` **parameters**`)`

C.4.3 Class RulesPatientDashboardTab

Provides a ”Rules” tab (used only for test purposes)

Declaration public class RulesPatientDashboardTab
extends PatientDashboardTabExt

Constructor summary

RulesPatientDashboardTab()

Method summary

getMediaType()
getPortletUrl()
getRequiredPrivilege()
getTabId()
getTabName()

Constructors

- **RulesPatientDashboardTab**
  `public` **RulesPatientDashboardTab**`()`

Methods

- **getMediaType**
  `public Extension.MEDIA_TYPE` **getMediaType**`()`

- **getPortletUrl**
  `public java.lang.String` **getPortletUrl**`()`

- **getRequiredPrivilege**
  `public java.lang.String` **getRequiredPrivilege**`()`

- **getTabId**
  `public java.lang.String` **getTabId**`()`

- **getTabName**
  `public java.lang.String` **getTabName**`()`

C.4.4 Class VitalsPatientDashboardTab

Provides a tab for viewing patient vitals

Declaration public class VitalsPatientDashboardTab
extends PatientDashboardTabExt

Constructor summary

VitalsPatientDashboardTab()

Method summary

getMediaType()
getPortletUrl()
getRequiredPrivilege()
getTabId()
getTabName()

Constructors

- **VitalsPatientDashboardTab**
  `public` **VitalsPatientDashboardTab**`()`

Methods

- **getMediaType**
  `public Extension.MEDIA_TYPE` **getMediaType**`()`

- **getPortletUrl**
  `public java.lang.String` **getPortletUrl**`()`

- **getRequiredPrivilege**
  `public java.lang.String` **getRequiredPrivilege**`()`

- **getTabId**
  `public java.lang.String` **getTabId**`()`

- **getTabName**
  `public java.lang.String` **getTabName**`()`

C.5 Package org.openmrs.module.dssmodule.parser

Package Contents *Page*

Classes

C.5.1 Class Parser

The Parser class performs recursive-descent parsing; as a by-product it will build the **Abstract Syntax Tree** representation for the source program
Following is the Grammar we are using:

```
PROGRAM -> program D* BLOCK ==> program
BLOCK -> { S* }  ==> block
D -> ’function’ NAME FUNHEAD BLOCK      ==> functionDecl
FUNHEAD  -> ’(’ (NAME list ’,’)? ’)’  ==> formals
S -> if EE then BLOCK (’else’ BLOCK)? ==> if
  -> if EE then BLOCK Elif ==> if
-> while EE BLOCK               ==> while
-> ’for’ NAME in NLIST BLOCK     ==> FOR
-> return EE                    ==> return
-> BLOCK
-> IdMod:= EE                    ==> assign
-> NAME ’(’ (EE list ’,’)? ’)’ ==> call

Elif -> elsif EE ’then’ BLOCK Elif                 ==> elsif
     -> elsif EE ’then’ BLOCK (’else’ BLOCK)? ==> elsif

EE -> E
-> EE ’||’ E

E -> SE
-> SE == SE   ==> =
-> SE != SE   ==> !=
-> SE   SE   ==>
-> SE = SE   ==> =

SE  ->  T
->  SE + T  ==> +
->  SE - T  ==> -
->  SE | T  ==> or

T  -> TT
-> T * F  ==> *
-> T / F  ==> /
-> T & F  ==> and

TT -> F
```

```
 -> TT**F    ==> **

 F  -> ( EE )
 -> IdMod
 ->
 -> NAME '(' (EE list ',')? ')' ==> call
 -> Object '(' (NAME list ',')? ')'    ==> ObjectDecl
 -> new NAME          ==> Object
 -> LIST
 IdMod -> NAME
       -> NAME '.' NAME    ==> fieldRef

  NLIST  -> NAME
         -> LIST

 LIST  -> '{' (E list ',')? '}'   ==> list

 NAME  ->
```

Declaration public class Parser
extends java.lang.Object

Constructor summary

Parser(String) Construct a new Parser;

Method summary

execute() Execute the parse command
getLex()
IdMod() IdMod ->NAME ->NAME '.' NAME ==>fieldRef
rBlock() pre>block ->'{' s* '}' ==>block
rDecl() pre>d ->'function' NAME FUNHEAD BLOCK ==>functionDecl
rEExpr()

```
    EE -> E
       -> EE '||' E
```

rElif() Elif ->elsif EE ’then’ BLOCK Elif ==>elsif ->elsif EE ’then’ BLOCK (’else’ BLOCK)? ==>elsif

rExpr()

```
e -> se
-> se ’==’ se ==> =
-> se ’!=’ se ==> !=
-> se ’’ se ==>
-> se ’=’ se ==> =
```

rFactor()

```
f -> ’(’ ee ’)’
-> name
->
-> name ’(’ (ee list ’,’)? ’)’ ==> call
-> Object ’(’ (NAME list ’,’)? ’)’    ==> ObjectDecl
-> new NAME          ==> Object
-> NAME ’.’ NAME     ==> fieldRef
-> LIST
```

rFunHead() pre>funHead ->’(’ (NAME list ’,’)? ’)’ ==>formals note a funhead is a list of zero or more decl’s separated by commas, all in parens

rList() LIST ->’{’ (EE list ’,’)? ’}’ ==>list

rName()

```
name ->
```

rNList() NLIST ->NAME ->LIST

rPowerTerm() tt ->F ->TT**F

rProgram() pre>PROGRAM ->program D* BLOCK ==>program

rSimpleExpr()

```
se -> t
-> se ’+’ t ==> +
-> se ’-’ t ==> -
-> se ’|’ t ==> or
This
rule indicates we should pick up as many  t’s as possible; the
 t’s will be left associative
```

rStatement()

```
S ->  if EE then BLOCK ( rElif* ’else’ BLOCK)? ==> if
-> while EE BLOCK ==> while
-> ’for’ NAME in NLIST BLOCK ==> FOR
-> return EE ==> return
-> BLOCK
-> IdMod := EE ==> assign
-> FieldRef ':=" EE             ==> assign
-> name '(' (ee list ',')? ) ==> call
```

rTerm()

```
t -> tt
-> t '*' tt ==> *
-> t '/' tt ==> /
-> t '&' tt ==> and

This
rule indicates we should pick up as many  tt's as possible; the
 tt's will be left associative
```

Constructors

- **Parser**
  `public` **Parser**`(java.lang.String` **sourceProgram**`) throws java.lang.Exception`
  - **Description**
    Construct a new Parser;
  - **Parameters**
    - `sourceProgram` – - source file name
  - **Throws**
    - `java.lang.Exception` – - thrown for any problems at startup (e.g. I/O)

Methods

- **execute**
  `public org.openmrs.module.dssmodule.ast.AST` **execute**`() throws java.lang.Exception`
  - **Description**
    Execute the parse command
  - **Returns** – the AST for the source program
  - **Throws**
    - `java.lang.Exception` – - pass on any type of exception raised

- **getLex**
  `public org.openmrs.module.dssmodule.lexer.Lexer` **getLex**`()`

- **IdMod**
  `public org.openmrs.module.dssmodule.ast.AST` **IdMod**`() throws org.openmrs.module.dssmodule.parser.SyntaxError`

  - **Description**
    IdMod ->NAME ->NAME '.' NAME ==>fieldRef
  - **Returns** –
  - **Throws**
    - `org.openmrs.module.dssmodule.parser.SyntaxError` –

- **rBlock**
  `public org.openmrs.module.dssmodule.ast.AST` **rBlock**`() throws org.openmrs.module.dssmodule.parser.SyntaxError`

  - **Description**
    pre>block ->'{' s* '}' ==>block
  - **Returns** – block tree
  - **Throws**
    - `org.openmrs.module.dssmodule.parser.SyntaxError` – - thrown for any syntax error e.g. an expected left brace isn't found

- **rDecl**
  `public org.openmrs.module.dssmodule.ast.AST` **rDecl**`() throws org.openmrs.module.dssmodule.parser.SyntaxError`

  - **Description**
    pre>d ->'function' NAME FUNHEAD BLOCK ==>functionDecl
  - **Returns** – either the decl tree or the functionDecl tree
  - **Throws**
    - `org.openmrs.module.dssmodule.parser.SyntaxError` – - thrown for any syntax error

- **rEExpr**
  `public org.openmrs.module.dssmodule.ast.AST` **`rEExpr`**`() throws org.openmrs.module.dssmodule.parser.SyntaxError`
  - **Description**

    ```
    EE -> E
       -> EE '||' E
    ```

  - **Returns** – the tree corresponding to the expression
  - **Throws**
    - `org.openmrs.module.dssmodule.parser.SyntaxError` – - thrown for any syntax error
- **rElif**
  `public org.openmrs.module.dssmodule.ast.AST` **`rElif`**`() throws org.openmrs.module.dssmodule.parser.SyntaxError`
  - **Description**
    Elif ->elsif EE 'then' BLOCK Elif ==>elsif ->elsif EE 'then' BLOCK ('else' BLOCK)? ==>elsif
- **rExpr**
  `public org.openmrs.module.dssmodule.ast.AST` **`rExpr`**`() throws org.openmrs.module.dssmodule.parser.SyntaxError`
  - **Description**

    ```
    e -> se
    -> se '==' se ==> =
    -> se '!=' se ==> !=
    -> se '' se ==>
    -> se '=' se ==> =
    ```

  - **Returns** – the tree corresponding to the expression
  - **Throws**

    - org.openmrs.module.dssmodule.parser.SyntaxError – - thrown for any syntax error

- **rFactor**
  `public org.openmrs.module.dssmodule.ast.AST` **rFactor**`() throws org.openmrs.module.dssmodule.parser.SyntaxError`
  - **Description**

    ```
    f -> '(' ee ')'
    -> name
    ->
    -> name '(' (ee list ',')? ')' ==> call
    -> Object '(' (NAME list ',')? ')'    ==> ObjectDecl
    -> new NAME          ==> Object
    -> NAME '.' NAME     ==> fieldRef
    -> LIST
    ```

  - **Returns** – the tree corresponding to the factor expression
  - **Throws**
    - org.openmrs.module.dssmodule.parser.SyntaxError – - thrown for any syntax error

- **rFunHead**
  `public org.openmrs.module.dssmodule.ast.AST` **rFunHead**`() throws org.openmrs.module.dssmodule.parser.SyntaxError`
  - **Description**
    pre>funHead ->'(' (NAME list ',')? ')' ==>formals note a funhead is a list of zero or more decl's separated by commas, all in parens
  - **Returns** – the formals tree describing this list of formals
  - **Throws**
    - org.openmrs.module.dssmodule.parser.SyntaxError – - thrown for any syntax error

- **rList**
  `public org.openmrs.module.dssmodule.ast.AST` **rList**`() throws org.openmrs.module.dssmodule.parser.SyntaxError`

  - **Description**
    LIST ->'{' (EE list ',')? '}' ==>list

- **rName**
  `public org.openmrs.module.dssmodule.ast.AST` **rName**`() throws org.openmrs.module.dssmodule.parser.SyntaxError`

  - **Description**

    ```
    name ->
    ```

  - **Returns** – the id tree
  - **Throws**
    * `org.openmrs.module.dssmodule.parser.SyntaxError` – - thrown for any syntax error

- **rNList**
  `public org.openmrs.module.dssmodule.ast.AST` **rNList**`() throws org.openmrs.module.dssmodule.parser.SyntaxError`

  - **Description**
    NLIST ->NAME ->LIST
  - **Returns** –
  - **Throws**
    * `org.openmrs.module.dssmodule.parser.SyntaxError` –

- **rPowerTerm**
  `public org.openmrs.module.dssmodule.ast.AST` **rPowerTerm**`() throws org.openmrs.module.dssmodule.parser.SyntaxError`

  - **Description**
    tt ->F ->TT**F

- **rProgram**
  `public org.openmrs.module.dssmodule.ast.AST` **rProgram**`() throws org.openmrs.module.dssmodule.parser.SyntaxError`

  - **Description**

    pre>PROGRAM ->program D* BLOCK ==>program

  - **Returns** – the program tree
  - **Throws**
    * `org.openmrs.module.dssmodule.parser.SyntaxError` – - thrown for any syntax error

- **rSimpleExpr**

  `public org.openmrs.module.dssmodule.ast.AST` **rSimpleExpr**`() throws org.openmrs.module.dssmodule.parser.SyntaxError`

  - **Description**

    ```
    se -> t
    -> se '+' t ==> +
    -> se '-' t ==> -
    -> se '|' t ==> or
    This
    rule indicates we should pick up as many  t's as possible; the
     t's will be left associative
    ```

  - **Returns** – the tree corresponding to the adding expression
  - **Throws**
    * `org.openmrs.module.dssmodule.parser.SyntaxError` – - thrown for any syntax error

- **rStatement**

  `public org.openmrs.module.dssmodule.ast.AST` **rStatement**`() throws org.openmrs.module.dssmodule.parser.SyntaxError`

  - **Description**

    ```
    S ->  if EE then BLOCK ( rElif* 'else' BLOCK)? ==> if
    -> while EE BLOCK ==> while
    -> 'for' NAME in NLIST BLOCK ==> FOR
    -> return EE ==> return
    ```

```
-> BLOCK
-> IdMod := EE ==> assign
-> FieldRef ':=" EE             ==> assign
-> name '(' (ee list ',')? ) ==> call
```

  - **Returns** – the tree corresponding to the statement found
  - **Throws**
    - org.openmrs.module.dssmodule.parser.SyntaxError – - thrown for any syntax error

- **rTerm**
  `public org.openmrs.module.dssmodule.ast.AST` **rTerm**`() throws org.openmrs.module.dssmodule.parser.SyntaxError`

  - **Description**

```
t -> tt
-> t '*' tt ==> *
-> t '/' tt ==> /
-> t '&' tt ==> and

This
rule indicates we should pick up as many  tt's as possible; the
 tt's will be left associative
```

  - **Returns** – the tree corresponding to the multiplying expression
  - **Throws**
    - org.openmrs.module.dssmodule.parser.SyntaxError – - thrown for any syntax error

C.6 Package org.openmrs.module.dssmodule.intrinsics

Package Contents *Page*

Interfaces

Classes

C.6.1 Interface DSSIdentifier

Annotation which indicates that an argument should be passed as a raw identifier when interpreted in a DSS1 program.

Declaration public interface DSSIdentifier
extends java.lang.annotation.Annotation

C.6.2 Interface DSSIntrinsic

Annotation which indicates that a method should be exposed as an intrinsic function, when declared by a subclass of AnnotatedDSSLibrary.

Declaration public interface DSSIntrinsic
extends java.lang.annotation.Annotation

C.6.3 Interface DSSLibrary

Base class for libraries. Used as a convenience for organizing related intrinsics.

Declaration public interface DSSLibrary

All known subinterfaces DSSRuleService.DSSAlertMap (in C.3.4, page 80), DemoLibrary (in C.6.6, page 103), IsLibrary (in C.6.18, page 116), ReadLibrary (in C.6.21, page 120), DateLibrary (in C.6.5, page 102), AnnotatedDSSLibrary (in C.6.4, page 101), LengthAndWithinLibrary (in C.6.19, page 117), DSSAlert (in C.6.9, page 107), ListLibrary (in C.6.20, page 118)

All classes known to implement interface IsLibrary (in C.6.18, page 116), ReadLibrary (in C.6.21, page 120), DateLibrary (in C.6.5, page 102), AnnotatedDSSLibrary (in C.6.4, page 101), LengthAndWithinLibrary (in C.6.19, page 117), DSSAlert (in C.6.9, page 107)

Method summary

getFunctions(ExecutionContext) Get all functions defined in this library.

Methods

- **getFunctions**
  `java.util.Map` **getFunctions**`(org.openmrs.module.dssmodule.state.ExecutionContext` **context**`)`

  - **Description**
    Get all functions defined in this library. These should be returned in a map from function name ->function object. Function name should be the name used to call the function from a DSS program (e.g. "read"); function object should be a Java object that extends DSSFunction (e.g. new DSSRead()) An ExecutionContext is provided in case a library needs to use or link against elements in that environment. DSSLibrary implementations are free to ignore this argument.
  - **Parameters**
    - `context` – the execution context to use for interactions
  - **Returns** – a map of functions defined by this library

C.6.4 Class AnnotatedDSSLibrary

An AnnotatedDSSLibrary will use reflection to examine its own methods and generate DSSFunctions if they are annotated as @DSSIntrinsic. See DemoLibrary for an example of this use. Arguments and return types will be converted to/from the corresponding DSS data types where necessary (although you can also use DSSValue or more specific types when desired.) Short synopsis of usage: - Create a class that extends AnnotatedDSSLibrary - Annotate methods which should be exposed as intrinsics with @DSSIntrinsic - Annotate arguments which use identifier syntax with @DSSIdentifier - Install the class in an interpreter (see DSSProgram.INTRINSICS) The following conversions should be supported (meaning that methods should generally restrict arguments/return values to the types on the right hand side.) DSS Type Java type ——————————————- integer ->long float ->double boolean ->boolean date ->java.util.Date string ->java.lang.String list ->java.util.List object ->java.util.Map

Declaration public abstract class AnnotatedDSSLibrary
extends java.lang.Object
implements DSSLibrary

All known subclasses DSSRuleService.DSSAlertMap (in C.3.4, page 80), DemoLibrary (in C.6.6, page 103), ListLibrary (in C.6.20, page 118)

Constructor summary

AnnotatedDSSLibrary()

Method summary

getFunctions(ExecutionContext)

Constructors

- **AnnotatedDSSLibrary**
  `public` **AnnotatedDSSLibrary**`()`

Methods

- **getFunctions**
  `java.util.Map` **getFunctions**`(org.openmrs.module.dssmodule.state.ExecutionContext` **context**`)`

- **Description copied from DSSLibrary (in C.6.3, page 100)**

  Get all functions defined in this library. These should be returned in a map from function name ->function object. Function name should be the name used to call the function from a DSS program (e.g. ”read”); function object should be a Java object that extends DSSFunction (e.g. new DSSRead()) An ExecutionContext is provided in case a library needs to use or link against elements in that environment. DSSLibrary implementations are free to ignore this argument.
- **Parameters**
  - `context` – the execution context to use for interactions
- **Returns** – a map of functions defined by this library

C.6.5 Class DateLibrary

Declaration public class DateLibrary
extends java.lang.Object
implements DSSLibrary

Constructor summary

DateLibrary()

Method summary

getFunctions(ExecutionContext)

Constructors

- **DateLibrary**
  `public` **DateLibrary**`()`

Methods

- **getFunctions**
  `java.util.Map` **getFunctions**`(org.openmrs.module.dssmodule.state.ExecutionContext` **context**`)`
  - **Description copied from DSSLibrary (in C.6.3, page 100)**

    Get all functions defined in this library. These should be returned in a map from function name ->function object. Function name should be the name used to call the function from a DSS program (e.g. ”read”); function object should be a

Java object that extends DSSFunction (e.g. new DSSRead()) An ExecutionContext is provided in case a library needs to use or link against elements in that environment. DSSLibrary implementations are free to ignore this argument.

- **Parameters**
  - `context` – the execution context to use for interactions
- **Returns** – a map of functions defined by this library

C.6.6 Class DemoLibrary

Demonstrates the use of AnnotatedDSSLibrary to semi-automatically create intrinsics. This library provides some basic math functions (sin, min, max) and a log function (similar to alert) as intrinsics. See the DSS source at the bottom for an example of how these methods appear & can be used by a DSS program.

Declaration public class DemoLibrary
extends org.openmrs.module.dssmodule.intrinsics.AnnotatedDSSLibrary (in C.6.4, page 101)

Constructor summary

DemoLibrary()

Method summary

log(String, String)
main(String[])
max(DSSValue, DSSValue)
min(DSSValue, DSSValue)
seq(int, int)
sin(double)
sum(DSSValueList)

Constructors

- **DemoLibrary**
  `public` **DemoLibrary**`()`

Methods

- **log**
  `public void` **log**`(java.lang.String` **level**`, java.lang.String` **message**`)`

- **main**
  `public static void` **main**`(java.lang.String[]` **args**`)`

- **max**
  `public org.openmrs.module.dssmodule.value.DSSValue`
  max`(org.openmrs.module.dssmodule.value.DSSValue` **a**`,`
  `org.openmrs.module.dssmodule.value.DSSValue` **b**`)`

- **min**
  `public org.openmrs.module.dssmodule.value.DSSValue`
  min`(org.openmrs.module.dssmodule.value.DSSValue` **a**`,`
  `org.openmrs.module.dssmodule.value.DSSValue` **b**`)`

- **seq**
  `public org.openmrs.module.dssmodule.value.DSSValue` **seq**`(int` **start**`, int` **end**`)`

- **sin**
  `public double` **sin**`(double` **r**`)`

- **sum**
  `public float` **sum**`(org.openmrs.module.dssmodule.value.DSSValueList` **list**`)`

Members inherited from class AnnotatedDSSLibrary
`org.openmrs.module.dssmodule.intrinsics.AnnotatedDSSLibrary` (in C.6.4, page 101)

getFunctions

C.6.7 Class DSSAddDays

addDays(time,numDays) - return a new time based on numDays The display format I used will match the system format. Default format yyyy-MM-dd hh:mm:ss e.g. Name Value System Date 2013-04-11 System Time 09:07:32

Declaration public class DSSAddDays
extends org.openmrs.module.dssmodule.state.DSSFunction (in C.2.5, page 69)

Constructor summary

DSSAddDays() constructor

Method summary

call(DSSValue[])

Constructors

- **DSSAddDays**
  `public` **DSSAddDays**`()`
  - **Description**
    constructor

Methods

- **call**
  `public abstract org.openmrs.module.dssmodule.value.DSSValue`
  call`(org.openmrs.module.dssmodule.value.DSSValue[]` **args**`)`
  - **Description copied from org.openmrs.module.dssmodule.state.DSSFunction (in C.2.5, page 69)**
    Call this function. The arguments provided are as observed by the interpreter. Note that the actual number of arguments may not match the number of parameters expected; it is ultimately the function's responsibility to handle this situation.
  - **Parameters**
    - `args` – the arguments to the function
  - **Returns** – the return value of the represented function

Members inherited from class DSSFunction `org.openmrs.module.dssmodule.state.DSSFunctic`
(in C.2.5, page 69)
call, passAsIdentifier

C.6.8 Class DSSAddMonths

addMonths(time,numMonths) - return a new time based on numMonths return a new time based on numMonths The display format I used will match the system format. Default format yyyy-MM-dd hh:mm:ss e.g. Name Value System Date 2013-04-11 System Time 09:07:32

Declaration public class DSSAddMonths
extends org.openmrs.module.dssmodule.state.DSSFunction (in C.2.5, page 69)

Constructor summary

DSSAddMonths()

Method summary

call(DSSValue[])

Constructors

- **DSSAddMonths**
  `public` **`DSSAddMonths`**`()`

Methods

- **call**
  `public abstract org.openmrs.module.dssmodule.value.DSSValue`
  `call``(org.openmrs.module.dssmodule.value.DSSValue[]` **`args`**`)`
  - **Description copied from org.openmrs.module.dssmodule.state.DSSFunction (in C.2.5, page 69)**
    Call this function. The arguments provided are as observed by the interpreter. Note that the actual number of arguments may not match the number of parameters expected; it is ultimately the function's responsibility to handle this situation.
  - **Parameters**
    - `args` – the arguments to the function
  - **Returns** – the return value of the represented function

Members inherited from class DSSFunction `org.openmrs.module.dssmodule.state.DSSFunctic`
(in C.2.5, page 69)
call, passAsIdentifier

C.6.9 Class DSSAlert

Implements the "alert" intrinsic; also exposes itself as a library. Note that this is not the class that will be used when running under OpenMRS; DSSRuleService defines its own version of this intrinsic to coordinate with insertion of alerts at appropriate locations. This version simply outputs to console.

See also

– `org.openmrs.module.dssmodule.DSSRuleService` (in C.3.3, page 78)

Declaration public class DSSAlert
extends org.openmrs.module.dssmodule.state.DSSFunction (in C.2.5, page 69)
implements DSSLibrary

Constructor summary

DSSAlert()

Method summary

call(DSSValue[])
getFunctions(ExecutionContext)

Constructors

- **DSSAlert**
  `public` **DSSAlert**`()`

Methods

- **call**
  `public abstract org.openmrs.module.dssmodule.value.DSSValue`
  call`(org.openmrs.module.dssmodule.value.DSSValue[]` **args**`)`

- **Description copied from org.openmrs.module.dssmodule.state.DSSFunction (in C.2.5, page 69)**

  Call this function. The arguments provided are as observed by the interpreter. Note that the actual number of arguments may not match the number of parameters expected; it is ultimately the function's responsibility to handle this situation.
- **Parameters**
  * `args` – the arguments to the function
- **Returns** – the return value of the represented function

- **getFunctions**
  `java.util.Map` **getFunctions**(`org.openmrs.module.dssmodule.state.ExecutionContext` **context**)

  - **Description copied from DSSLibrary (in C.6.3, page 100)**

    Get all functions defined in this library. These should be returned in a map from function name ->function object. Function name should be the name used to call the function from a DSS program (e.g. "read"); function object should be a Java object that extends DSSFunction (e.g. new DSSRead()) An ExecutionContext is provided in case a library needs to use or link against elements in that environment. DSSLibrary implementations are free to ignore this argument.
  - **Parameters**
    * `context` – the execution context to use for interactions
  - **Returns** – a map of functions defined by this library

Members inherited from class DSSFunction `org.openmrs.module.dssmodule.state.DSSFunctic`
(in C.2.5, page 69)
call, passAsIdentifier

C.6.10 Class DSSBefore

before(time1,time2) return true if time1 is before time2

Declaration public class DSSBefore
extends org.openmrs.module.dssmodule.state.DSSFunction (in C.2.5, page 69)

Constructor summary

DSSBefore()

Method summary

call(DSSValue[])

Constructors

- **DSSBefore**
  `public` **DSSBefore**`()`

Methods

- **call**
  `public abstract org.openmrs.module.dssmodule.value.DSSValue`
  call`(org.openmrs.module.dssmodule.value.DSSValue[]` **args**`)`
  - **Description copied from org.openmrs.module.dssmodule.state.DSSFunction (in C.2.5, page 69)**

    Call this function. The arguments provided are as observed by the interpreter. Note that the actual number of arguments may not match the number of parameters expected; it is ultimately the function's responsibility to handle this situation.
  - **Parameters**
    - `args` – the arguments to the function
  - **Returns** – the return value of the represented function

Members inherited from class DSSFunction `org.openmrs.module.dssmodule.state.DSSFunctic`
(in C.2.5, page 69)
call, passAsIdentifier

C.6.11 Class DSSCurrentTime

currenttime() return current time; e.g., Tue Nov 06 10:33:56 PST 2012 DSSCurrentTime class extends DSSFunction and return the current time in specific format. The display format I used will match the system format. Default format yyyy-MM-dd hh:mm:ss e.g. Name Value System Date 2013-04-11 System Time 09:07:32

Declaration public class DSSCurrentTime
extends org.openmrs.module.dssmodule.state.DSSFunction (in C.2.5, page 69)

Constructor summary

DSSCurrentTime()

Method summary

call(DSSValue[]**)** Construct a DSSValueDate object and return it.
main(String[]**)** Testing currenttime() function

Constructors

- **DSSCurrentTime**
  `public` **`DSSCurrentTime`**`()`

Methods

- **call**
  `public org.openmrs.module.dssmodule.value.DSSValue`
  `call``(org.openmrs.module.dssmodule.value.DSSValue[]` **`args`**`)`
  - **Description**
    Construct a DSSValueDate object and return it. Convert the DSSValue using getDSSValue method in DSSValue Factory.java
  - **Returns** – DSSValueDate: a new data represents the current time
- **main**
  `public static void` **`main`**`(java.lang.String[]` **`args`**`)`
  - **Description**
    Testing currenttime() function

Members inherited from class DSSFunction `org.openmrs.module.dssmodule.state.DSSFunctic`
(in C.2.5, page 69)
call, passAsIdentifier

C.6.12 Class DSSOldestTimeItem

oldestTimeItem(list) return the item with the oldest time The display format I used will match the system format. Default format yyyy-MM-dd hh:mm:ss e.g. Name Value System Date 2013-04-11 System Time 09:07:32

Declaration public class DSSOldestTimeItem
extends org.openmrs.module.dssmodule.state.DSSFunction (in C.2.5, page 69)

Constructor summary

DSSOldestTimeItem()

Method summary

call(DSSValue[])

Constructors

- **DSSOldestTimeItem**
  `public` **DSSOldestTimeItem**`()`

Methods

- **call**
  `public abstract org.openmrs.module.dssmodule.value.DSSValue`
  call`(org.openmrs.module.dssmodule.value.DSSValue[]` **args**`)`
  - **Description copied from org.openmrs.module.dssmodule.state.DSSFunction (in C.2.5, page 69)**
    Call this function. The arguments provided are as observed by the interpreter. Note that the actual number of arguments may not match the number of parameters expected; it is ultimately the function's responsibility to handle this situation.
  - **Parameters**
    - `args` – the arguments to the function
  - **Returns** – the return value of the represented function

Members inherited from class DSSFunction `org.openmrs.module.dssmodule.state.DSSFunctic`
(in C.2.5, page 69)
call, passAsIdentifier

C.6.13 Class DSSRead

Declaration public class DSSRead
extends org.openmrs.module.dssmodule.state.DSSFunction (in C.2.5, page 69)

All known subclasses DSSReadLatestEncounter (in C.6.15, page 113), DSSReadInitialEncounter (in C.6.14, page 112)

Constructor summary

DSSRead()

Method summary

call(DSSValue[]) Call this function.

Constructors

- **DSSRead**
  `public` **DSSRead**`()`

Methods

- **call**
  `public org.openmrs.module.dssmodule.value.DSSValue`
  call`(org.openmrs.module.dssmodule.value.DSSValue[]` **args**`)`
  - **Description**
    Call this function. The arguments provided are as observed by the interpreter.
  - **Parameters**
    - `args` – the arguments to the function
  - **Returns** – a list containing all observations for a given concept associated with a patient DSSNullValue for argument mismatch

Members inherited from class DSSFunction `org.openmrs.module.dssmodule.state.DSSFunctic`
(in C.2.5, page 69)
call, passAsIdentifier

C.6.14 Class DSSReadInitialEncounter

Declaration public class DSSReadInitialEncounter
extends org.openmrs.module.dssmodule.intrinsics.DSSRead (in C.6.13, page 111)

Constructor summary

DSSReadInitialEncounter()

Method summary

call(DSSValue[]) Call this function.

Constructors

- **DSSReadInitialEncounter**
  `public` **`DSSReadInitialEncounter`**`()`

Methods

- **call**
  `public org.openmrs.module.dssmodule.value.DSSValue`
  `call``(org.openmrs.module.dssmodule.value.DSSValue[]` **`args`**`)`
  - **Description**
    Call this function. The arguments provided are as observed by the interpreter.
  - **Parameters**
    - `args` – the arguments to the function
  - **Returns** – a list containing all observations for a given concept belonging to the first encounter associated with a patient DSSNullValue for argument mismatch

Members inherited from class DSSRead `org.openmrs.module.dssmodule.intrinsics.DSSRead`
(in C.6.13, page 111)
call

Members inherited from class DSSFunction `org.openmrs.module.dssmodule.state.DSSFunctio`
(in C.2.5, page 69)
call, passAsIdentifier

C.6.15 Class DSSReadLatestEncounter

Declaration public class DSSReadLatestEncounter
extends org.openmrs.module.dssmodule.intrinsics.DSSRead (in C.6.13, page 111)

Constructor summary

DSSReadLatestEncounter()

Method summary

call(DSSValue[]) Call this function.

Constructors

- **DSSReadLatestEncounter**
  `public` **DSSReadLatestEncounter**`()`

Methods

- **call**
  `public org.openmrs.module.dssmodule.value.DSSValue`
  call`(org.openmrs.module.dssmodule.value.DSSValue[]` **args**`)`
  - **Description**
    Call this function. The arguments provided are as observed by the interpreter.
  - **Parameters**
    - `args` – the arguments to the function
  - **Returns** – a list containing all observations for a given concept belonging to the last encounter associated with a patient DSSNullValue for argument mismatch

Members inherited from class DSSRead `org.openmrs.module.dssmodule.intrinsics.DSSRead`
(in C.6.13, page 111)
call

Members inherited from class DSSFunction `org.openmrs.module.dssmodule.state.DSSFunctic`
(in C.2.5, page 69)
call, passAsIdentifier

C.6.16 Class DSSRecentTimeItem

recentTimeItem(list) return the item with the most recent time DSSCurrentTime class extends DSSFunction and return the current time in specific format.DSSCurrentTIme also owns instances of DSSFunction. The display format will match the system format. Default format yyyy-MM-dd hh:mm:ss e.g. Name Value System Date 2013-04-11 System Time 09:07:32

Declaration public class DSSRecentTimeItem
extends org.openmrs.module.dssmodule.state.DSSFunction (in C.2.5, page 69)

Constructor summary

DSSRecentTimeItem()

Method summary

call(DSSValue[])

Constructors

- **DSSRecentTimeItem**
  `public` **DSSRecentTimeItem**`()`

Methods

- **call**
  `public abstract org.openmrs.module.dssmodule.value.DSSValue`
  call`(org.openmrs.module.dssmodule.value.DSSValue[]` **args**`)`
  - **Description copied from org.openmrs.module.dssmodule.state.DSSFunction (in C.2.5, page 69)**

    Call this function. The arguments provided are as observed by the interpreter. Note that the actual number of arguments may not match the number of parameters expected; it is ultimately the function's responsibility to handle this situation.
  - **Parameters**
    - `args` – the arguments to the function
  - **Returns** – the return value of the represented function

Members inherited from class DSSFunction `org.openmrs.module.dssmodule.state.DSSFunctic`
(in C.2.5, page 69)
call, passAsIdentifier

C.6.17 Class DSSTime

time(v) return time associated with v The display format I used will match the system format. Default format yyyy-MM-dd hh:mm:ss e.g. Name Value System Date 2013-04-11 System Time 09:07:32

Declaration public class DSSTime
extends org.openmrs.module.dssmodule.state.DSSFunction (in C.2.5, page 69)

Constructor summary

DSSTime()

Method summary

call(DSSValue[])

Constructors

- **DSSTime**
  `public` **DSSTime**`()`

Methods

- **call**
  `public abstract org.openmrs.module.dssmodule.value.DSSValue`
  call`(org.openmrs.module.dssmodule.value.DSSValue[]` **args**`)`
  - **Description copied from org.openmrs.module.dssmodule.state.DSSFunction (in C.2.5, page 69)**

    Call this function. The arguments provided are as observed by the interpreter. Note that the actual number of arguments may not match the number of parameters expected; it is ultimately the function's responsibility to handle this situation.
  - **Parameters**
    - `args` – the arguments to the function
  - **Returns** – the return value of the represented function

Members inherited from class DSSFunction `org.openmrs.module.dssmodule.state.DSSFunctic`
(in C.2.5, page 69)
call, passAsIdentifier

C.6.18 Class IsLibrary

The various "is" intrinsics (isString, isInteger, etc)

Declaration public class IsLibrary
extends java.lang.Object
implements DSSLibrary

Constructor summary

IsLibrary()

Method summary

getFunctions(ExecutionContext)

Constructors

- **IsLibrary**
  `public` **IsLibrary**`()`

Methods

- **getFunctions**
  `java.util.Map` **getFunctions**`(org.openmrs.module.dssmodule.state.ExecutionContext` **context**`)`
  - **Description copied from DSSLibrary (in C.6.3, page 100)**
    Get all functions defined in this library. These should be returned in a map from function name ->function object. Function name should be the name used to call the function from a DSS program (e.g. ”read”); function object should be a Java object that extends DSSFunction (e.g. new DSSRead()) An ExecutionContext is provided in case a library needs to use or link against elements in that environment. DSSLibrary implementations are free to ignore this argument.
  - **Parameters**
    - `context` – the execution context to use for interactions
  - **Returns** – a map of functions defined by this library

C.6.19 Class LengthAndWithinLibrary

Declaration public class LengthAndWithinLibrary
extends java.lang.Object
implements DSSLibrary

Constructor summary

LengthAndWithinLibrary()

Method summary

getFunctions(ExecutionContext)

Constructors

- **LengthAndWithinLibrary**
  `public` **LengthAndWithinLibrary**`()`

Methods

- **getFunctions**
  `java.util.Map` **getFunctions**`(org.openmrs.module.dssmodule.state.ExecutionContext` **context**`)`
  - **Description copied from DSSLibrary** (**in C.6.3, page 100**)
    Get all functions defined in this library. These should be returned in a map from function name ->function object. Function name should be the name used to call the function from a DSS program (e.g. ”read”); function object should be a Java object that extends DSSFunction (e.g. new DSSRead()) An ExecutionContext is provided in case a library needs to use or link against elements in that environment. DSSLibrary implementations are free to ignore this argument.
  - **Parameters**
    * `context` – the execution context to use for interactions
  - **Returns** – a map of functions defined by this library

C.6.20 Class ListLibrary

Implements and organizes list-related intrinsic functions.

Declaration public class ListLibrary
extends org.openmrs.module.dssmodule.intrinsics.AnnotatedDSSLibrary (in C.6.4, page 101)

Constructor summary

ListLibrary()

Method summary

first(List)
get(List, int) Utility method to get a DSSValue from a list, with bounds checking.
last(List)
merge(Collection, Collection)
sortData(Collection)
sortTime(Collection)

Constructors

- **ListLibrary**
  `public` **ListLibrary**`()`

Methods

- **first**
  `public org.openmrs.module.dssmodule.value.DSSValue` **first**`(java.util.List` **list**`)`
- **get**
  `public org.openmrs.module.dssmodule.value.DSSValue` **get**`(java.util.List` **list**`, int` **index**`)`
  - **Description**
    Utility method to get a DSSValue from a list, with bounds checking. Defaults to DSS null when out of bounds. Note that this is not actual public API; however, it cannot be private without blocking access to the intrinsic functions that AnnotatedDSSLibrary generates.
  - **Parameters**
    - `list` –
    - `index` –
  - **Returns** –
- **last**
  `public org.openmrs.module.dssmodule.value.DSSValue` **last**`(java.util.List` **list**`)`
- **merge**
  `public java.util.List` **merge**`(java.util.Collection` **a**`, java.util.Collection` **b**`)`

- **sortData**
  `public java.util.List` **sortData**`(java.util.Collection` **c**`)`

- **sortTime**
  `public java.util.List` **sortTime**`(java.util.Collection` **c**`)`

Members inherited from class AnnotatedDSSLibrary
`org.openmrs.module.dssmodule.intrinsics.AnnotatedDSSLibrary` (in C.6.4, page 101)

getFunctions

C.6.21 Class ReadLibrary

Declaration public class ReadLibrary
extends java.lang.Object
implements DSSLibrary

Constructor summary

ReadLibrary()

Method summary

getFunctions(ExecutionContext)

Constructors

- **ReadLibrary**
  `public` **ReadLibrary**`()`

Methods

- **getFunctions**
  `java.util.Map` **getFunctions**`(org.openmrs.module.dssmodule.state.ExecutionContext` **context**`)`
  - **Description copied from DSSLibrary (in C.6.3, page 100)**
    Get all functions defined in this library. These should be returned in a map from function name ->function object. Function name should be the name used to call the function from a DSS program (e.g. ”read”); function object should be a Java object that extends DSSFunction (e.g. new DSSRead()) An ExecutionContext is provided in case a library needs to use or link against elements in that environment. DSSLibrary implementations are free to ignore this argument.

- **Parameters**
  - `context` – the execution context to use for interactions
- **Returns** – a map of functions defined by this library

C.7 Package org.openmrs.module.dssmodule.lexer

Package Contents *Page*

Classes

C.7.1 Class Lexer

The Lexer class is responsible for scanning the source file which is a stream of characters and returning a stream of tokens; each token object will contain the string (or access to the string) that describes the token along with an indication of its location in the source program to be used for error reporting; we are tracking line numbers; white spaces are space, tab, newlines

Declaration public class Lexer
extends java.lang.Object

Constructor summary

Lexer(String)

Method summary

makeToken(String, int, int) build the token for operators (+ -) or separators (parens, braces) filter out comments which begin with two slashes
newIdToken(String, int, int) newIdTokens are either ids or reserved words; new id's will be inserted in the symbol table with an indication that they are id's
newNumberToken(String, int, int) number tokens are inserted in the symbol table; we don't convert the numeric strings to numbers until we load the bytecodes for interpreting; this ensures that any machine numeric dependencies are deferred until we actually run the program; i.e.
newStringToken(String, int, int)
nextToken()

Constructors

- **Lexer**
  `public` **Lexer**`(java.lang.String` **sourceFile**`) throws java.lang.Exception`

Methods

- **makeToken**
  `public Token` **makeToken**`(java.lang.String s, int` **startPosition**`, int` **endPosition**`)`
  - **Description**
    build the token for operators (+ -) or separators (parens, braces) filter out comments which begin with two slashes
  - **Parameters**

        - `s` – is the String representing the token
        - `startPosition` – is the column in the source file where the token begins
        - `endPosition` – is the column in the source file where the token ends
    - **Returns** – the Token just found

- **newIdToken**
  `public Token` **newIdToken**`(java.lang.String` **id**`, int` **startPosition**`, int` **endPosition**`)`
    - **Description**
      newIdTokens are either ids or reserved words; new id's will be inserted in the symbol table with an indication that they are id's
    - **Parameters**
        - `id` – is the String just scanned - it's either an id or reserved word
        - `startPosition` – is the column in the source file where the token begins
        - `endPosition` – is the column in the source file where the token ends
    - **Returns** – the Token; either an id or one for the reserved words

- **newNumberToken**
  `public Token` **newNumberToken**`(java.lang.String` **number**`, int` **startPosition**`, int` **endPosition**`)`
    - **Description**
      number tokens are inserted in the symbol table; we don't convert the numeric strings to numbers until we load the bytecodes for interpreting; this ensures that any machine numeric dependencies are deferred until we actually run the program; i.e. the numeric constraints of the hardware used to compile the source program are not used
    - **Parameters**
        - `number` – is the int String just scanned
        - `startPosition` – is the column in the source file where the int begins
        - `endPosition` – is the column in the source file where the int ends
    - **Returns** – the int Token

- **newStringToken**
  `public Token` **newStringToken**`(java.lang.String` **str**`, int` **startPosition**`,`
  `int` **endPosition**`)`

- **nextToken**
  `public Token` **nextToken**`()`

  - **Returns** – the next Token found in the source file

C.7.2 Class SourceReader

This class is used to manage the source program input stream; each read request will return the next usable character; it maintains the source column position of the character

Declaration public class SourceReader
extends java.lang.Object

Constructor summary

SourceReader(String) Construct a new SourceReader

Method summary

getLineno()
getPosition()
read() read next char; track line #, character position in line
return space for newline

Constructors

- **SourceReader**
  `public` **SourceReader**`(java.lang.String` **sourceFile**`) throws`
  `java.io.IOException`

  - **Description**
    Construct a new SourceReader
  - **Parameters**
    - `sourceFile` – the String describing the user's source file
  - **Throws**
    - `java.io.IOException` – is thrown if there is an I/O problem

Methods

- **getLineno**
  `public int getLineno()`
  - **Returns** – the line number of the character just read in
- **getPosition**
  `public int getPosition()`
  - **Returns** – the position of the character just read in
- **read**
  `public char read() throws java.io.IOException`
  - **Description**
    read next char; track line #, character position in line
    return space for newline
  - **Returns** – the character just read in

C.7.3 Class Symbol

The Symbol class is used to store all user strings along with an indication of the kind of strings they are; e.g. the id ”abc” will store the ”abc” in name and Sym.Tokens.Identifier in kind

Declaration public class Symbol
extends java.lang.Object

Method summary

getKind()
symbol(String, Tokens) Return the unique symbol associated with a string.
toString()

Methods

- **getKind**
  `public Tokens` **getKind**`()`

- **symbol**
  `public static Symbol` **symbol**`(java.lang.String` **newTokenString**`, Tokens` **kind**`)`

  - **Description**
    Return the unique symbol associated with a string. Repeated calls to `symbol("abc")` will return the same Symbol.

- **toString**
  `public java.lang.String` **toString**`()`

C.7.4 Class Token

```
The Token class records the information for a token:
1. The Symbol that describes the characters in the token
2. The starting column in the source file of the token and
3. The ending column in the source file of the token
```

Declaration public class Token
extends java.lang.Object

Constructor summary

Token(int, int, Symbol) Create a new Token based on the given Symbol

Method summary

getKind()
getLeftPosition()
getRightPosition()
getSymbol()

print()
toString()

Constructors

- **Token**
  `public` **Token**`(int` **leftPosition**`, int` **rightPosition**`, Symbol` **sym**`)`
  - **Description**
    Create a new Token based on the given Symbol
  - **Parameters**
    * `leftPosition` – is the source file column where the Token begins
    * `rightPosition` – is the source file column where the Token ends

Methods

- **getKind**
  `public Tokens` **getKind**`()`
  - **Returns** – the integer that represents the kind of symbol we have which is actually the type of token associated with the symbol
- **getLeftPosition**
  `public int` **getLeftPosition**`()`
- **getRightPosition**
  `public int` **getRightPosition**`()`
- **getSymbol**
  `public Symbol` **getSymbol**`()`
- **print**
  `public void` **print**`()`
- **toString**
  `public java.lang.String` **toString**`()`

C.7.5 Class Tokens

This file is automatically generated
- it contains the enumeration of all of the tokens

Declaration public final class Tokens
extends java.lang.Enum

Field summary

- **And**
- **Assign**
- **BogusToken**
- **BOOLean**
- **Comma**
- **Comment**
- **Concat**
- **Divide**
- **Dot**
- **Else**
- **Elsif**
- **Equal**
- **False**
- **Float**
- **For**
- **Function**
- **Identifier**
- **If**
- **In**
- **Int**
- **INTeger**
- **LeftBrace**
- **LeftParen**
- **Less**
- **LessEqual**
- **Minus**
- **Multiply**
- **New**
- **Not**
- **NotEqual**
- **Null**
- **Object**

Or
Plus
Power
Program
Return
RightBrace
RightParen
STRing
Then
True
While

Method summary

valueOf(String)
values()

Fields

- public static final Tokens **BogusToken**
- public static final Tokens **Program**
- public static final Tokens **Int**
- public static final Tokens **BOOLean**
- public static final Tokens **If**
- public static final Tokens **Then**
- public static final Tokens **Else**
- public static final Tokens **Elsif**
- public static final Tokens **While**
- public static final Tokens **For**
- public static final Tokens **Function**

- public static final Tokens **Return**
- public static final Tokens **New**
- public static final Tokens **In**
- public static final Tokens **True**
- public static final Tokens **False**
- public static final Tokens **Null**
- public static final Tokens **Identifier**
- public static final Tokens **INTeger**
- public static final Tokens **Float**
- public static final Tokens **STRing**
- public static final Tokens **Object**
- public static final Tokens **LeftBrace**
- public static final Tokens **RightBrace**
- public static final Tokens **LeftParen**
- public static final Tokens **RightParen**
- public static final Tokens **Comma**
- public static final Tokens **Dot**
- public static final Tokens **Assign**
- public static final Tokens **Equal**
- public static final Tokens **NotEqual**
- public static final Tokens **Not**
- public static final Tokens **Less**

- public static final Tokens **LessEqual**
- public static final Tokens **Plus**
- public static final Tokens **Minus**
- public static final Tokens **Or**
- public static final Tokens **And**
- public static final Tokens **Multiply**
- public static final Tokens **Power**
- public static final Tokens **Divide**
- public static final Tokens **Concat**
- public static final Tokens **Comment**

Methods

- **valueOf**
  `public static Tokens` **valueOf**`(java.lang.String` **name**`)`
- **values**
  `public static Tokens[]` **values**`()`

Members inherited from class Enum `java.lang.Enum`
compareTo, equals, getDeclaringClass, hashCode, name, ordinal, toString, valueOf

C.7.6 Class TokenType

This file is automatically generated
it contains the table of mappings from token constants to their Symbols

Declaration public class TokenType
extends java.lang.Object

Field summary

tokens

Constructor summary

TokenType()

Fields

- public static java.util.HashMap **tokens**

Constructors

- **TokenType**
  `public` **TokenType**`()`

C.8 Package org.openmrs.module.dssmodule.flowcontrol

Package Contents *Page*

Interfaces

Classes

Handles interpretation of ”if” and ”elsif” sub-trees.

InterpreterVisitor 141

Handles flow control and interactions with an execution context to interpret a DSS1 program.

ListInterpreter 144

Handles interpretation of List literals

LiteralInterpreter 145

Handles interpretation of Literal nodes

ObjectDeclInterpreter 145

Handles interpretation of object declarations.

ObjectInterpreter 146

Handles interpretation of object sub-trees, used in DSS1 to instantiate new objects based on prototypes.

OpInterpreter 147

Handles operations in DSS1 (arithmetic, logical, et cetera).

ProgramInterpreter 147

Interpret a program tree.

ReturnInterpreter 148

Handle interpretation of return statements in a DSS1 program.

WhileInterpreter 149

Handles interpretation of while loops in a DSS1 program.

C.8.1 Interface ASTInterpreter

Describes the common interface used to handle interpretation of specific node types within a compiled AST.

Declaration public interface ASTInterpreter

All known subinterfaces FunctionDeclInterpreter (in C.8.8, page 139), ObjectInterpreter (in C.8.15, page 146), CallInterpreter (in C.8.4, page 136), ReturnInterpreter (in C.8.18, page 148), ListInterpreter (in C.8.12, page 144), IdInterpreter (in C.8.9, page 139), ForInterpreter (in C.8.6, page 137), IfInterpreter (in C.8.10, page 140), BlockInterpreter (in C.8.3, page 135), AssignInterpreter (in C.8.2, page 134), FieldRefInterpreter (in C.8.5, page 136), FormalsInterpreter (in C.8.7, page 138), LiteralInterpreter (in C.8.13, page 145), OpInterpreter (in C.8.16, page 147), ProgramInterpreter (in C.8.17, page 147), ObjectDeclInterpreter (in C.8.14, page 145), WhileInterpreter (in

C.8.19, page 149)

All classes known to implement interface FunctionDeclInterpreter (in C.8.8, page 139), ObjectInterpreter (in C.8.15, page 146), CallInterpreter (in C.8.4, page 136), ReturnInterpreter (in C.8.18, page 148), ListInterpreter (in C.8.12, page 144), IdInterpreter (in C.8.9, page 139), ForInterpreter (in C.8.6, page 137), IfInterpreter (in C.8.10, page 140), BlockInterpreter (in C.8.3, page 135), AssignInterpreter (in C.8.2, page 134), FieldRefInterpreter (in C.8.5, page 136), FormalsInterpreter (in C.8.7, page 138), LiteralInterpreter (in C.8.13, page 145), OpInterpreter (in C.8.16, page 147), ProgramInterpreter (in C.8.17, page 147), ObjectDeclInterpreter (in C.8.14, page 145), WhileInterpreter (in C.8.19, page 149)

Method summary

interpret(T, ExecutionContext, ASTVisitor) Interpret a given node in the tree.

Methods

- **interpret**
  `java.lang.Object` **interpret**(`org.openmrs.module.dssmodule.ast.AST` **tree**, `org.openmrs.module.dssmodule.state.ExecutionContext` **context**, `org.openmrs.module.dssmodule.visitor.ASTVisitor` **visitor**)
  - **Description**
    Interpret a given node in the tree. Note that the object returned may vary based on node type, but is commonly the result of interpretation (i.e. the value of an expression)
  - **Parameters**
    - `tree` – the tree to interpret
    - `context` – the execution (stores variable meanings, for ex)
    - `visitor` – the visitor which may interpret children
  - **Returns** –

C.8.2 Class AssignInterpreter

Interprets Assign sub-trees in a compiled AST

Declaration public class AssignInterpreter
extends java.lang.Object
implements ASTInterpreter

Constructor summary

AssignInterpreter()

Method summary

interpret(AssignTree, ExecutionContext, ASTVisitor)

Constructors

- **AssignInterpreter**
  `public` **AssignInterpreter**`()`

Methods

- **interpret**
  `public java.lang.Object` **interpret**`(org.openmrs.module.dssmodule.ast.AssignTree` **tree**`, org.openmrs.module.dssmodule.state.ExecutionContext` **context**`, org.openmrs.module.dssmodule.visitor.ASTVisitor` **visitor**`)`

C.8.3 Class BlockInterpreter

Interprets Block sub-trees in a compiled AST

Declaration public class BlockInterpreter
extends java.lang.Object
implements ASTInterpreter

Constructor summary

BlockInterpreter()

Method summary

interpret(BlockTree, ExecutionContext, ASTVisitor)

Constructors

- **BlockInterpreter**
  `public` **BlockInterpreter**`()`

Methods

- **interpret**
  `public java.lang.Object` **interpret**`(org.openmrs.module.dssmodule.ast.BlockTree` **tree**`, org.openmrs.module.dssmodule.state.ExecutionContext` **context**`, org.openmrs.module.dssmodule.visitor.ASTVisitor` **visitor**`)`

C.8.4 Class CallInterpreter

Handles 'call' nodes in the abstract syntax tree for DSS1

Declaration public class CallInterpreter
extends java.lang.Object
implements ASTInterpreter

Constructor summary

CallInterpreter()

Method summary

interpret(CallTree, ExecutionContext, ASTVisitor)

Constructors

- **CallInterpreter**
  `public` **CallInterpreter**`()`

Methods

- **interpret**
  `public java.lang.Object` **interpret**`(org.openmrs.module.dssmodule.ast.CallTree` **tree**`, org.openmrs.module.dssmodule.state.ExecutionContext` **context**`, org.openmrs.module.dssmodule.visitor.ASTVisitor` **visitor**`)`

C.8.5 Class FieldRefInterpreter

Interprets field references in a compiled AST. Per convention, this will interpret to the value held within a field.

Declaration public class FieldRefInterpreter
extends java.lang.Object
implements ASTInterpreter

Constructor summary

FieldRefInterpreter()

Method summary

interpret(FieldRefTree, ExecutionContext, ASTVisitor)

Constructors

- **FieldRefInterpreter**
  `public` **FieldRefInterpreter**`()`

Methods

- **interpret**
  `public java.lang.Object` **interpret**`(org.openmrs.module.dssmodule.ast.FieldRefTree` **tree**`, org.openmrs.module.dssmodule.state.ExecutionContext` **context**`, org.openmrs.module.dssmodule.visitor.ASTVisitor` **visitor**`)`

C.8.6 Class ForInterpreter

Interprets For loops in a compiled AST. Note that DSS grammar defines this as a ”for-each” style loop

Declaration public class ForInterpreter
extends java.lang.Object
implements ASTInterpreter

Constructor summary

ForInterpreter()

Method summary

interpret(ForTree, ExecutionContext, ASTVisitor)

Constructors

- **ForInterpreter**
  `public` **ForInterpreter**`()`

Methods

- **interpret**
  `public java.lang.Object` **interpret**`(org.openmrs.module.dssmodule.ast.ForTree` **tree**`, org.openmrs.module.dssmodule.state.ExecutionContext` **context**`, org.openmrs.module.dssmodule.visitor.ASTVisitor` **visitor**`)`

C.8.7 Class FormalsInterpreter

Handles interpretation of Formals in a DSS program. Note that these should never be visited directly under normal operation.

Declaration public class FormalsInterpreter
extends java.lang.Object
implements ASTInterpreter

Constructor summary

FormalsInterpreter()

Method summary

interpret(FormalsTree, ExecutionContext, ASTVisitor)

Constructors

- **FormalsInterpreter**
  `public` **FormalsInterpreter**`()`

Methods

- **interpret**
  `public java.lang.Object` **interpret**`(org.openmrs.module.dssmodule.ast.FormalsTree` **tree**`, org.openmrs.module.dssmodule.state.ExecutionContext` **context**`, org.openmrs.module.dssmodule.visitor.ASTVisitor` **visitor**`)`

C.8.8 Class FunctionDeclInterpreter

Handles function declaration in a DSS program. After executing, the described function should be available within the execution context.

Declaration public class FunctionDeclInterpreter
extends java.lang.Object
implements ASTInterpreter

Constructor summary

FunctionDeclInterpreter()

Method summary

interpret(FunctionDeclTree, ExecutionContext, ASTVisitor)

Constructors

- **FunctionDeclInterpreter**
  `public` **FunctionDeclInterpreter**`()`

Methods

- **interpret**
  `public java.lang.Object` **interpret**`(org.openmrs.module.dssmodule.ast.FunctionDeclTree` **tree**`, org.openmrs.module.dssmodule.state.ExecutionContext` **context**`, org.openmrs.module.dssmodule.visitor.ASTVisitor` **visitor**`)`

C.8.9 Class IdInterpreter

Evaluate an identifier. This evaluates as though it were an expression; interpreters that need to get the actual name of the identifier should look at the IdTree directly.

Declaration public class IdInterpreter
extends java.lang.Object
implements ASTInterpreter

Constructor summary

IdInterpreter()

Method summary

interpret(IdTree, ExecutionContext, ASTVisitor)

Constructors

- **IdInterpreter**
  `public` **IdInterpreter**`()`

Methods

- **interpret**
  `public java.lang.Object` **interpret**`(org.openmrs.module.dssmodule.ast.IdTree` **tree**`, org.openmrs.module.dssmodule.state.ExecutionContext` **context**`, org.openmrs.module.dssmodule.visitor.ASTVisitor` **visitor**`)`

C.8.10 Class IfInterpreter

Handles interpretation of ”if” and ”elsif” sub-trees.

Declaration public class IfInterpreter
extends java.lang.Object
implements ASTInterpreter

Constructor summary

IfInterpreter()

Method summary

interpret(AST, ExecutionContext, ASTVisitor)

Constructors

- **IfInterpreter**
  `public` **IfInterpreter**`()`

Methods

- **interpret**
  `public java.lang.Object` **interpret**`(org.openmrs.module.dssmodule.ast.AST` **tree**`, org.openmrs.module.dssmodule.state.ExecutionContext` **context**`, org.openmrs.module.dssmodule.visitor.ASTVisitor` **visitor**`)`

C.8.11 Class InterpreterVisitor

Handles flow control and interactions with an execution context to interpret a DSS1 program. Follows the visitor design pattern.

Declaration public class InterpreterVisitor
extends org.openmrs.module.dssmodule.visitor.ASTVisitor (in C.9.1, page 150)

Constructor summary

InterpreterVisitor(ExecutionContext) Create a new InterpreterVisitor.

Method summary

visitActualArgsTree(AST)
visitAssignTree(AST)
visitBlockTree(AST)
visitCallTree(AST)
visitElsifTree(AST)
visitFieldRefTree(AST)
visitFormalsTree(AST)
visitForTree(AST)
visitFunctionDeclTree(AST)
visitIdTree(AST)
visitIfTree(AST)
visitListTree(AST)
visitLiteralTree(AST)
visitObjectDeclTree(AST)
visitObjectTree(AST)
visitOpTree(AST)
visitProgramTree(AST)
visitReturnTree(AST)
visitWhileTree(AST)

Constructors

- **InterpreterVisitor**
  `public` **InterpreterVisitor**`(org.openmrs.module.dssmodule.state.ExecutionContext` **context**`)`

- **Description**

  Create a new InterpreterVisitor. This will operate using the specified ExecutionContext to mediate interactions with state or with the evaluation subsystem.

- **Parameters**

  * `context` –

Methods

- **visitActualArgsTree**

  `public abstract java.lang.Object` **visitActualArgsTree**`(org.openmrs.module.dssmodule.a`
  t`)`

- **visitAssignTree**

  `public abstract java.lang.Object` **visitAssignTree**`(org.openmrs.module.dssmodule.ast.AS`
  t`)`

- **visitBlockTree**

  `public abstract java.lang.Object` **visitBlockTree**`(org.openmrs.module.dssmodule.ast.AST`
  t`)`

- **visitCallTree**

  `public abstract java.lang.Object` **visitCallTree**`(org.openmrs.module.dssmodule.ast.AST`
  t`)`

- **visitElsifTree**

  `public abstract java.lang.Object` **visitElsifTree**`(org.openmrs.module.dssmodule.ast.AST`
  t`)`

- **visitFieldRefTree**

  `public abstract java.lang.Object` **visitFieldRefTree**`(org.openmrs.module.dssmodule.ast.`
  t`)`

- **visitFormalsTree**

  `public abstract java.lang.Object` **visitFormalsTree**`(org.openmrs.module.dssmodule.ast.`
  t`)`

- **visitForTree**

  `public abstract java.lang.Object` **visitForTree**`(org.openmrs.module.dssmodule.ast.AST`
  t`)`

- **visitFunctionDeclTree**
  public abstract java.lang.Object **visitFunctionDeclTree**(org.openmrs.module.dssmodule.ast.AST **t**)

- **visitIdTree**
  public abstract java.lang.Object **visitIdTree**(org.openmrs.module.dssmodule.ast.AST **t**)

- **visitIfTree**
  public abstract java.lang.Object **visitIfTree**(org.openmrs.module.dssmodule.ast.AST **t**)

- **visitListTree**
  public abstract java.lang.Object **visitListTree**(org.openmrs.module.dssmodule.ast.AST **t**)

- **visitLiteralTree**
  public abstract java.lang.Object **visitLiteralTree**(org.openmrs.module.dssmodule.ast.AS
  t)

- **visitObjectDeclTree**
  public abstract java.lang.Object **visitObjectDeclTree**(org.openmrs.module.dssmodule.ast.AST **t**)

- **visitObjectTree**
  public abstract java.lang.Object **visitObjectTree**(org.openmrs.module.dssmodule.ast.AS
  t)

- **visitOpTree**
  public abstract java.lang.Object **visitOpTree**(org.openmrs.module.dssmodule.ast.AST **t**)

- **visitProgramTree**
  public abstract java.lang.Object **visitProgramTree**(org.openmrs.module.dssmodule.ast.
  t)

- **visitReturnTree**
  public abstract java.lang.Object **visitReturnTree**(org.openmrs.module.dssmodule.ast.A
  t)

- **visitWhileTree**
  public abstract java.lang.Object **visitWhileTree**(org.openmrs.module.dssmodule.ast.AS
  t)

Members inherited from class ASTVisitor org.openmrs.module.dssmodule.visitor.ASTVisito
(in C.9.1, page 150)

visitActualArgsTree, visitAssignTree, visitBlockTree, visitCallTree, visitElsifTree, visitFieldRefTree, visitFormalsTree, visitForTree, visitFunctionDeclTree, visitIdTree, visitIfTree, visitKids, visitListTree, visitLiteralTree, visitObjectDeclTree, visitObjectTree, visitOpTree, visitProgramTree, visitReturnTree, visitWhileTree

C.8.12 Class ListInterpreter

Handles interpretation of List literals

Declaration public class ListInterpreter
extends java.lang.Object
implements ASTInterpreter

Constructor summary

ListInterpreter()

Method summary

interpret(ListTree, ExecutionContext, ASTVisitor)

Constructors

- **ListInterpreter**
  public **ListInterpreter**()

Methods

- **interpret**
  public java.lang.Object **interpret**(org.openmrs.module.dssmodule.ast.ListTree **tree**, org.openmrs.module.dssmodule.state.ExecutionContext **context**, org.openmrs.module.dssmodule.visitor.ASTVisitor **visitor**)

C.8.13 Class LiteralInterpreter

Handles interpretation of Literal nodes

Declaration public class LiteralInterpreter
extends java.lang.Object
implements ASTInterpreter

Constructor summary

LiteralInterpreter()

Method summary

interpret(LiteralTree, ExecutionContext, ASTVisitor)

Constructors

- **LiteralInterpreter**
  `public` **LiteralInterpreter**`()`

Methods

- **interpret**
  `public java.lang.Object` **interpret**`(org.openmrs.module.dssmodule.ast.LiteralTree` **tree**`, org.openmrs.module.dssmodule.state.ExecutionContext` **context**`, org.openmrs.module.dssmodule.visitor.ASTVisitor` **visitor**`)`

C.8.14 Class ObjectDeclInterpreter

Handles interpretation of object declarations.

Declaration public class ObjectDeclInterpreter
extends java.lang.Object
implements ASTInterpreter

Constructor summary

ObjectDeclInterpreter()

Method summary

interpret(**ObjectDeclTree**, **ExecutionContext**, **ASTVisitor**)

Constructors

- **ObjectDeclInterpreter**
  `public` **ObjectDeclInterpreter**`()`

Methods

- **interpret**
  `public java.lang.Object` **interpret**`(org.openmrs.module.dssmodule.ast.ObjectDeclTree` **tree**`, org.openmrs.module.dssmodule.state.ExecutionContext` **context**`, org.openmrs.module.dssmodule.visitor.ASTVisitor` **visitor**`)`

C.8.15 Class ObjectInterpreter

Handles interpretation of object sub-trees, used in DSS1 to instantiate new objects based on prototypes.

Declaration public class ObjectInterpreter
extends java.lang.Object
implements ASTInterpreter

Constructor summary

ObjectInterpreter()

Method summary

interpret(**ObjectTree**, **ExecutionContext**, **ASTVisitor**)

Constructors

- **ObjectInterpreter**
  `public` **ObjectInterpreter**`()`

Methods

- **interpret**
  `public java.lang.Object` **interpret**`(org.openmrs.module.dssmodule.ast.ObjectTree` **tree**`, org.openmrs.module.dssmodule.state.ExecutionContext` **context**`, org.openmrs.module.dssmodule.visitor.ASTVisitor` **visitor**`)`

C.8.16 Class OpInterpreter

Handles operations in DSS1 (arithmetic, logical, et cetera). Note that specific semantics are deferred to the evaluator.

Declaration public class OpInterpreter
extends java.lang.Object
implements ASTInterpreter

Constructor summary

OpInterpreter()

Method summary

interpret(OpTree, ExecutionContext, ASTVisitor)

Constructors

- **OpInterpreter**
  `public` **OpInterpreter**`()`

Methods

- **interpret**
  `public java.lang.Object` **interpret**`(org.openmrs.module.dssmodule.ast.OpTree` **tree**`, org.openmrs.module.dssmodule.state.ExecutionContext` **context**`, org.openmrs.module.dssmodule.visitor.ASTVisitor` **visitor**`)`

C.8.17 Class ProgramInterpreter

Interpret a program tree. This is typically the top-level node of an AST.

Declaration public class ProgramInterpreter
extends java.lang.Object
implements ASTInterpreter

Constructor summary

ProgramInterpreter()

Method summary

interpret(ProgramTree, ExecutionContext, ASTVisitor)

Constructors

- **ProgramInterpreter**
  `public` **ProgramInterpreter**`()`

Methods

- **interpret**
  `public java.lang.Object` **interpret**`(org.openmrs.module.dssmodule.ast.ProgramTree` **tree**`, org.openmrs.module.dssmodule.state.ExecutionContext` **context**`, org.openmrs.module.dssmodule.visitor.ASTVisitor` **visitor**`)`

C.8.18 Class ReturnInterpreter

Handle interpretation of return statements in a DSS1 program.

Declaration public class ReturnInterpreter
extends java.lang.Object
implements ASTInterpreter

Constructor summary

ReturnInterpreter()

Method summary

interpret(ReturnTree, ExecutionContext, ASTVisitor)

Constructors

- **ReturnInterpreter**
  `public` **ReturnInterpreter**`()`

Methods

- **interpret**
  `public java.lang.Object` **interpret**`(org.openmrs.module.dssmodule.ast.ReturnTree` **tree**`, org.openmrs.module.dssmodule.state.ExecutionContext` **context**`, org.openmrs.module.dssmodule.visitor.ASTVisitor` **visitor**`)`

C.8.19 Class WhileInterpreter

Handles interpretation of while loops in a DSS1 program.

Declaration public class WhileInterpreter
extends java.lang.Object
implements ASTInterpreter

Constructor summary

WhileInterpreter()

Method summary

interpret(WhileTree, ExecutionContext, ASTVisitor)

Constructors

- **WhileInterpreter**
  `public` **WhileInterpreter**`()`

Methods

- **interpret**
  `public java.lang.Object` **interpret**`(org.openmrs.module.dssmodule.ast.WhileTree` **tree**`, org.openmrs.module.dssmodule.state.ExecutionContext` **context**`, org.openmrs.module.dssmodule.visitor.ASTVisitor` **visitor**`)`

C.9 Package org.openmrs.module.dssmodule.visitor

Package Contents *Page*

Classes

PrintVisitor is used to visit an AST and print it using appropriate indentation:

C.9.1 Class ASTVisitor

ASTVisitor class is the root of the Visitor hierarchy for visiting various AST's; each visitor asks each node in the AST it is given to *accept* its visit;
each subclass **must** provide all of the visitors mentioned in this class;
after visiting a tree the visitor can return any Object of interest
e.g. when the constrainer visits an expression tree it will return a reference to the type tree representing the type of the expression

Declaration public abstract class ASTVisitor
extends java.lang.Object

All known subclasses InterpreterVisitor (in C.8.11, page 141), PrintVisitor (in C.9.2, page 153)

Constructor summary

ASTVisitor()

Method summary

visitActualArgsTree(AST)
visitAssignTree(AST)
visitBlockTree(AST)
visitCallTree(AST)
visitElsifTree(AST)
visitFieldRefTree(AST)
visitFormalsTree(AST)
visitForTree(AST)
visitFunctionDeclTree(AST)
visitIdTree(AST)
visitIfTree(AST)

visitKids(AST)
visitListTree(AST)
visitLiteralTree(AST)
visitObjectDeclTree(AST)
visitObjectTree(AST)
visitOpTree(AST)
visitProgramTree(AST)
visitReturnTree(AST)
visitWhileTree(AST)

Constructors

- **ASTVisitor**
  `public` **ASTVisitor**`()`

Methods

- **visitActualArgsTree**
  `public abstract java.lang.Object` **visitActualArgsTree**`(org.openmrs.module.dssmodule.a`
  t`)`

- **visitAssignTree**
  `public abstract java.lang.Object` **visitAssignTree**`(org.openmrs.module.dssmodule.ast.AS`
  t`)`

- **visitBlockTree**
  `public abstract java.lang.Object` **visitBlockTree**`(org.openmrs.module.dssmodule.ast.AST`
  t`)`

- **visitCallTree**
  `public abstract java.lang.Object` **visitCallTree**`(org.openmrs.module.dssmodule.ast.AST`
  t`)`

- **visitElsifTree**
  `public abstract java.lang.Object` **visitElsifTree**`(org.openmrs.module.dssmodule.ast.AST`
  t`)`

- **visitFieldRefTree**
  `public abstract java.lang.Object` **visitFieldRefTree**`(org.openmrs.module.dssmodule.ast.`
  t`)`

- **visitFormalsTree**
  public abstract java.lang.Object **visitFormalsTree**(org.openmrs.module.dssmodule.ast.A
  t)

- **visitForTree**
  public abstract java.lang.Object **visitForTree**(org.openmrs.module.dssmodule.ast.AST
  t)

- **visitFunctionDeclTree**
  public abstract java.lang.Object **visitFunctionDeclTree**(org.openmrs.module.dssmodule.ast.AST **t**)

- **visitIdTree**
  public abstract java.lang.Object **visitIdTree**(org.openmrs.module.dssmodule.ast.AST
  t)

- **visitIfTree**
  public abstract java.lang.Object **visitIfTree**(org.openmrs.module.dssmodule.ast.AST
  t)

- **visitKids**
  public void **visitKids**(org.openmrs.module.dssmodule.ast.AST **t**)

- **visitListTree**
  public abstract java.lang.Object **visitListTree**(org.openmrs.module.dssmodule.ast.AST
  t)

- **visitLiteralTree**
  public abstract java.lang.Object **visitLiteralTree**(org.openmrs.module.dssmodule.ast.AS
  t)

- **visitObjectDeclTree**
  public abstract java.lang.Object **visitObjectDeclTree**(org.openmrs.module.dssmodule.ast.AST **t**)

- **visitObjectTree**
  public abstract java.lang.Object **visitObjectTree**(org.openmrs.module.dssmodule.ast.AS
  t)

- **visitOpTree**
  `public abstract java.lang.Object` **visitOpTree**`(org.openmrs.module.dssmodule.ast.AST` **t**`)`

- **visitProgramTree**
  `public abstract java.lang.Object` **visitProgramTree**`(org.openmrs.module.dssmodule.ast.` **t**`)`

- **visitReturnTree**
  `public abstract java.lang.Object` **visitReturnTree**`(org.openmrs.module.dssmodule.ast.A` **t**`)`

- **visitWhileTree**
  `public abstract java.lang.Object` **visitWhileTree**`(org.openmrs.module.dssmodule.ast.AS` **t**`)`

C.9.2 Class PrintVisitor

PrintVisitor is used to visit an AST and print it using appropriate indentation:

```
root
  Kid1
  Kid2
    Kid21
    Kid22
    Kid23
  Kid3
```

Declaration public class PrintVisitor
extends org.openmrs.module.dssmodule.visitor.ASTVisitor (in C.9.1, page 150)

Constructor summary

PrintVisitor()

Method summary

print(String, AST) Print the tree
visitActualArgsTree(AST)
visitAssignTree(AST)
visitBlockTree(AST)
visitCallTree(AST)
visitElsifTree(AST)
visitFieldRefTree(AST)
visitFormalsTree(AST)
visitForTree(AST)
visitFunctionDeclTree(AST)
visitIdTree(AST)
visitIfTree(AST)
visitListTree(AST)
visitLiteralTree(AST)
visitObjectDeclTree(AST)
visitObjectTree(AST)
visitOpTree(AST)
visitProgramTree(AST)
visitReturnTree(AST)
visitWhileTree(AST)

Constructors

- **PrintVisitor**
  `public` **PrintVisitor**`()`

Methods

- **print**
  `public void` **print**`(java.lang.String s, org.openmrs.module.dssmodule.ast.AST t)`
  - **Description**
    Print the tree
  - **Parameters**
    - `s` – is the String for the root of t
    - `t` – is the tree to print - print the information in the node at the root (e.g. decoration) and its kids indented appropriately

- **visitActualArgsTree**
  public abstract java.lang.Object **visitActualArgsTree**(org.openmrs.module.dssmodule.a
  t)

- **visitAssignTree**
  public abstract java.lang.Object **visitAssignTree**(org.openmrs.module.dssmodule.ast.AS
  t)

- **visitBlockTree**
  public abstract java.lang.Object **visitBlockTree**(org.openmrs.module.dssmodule.ast.AST
  t)

- **visitCallTree**
  public abstract java.lang.Object **visitCallTree**(org.openmrs.module.dssmodule.ast.AST
  t)

- **visitElsifTree**
  public abstract java.lang.Object **visitElsifTree**(org.openmrs.module.dssmodule.ast.AST
  t)

- **visitFieldRefTree**
  public abstract java.lang.Object **visitFieldRefTree**(org.openmrs.module.dssmodule.ast.
  t)

- **visitFormalsTree**
  public abstract java.lang.Object **visitFormalsTree**(org.openmrs.module.dssmodule.ast.
  t)

- **visitForTree**
  public abstract java.lang.Object **visitForTree**(org.openmrs.module.dssmodule.ast.AST
  t)

- **visitFunctionDeclTree**
  public abstract java.lang.Object **visitFunctionDeclTree**(org.openmrs.module.dssmodule.ast.AST t)

- **visitIdTree**
  public abstract java.lang.Object **visitIdTree**(org.openmrs.module.dssmodule.ast.AST
  t)

- **visitIfTree**
  public abstract java.lang.Object **visitIfTree**(org.openmrs.module.dssmodule.ast.AST t)

- **visitListTree**
  public abstract java.lang.Object **visitListTree**(org.openmrs.module.dssmodule.ast.AST t)

- **visitLiteralTree**
  public abstract java.lang.Object **visitLiteralTree**(org.openmrs.module.dssmodule.ast.AS t)

- **visitObjectDeclTree**
  public abstract java.lang.Object **visitObjectDeclTree**(org.openmrs.module.dssmodule.ast.AST t)

- **visitObjectTree**
  public abstract java.lang.Object **visitObjectTree**(org.openmrs.module.dssmodule.ast.AS t)

- **visitOpTree**
  public abstract java.lang.Object **visitOpTree**(org.openmrs.module.dssmodule.ast.AST t)

- **visitProgramTree**
  public abstract java.lang.Object **visitProgramTree**(org.openmrs.module.dssmodule.ast t)

- **visitReturnTree**
  public abstract java.lang.Object **visitReturnTree**(org.openmrs.module.dssmodule.ast.A t)

- **visitWhileTree**
  public abstract java.lang.Object **visitWhileTree**(org.openmrs.module.dssmodule.ast.AS t)

Members inherited from class ASTVisitor org.openmrs.module.dssmodule.visitor.ASTVisitor
(in C.9.1, page 150)

visitActualArgsTree, visitAssignTree, visitBlockTree, visitCallTree, visitElsifTree, visitFieldRefTree, visitFormalsTree, visitForTree, visitFunctionDeclTree, visitIdTree, visitIfTree, visitKids, visitListTree, visitLiteralTree, visitObjectDeclTree, visitObjectTree, visitOpTree, visitProgramTree, visitReturnTree, visitWhileTree

C.10 Package org.openmrs.module.dssmodule.ast

Package Contents *Page*

Classes

C.10.1 Class ActualArgsTree

Declaration public class ActualArgsTree
extends org.openmrs.module.dssmodule.ast.AST (in C.10.3, page 160)

Constructor summary

ActualArgsTree()

Method summary

accept(ASTVisitor)

Constructors

- **ActualArgsTree**
  `public` **ActualArgsTree**`()`

Methods

- **accept**
  `public abstract java.lang.Object` **accept**`(org.openmrs.module.dssmodule.visitor.ASTVis` **v**`)`
  - **Description copied from AST** (**in C.10.3, page 160**)
    accept the visitor for this node - this method must be defined in each of the subclasses of AST
  - **Parameters**
    - `v` – is the ASTVisitor visiting this node (currently, a printer, constrainer and code generator)
  - **Returns** – the desired Object, as determined by the visitor

Members inherited from class AST `org.openmrs.module.dssmodule.ast.AST` (in C.10.3, page 160)

accept, addKid, getDecoration, getKid, getKids, getLabel, getNodeNum, kidCount, setDecoration, setLabel

C.10.2 Class AssignTree

Declaration public class AssignTree
extends org.openmrs.module.dssmodule.ast.AST (in C.10.3, page 160)

Constructor summary

AssignTree()

Method summary

accept(ASTVisitor)

Constructors

- **AssignTree**
  `public` **AssignTree**`()`

Methods

- **accept**
  `public abstract java.lang.Object` **accept**`(org.openmrs.module.dssmodule.visitor.ASTVis` **v**`)`
  - **Description copied from AST (in C.10.3, page 160)**
    accept the visitor for this node - this method must be defined in each of the subclasses of AST
  - **Parameters**
    * `v` – is the ASTVisitor visiting this node (currently, a printer, constrainer and code generator)
  - **Returns** – the desired Object, as determined by the visitor

Members inherited from class AST `org.openmrs.module.dssmodule.ast.AST` (in C.10.3, page 160)
accept, addKid, getDecoration, getKid, getKids, getLabel, getNodeNum, kidCount, setDecoration, setLabel

C.10.3 Class AST

The AST Abstract class is the Abstract Syntax Tree representation; each node contains

1. references to its kids,
2. its unique node number used for printing/debugging,
3. its decoration used for constraining and code generation, and
4. a label for code generation

The AST is built by the Parser

Declaration public abstract class AST
extends java.lang.Object

All known subclasses FieldRefTree (in C.10.7, page 165), BlockTree (in C.10.4, page 163), CallTree (in C.10.5, page 163), ActualArgsTree (in C.10.1, page 158), FunctionDeclTree (in C.10.10, page 168), ForTree (in C.10.9, page 167), WhileTree (in C.10.20, page 178), ElsifTree (in C.10.6, page 164), LiteralTree (in C.10.14, page 172), IdTree (in C.10.11, page 169), ReturnTree (in C.10.19, page 177), IfTree (in C.10.12, page 171), FormalsTree (in C.10.8, page 166), OpTree (in C.10.17, page 175), ObjectTree (in C.10.16, page 174), ObjectDeclTree (in C.10.15, page 173), ListTree (in C.10.13, page 171), ProgramTree (in C.10.18, page 176), AssignTree (in C.10.2, page 159)

Constructor summary

AST()

Method summary

accept(ASTVisitor) accept the visitor for this node - this method must be defined in each of the subclasses of AST
addKid(AST)
getDecoration()
getKid(int) get the AST corresponding to the kid
getKids()
getLabel()
getNodeNum()
kidCount()
setDecoration(AST)
setLabel(String)

Constructors

- **AST**

```
public AST()
```

Methods

- **accept**

```
public abstract java.lang.Object accept(org.openmrs.module.dssmodule.visitor.ASTVis
v)
```

  - **Description**

    accept the visitor for this node - this method must be defined in each of the subclasses of AST

  - **Parameters**

    - `v` – is the ASTVisitor visiting this node (currently, a printer, constrainer and code generator)
  - **Returns** – the desired Object, as determined by the visitor

- **addKid**
  `public AST` **addKid**`(AST` **kid**`)`

- **getDecoration**
  `public AST` **getDecoration**`()`

- **getKid**
  `public AST` **getKid**`(int` **i**`)`

  - **Description**
    get the AST corresponding to the kid
  - **Parameters**
    - `i` – is the number of the needed kid; it starts with kid number one
  - **Returns** – the AST for the indicated kid

- **getKids**
  `public java.util.ArrayList` **getKids**`()`

- **getLabel**
  `public java.lang.String` **getLabel**`()`

- **getNodeNum**
  `public int` **getNodeNum**`()`

- **kidCount**
  `public int` **kidCount**`()`

  - **Returns** – the number of kids at this node

- **setDecoration**
  `public void` **setDecoration**`(AST` **t**`)`

- **setLabel**
  `public void` **setLabel**`(java.lang.String` **label**`)`

C.10.4 Class BlockTree

Declaration public class BlockTree
extends org.openmrs.module.dssmodule.ast.AST (in C.10.3, page 160)

Constructor summary

BlockTree()

Method summary

accept(ASTVisitor)

Constructors

- **BlockTree**
  `public` **BlockTree**`()`

Methods

- **accept**
  `public abstract java.lang.Object` **accept**`(org.openmrs.module.dssmodule.visitor.ASTVis`
  `v)`
  - **Description copied from AST (in C.10.3, page 160)**
    accept the visitor for this node - this method must be defined in each of the subclasses of AST
  - **Parameters**
    - `v` – is the ASTVisitor visiting this node (currently, a printer, constrainer and code generator)
  - **Returns** – the desired Object, as determined by the visitor

Members inherited from class AST `org.openmrs.module.dssmodule.ast.AST` (in C.10.3, page 160)

accept, addKid, getDecoration, getKid, getKids, getLabel, getNodeNum, kidCount, setDecoration, setLabel

C.10.5 Class CallTree

Declaration public class CallTree
extends org.openmrs.module.dssmodule.ast.AST (in C.10.3, page 160)

Constructor summary

CallTree()

Method summary

accept(ASTVisitor)

Constructors

- **CallTree**
  `public` **CallTree**`()`

Methods

- **accept**
  `public abstract java.lang.Object` **accept**`(org.openmrs.module.dssmodule.visitor.ASTVis v)`
  - **Description copied from AST** (**in C.10.3, page 160**)
    accept the visitor for this node - this method must be defined in each of the subclasses of AST
  - **Parameters**
    - `v` – is the ASTVisitor visiting this node (currently, a printer, constrainer and code generator)
  - **Returns** – the desired Object, as determined by the visitor

Members inherited from class AST `org.openmrs.module.dssmodule.ast.AST` (in C.10.3, page 160)

accept, addKid, getDecoration, getKid, getKids, getLabel, getNodeNum, kidCount, setDecoration, setLabel

C.10.6 Class ElsifTree

Declaration public class ElsifTree
extends org.openmrs.module.dssmodule.ast.AST (in C.10.3, page 160)

Constructor summary

ElsifTree()

Method summary

accept(ASTVisitor)

Constructors

- **ElsifTree**
  `public` **ElsifTree**`()`

Methods

- **accept**
  `public abstract java.lang.Object` **accept**`(org.openmrs.module.dssmodule.visitor.ASTVis`
  v`)`
  - **Description copied from AST** (**in C.10.3, page 160**)
    accept the visitor for this node - this method must be defined in each of the subclasses of AST
  - **Parameters**
    - `v` – is the ASTVisitor visiting this node (currently, a printer, constrainer and code generator)
  - **Returns** – the desired Object, as determined by the visitor

Members inherited from class AST `org.openmrs.module.dssmodule.ast.AST` (in C.10.3, page 160)

accept, addKid, getDecoration, getKid, getKids, getLabel, getNodeNum, kidCount, setDecoration, setLabel

C.10.7 Class FieldRefTree

Declaration public class FieldRefTree
extends org.openmrs.module.dssmodule.ast.AST (in C.10.3, page 160)

Constructor summary

FieldRefTree()

Method summary

accept(ASTVisitor)

Constructors

- **FieldRefTree**
  `public` **FieldRefTree**`()`

Methods

- **accept**
  `public abstract java.lang.Object` **accept**`(org.openmrs.module.dssmodule.visitor.ASTVis`
  `v)`
  - **Description copied from AST (in C.10.3, page 160)**
    accept the visitor for this node - this method must be defined in each of the subclasses of AST
  - **Parameters**
    - `v` – is the ASTVisitor visiting this node (currently, a printer, constrainer and code generator)
  - **Returns** – the desired Object, as determined by the visitor

Members inherited from class AST `org.openmrs.module.dssmodule.ast.AST` (in C.10.3, page 160)

accept, addKid, getDecoration, getKid, getKids, getLabel, getNodeNum, kidCount, setDecoration, setLabel

C.10.8 Class FormalsTree

Declaration public class FormalsTree
extends org.openmrs.module.dssmodule.ast.AST (in C.10.3, page 160)

Constructor summary

FormalsTree()

Method summary

accept(ASTVisitor)

Constructors

- **FormalsTree**
  `public` **FormalsTree**`()`

Methods

- **accept**
  `public abstract java.lang.Object` **accept**`(org.openmrs.module.dssmodule.visitor.ASTVis`
  v`)`
  - **Description copied from AST** (**in C.10.3, page 160**)
    accept the visitor for this node - this method must be defined in each of the subclasses of AST
  - **Parameters**
    * `v` – is the ASTVisitor visiting this node (currently, a printer, constrainer and code generator)
  - **Returns** – the desired Object, as determined by the visitor

Members inherited from class AST `org.openmrs.module.dssmodule.ast.AST` (in C.10.3, page 160)

accept, addKid, getDecoration, getKid, getKids, getLabel, getNodeNum, kidCount, setDecoration, setLabel

C.10.9 Class ForTree

Declaration public class ForTree
extends org.openmrs.module.dssmodule.ast.AST (in C.10.3, page 160)

Constructor summary

ForTree()

Method summary

accept(ASTVisitor)

Constructors

- **ForTree**
  `public` **ForTree**`()`

Methods

- **accept**
  `public abstract java.lang.Object` **accept**`(org.openmrs.module.dssmodule.visitor.ASTVis`
  v`)`
  - **Description copied from AST** (**in C.10.3, page 160**)
    accept the visitor for this node - this method must be defined in each of the subclasses of AST
  - **Parameters**
    - `v` – is the ASTVisitor visiting this node (currently, a printer, constrainer and code generator)
  - **Returns** – the desired Object, as determined by the visitor

Members inherited from class AST `org.openmrs.module.dssmodule.ast.AST` (in C.10.3, page 160)

accept, addKid, getDecoration, getKid, getKids, getLabel, getNodeNum, kidCount, setDecoration, setLabel

C.10.10 Class FunctionDeclTree

Declaration public class FunctionDeclTree
extends org.openmrs.module.dssmodule.ast.AST (in C.10.3, page 160)

Constructor summary

FunctionDeclTree()

Method summary

accept(ASTVisitor)

Constructors

- **FunctionDeclTree**
  `public` **FunctionDeclTree**`()`

Methods

- **accept**
  public abstract java.lang.Object **accept**(org.openmrs.module.dssmodule.visitor.ASTVis
  v)
  - **Description copied from AST (in C.10.3, page 160)**
    accept the visitor for this node - this method must be defined in each of the subclasses of AST
  - **Parameters**
    - v – is the ASTVisitor visiting this node (currently, a printer, constrainer and code generator)
  - **Returns** – the desired Object, as determined by the visitor

Members inherited from class AST `org.openmrs.module.dssmodule.ast.AST` (in C.10.3, page 160)
accept, addKid, getDecoration, getKid, getKids, getLabel, getNodeNum, kidCount, setDecoration, setLabel

C.10.11 Class IdTree

Declaration public class IdTree
extends org.openmrs.module.dssmodule.ast.AST (in C.10.3, page 160)

Constructor summary

IdTree(Token)

Method summary

accept(ASTVisitor)
getFrameOffset()
getSymbol()
setFrameOffset(int)

Constructors

- **IdTree**
  public **IdTree**(org.openmrs.module.dssmodule.lexer.Token **tok**)
  - **Parameters**
    - tok – - record the symbol from the token Symbol

Methods

- **accept**
  public abstract java.lang.Object **accept**(org.openmrs.module.dssmodule.visitor.ASTVis **v**)
  - **Description copied from AST** (**in C.10.3, page 160**)
    accept the visitor for this node - this method must be defined in each of the subclasses of AST
  - **Parameters**
    - v – is the ASTVisitor visiting this node (currently, a printer, constrainer and code generator)
  - **Returns** – the desired Object, as determined by the visitor
- **getFrameOffset**
  public int **getFrameOffset**()
  - **Returns** – the frame offset for this variable - used by codegen
- **getSymbol**
  public org.openmrs.module.dssmodule.lexer.Symbol **getSymbol**()
- **setFrameOffset**
  public void **setFrameOffset**(int **i**)
  - **Parameters**
    - i – is the offset for this variable as determined by the code generator

Members inherited from class AST org.openmrs.module.dssmodule.ast.AST (in C.10.3, page 160)

accept, addKid, getDecoration, getKid, getKids, getLabel, getNodeNum, kidCount, setDecoration, setLabel

C.10.12 Class IfTree

Declaration public class IfTree
extends org.openmrs.module.dssmodule.ast.AST (in C.10.3, page 160)

Constructor summary

IfTree()

Method summary

accept(ASTVisitor)

Constructors

- **IfTree**
  `public` **IfTree**`()`

Methods

- **accept**
  `public abstract java.lang.Object` **accept**`(org.openmrs.module.dssmodule.visitor.ASTVis`
  `v)`
  - **Description copied from AST (in C.10.3, page 160)**
    accept the visitor for this node - this method must be defined in each of the subclasses of AST
  - **Parameters**
    - `v` – is the ASTVisitor visiting this node (currently, a printer, constrainer and code generator)
  - **Returns** – the desired Object, as determined by the visitor

Members inherited from class AST `org.openmrs.module.dssmodule.ast.AST` (in C.10.3, page 160)
accept, addKid, getDecoration, getKid, getKids, getLabel, getNodeNum, kidCount, setDecoration, setLabel

C.10.13 Class ListTree

Declaration public class ListTree
extends org.openmrs.module.dssmodule.ast.AST (in C.10.3, page 160)

Constructor summary

ListTree()

Method summary

accept(ASTVisitor)

Constructors

- **ListTree**
  `public` **ListTree()**

Methods

- **accept**
  `public abstract java.lang.Object` **accept**`(org.openmrs.module.dssmodule.visitor.ASTVis`
  v`)`
  - **Description copied from AST** (**in C.10.3, page 160**)
    accept the visitor for this node - this method must be defined in each of the subclasses of AST
  - **Parameters**
    - `v` – is the ASTVisitor visiting this node (currently, a printer, constrainer and code generator)
  - **Returns** – the desired Object, as determined by the visitor

Members inherited from class AST `org.openmrs.module.dssmodule.ast.AST` (in C.10.3, page 160)

accept, addKid, getDecoration, getKid, getKids, getLabel, getNodeNum, kidCount, setDecoration, setLabel

C.10.14 Class LiteralTree

Declaration public class LiteralTree
extends org.openmrs.module.dssmodule.ast.AST (in C.10.3, page 160)

Constructor summary

LiteralTree(Token)

Method summary

accept(ASTVisitor)
getSymbol()

Constructors

- **LiteralTree**
  `public` **LiteralTree**`(org.openmrs.module.dssmodule.lexer.Token` **tok**`)`
  - **Parameters**
    - `tok` – is the Token containing the String representation of the integer literal; we keep the String rather than converting to an integer value so we don't introduce any machine dependencies with respect to integer representations

Methods

- **accept**
  `public abstract java.lang.Object` **accept**`(org.openmrs.module.dssmodule.visitor.ASTVis`
  v`)`
  - **Description copied from AST (in C.10.3, page 160)**
    accept the visitor for this node - this method must be defined in each of the subclasses of AST
  - **Parameters**
    - `v` – is the ASTVisitor visiting this node (currently, a printer, constrainer and code generator)
  - **Returns** – the desired Object, as determined by the visitor
- **getSymbol**
  `public org.openmrs.module.dssmodule.lexer.Symbol` **getSymbol**`()`

Members inherited from class AST `org.openmrs.module.dssmodule.ast.AST` (in C.10.3, page 160)

accept, addKid, getDecoration, getKid, getKids, getLabel, getNodeNum, kidCount, setDecoration, setLabel

C.10.15 Class ObjectDeclTree

Declaration public class ObjectDeclTree
extends org.openmrs.module.dssmodule.ast.AST (in C.10.3, page 160)

Constructor summary

ObjectDeclTree()

Method summary

accept(ASTVisitor)

Constructors

- **ObjectDeclTree**
  `public` **ObjectDeclTree**`()`

Methods

- **accept**
  `public abstract java.lang.Object` **accept**`(org.openmrs.module.dssmodule.visitor.ASTVis`
  `v)`
  - **Description copied from AST (in C.10.3, page 160)**
    accept the visitor for this node - this method must be defined in each of the subclasses of AST
  - **Parameters**
    * `v` – is the ASTVisitor visiting this node (currently, a printer, constrainer and code generator)
  - **Returns** – the desired Object, as determined by the visitor

Members inherited from class AST `org.openmrs.module.dssmodule.ast.AST` (in C.10.3, page 160)
accept, addKid, getDecoration, getKid, getKids, getLabel, getNodeNum, kidCount, setDecoration, setLabel

C.10.16 Class ObjectTree

Declaration public class ObjectTree
extends org.openmrs.module.dssmodule.ast.AST (in C.10.3, page 160)

Constructor summary

ObjectTree()

Method summary

accept(ASTVisitor)

Constructors

- **ObjectTree**
  `public` **ObjectTree**`()`

Methods

- **accept**
  `public abstract java.lang.Object` **accept**`(org.openmrs.module.dssmodule.visitor.ASTVis`
  `v)`

  – **Description copied from AST** (**in C.10.3, page 160**)
  accept the visitor for this node - this method must be defined in each of the subclasses of AST

  – **Parameters**

    ∗ `v` – is the ASTVisitor visiting this node (currently, a printer, constrainer and code generator)

  – **Returns** – the desired Object, as determined by the visitor

Members inherited from class AST `org.openmrs.module.dssmodule.ast.AST` (in C.10.3, page 160)

accept, addKid, getDecoration, getKid, getKids, getLabel, getNodeNum, kidCount, setDecoration, setLabel

C.10.17 Class OpTree

Declaration public class OpTree
extends org.openmrs.module.dssmodule.ast.AST (in C.10.3, page 160)

Constructor summary

OpTree(Token)

Method summary

accept(ASTVisitor)
getSymbol()

Constructors

- **OpTree**
  `public OpTree(org.openmrs.module.dssmodule.lexer.Token tok)`
  - **Parameters**
    - `tok` – contains the Symbol which indicates the specific relational operator

Methods

- **accept**
  `public abstract java.lang.Object accept(org.openmrs.module.dssmodule.visitor.ASTVis`
  `v)`
  - **Description copied from AST (in C.10.3, page 160)**

    accept the visitor for this node - this method must be defined in each of the subclasses of AST
  - **Parameters**
    - `v` – is the ASTVisitor visiting this node (currently, a printer, constrainer and code generator)
  - **Returns** – the desired Object, as determined by the visitor
- **getSymbol**
  `public org.openmrs.module.dssmodule.lexer.Symbol getSymbol()`

Members inherited from class AST `org.openmrs.module.dssmodule.ast.AST` (in C.10.3, page 160)

accept, addKid, getDecoration, getKid, getKids, getLabel, getNodeNum, kidCount, setDecoration, setLabel

C.10.18 Class ProgramTree

Declaration public class ProgramTree
extends org.openmrs.module.dssmodule.ast.AST (in C.10.3, page 160)

Constructor summary

ProgramTree()

Method summary

accept(ASTVisitor)

Constructors

- **ProgramTree**
  `public` **ProgramTree**`()`

Methods

- **accept**
  `public abstract java.lang.Object` **accept**`(org.openmrs.module.dssmodule.visitor.ASTVis`
  v`)`
  - **Description copied from AST** (**in C.10.3, page 160**)
    accept the visitor for this node - this method must be defined in each of the subclasses of AST
  - **Parameters**
    - `v` – is the ASTVisitor visiting this node (currently, a printer, constrainer and code generator)
  - **Returns** – the desired Object, as determined by the visitor

Members inherited from class AST `org.openmrs.module.dssmodule.ast.AST` (in C.10.3, page 160)

accept, addKid, getDecoration, getKid, getKids, getLabel, getNodeNum, kidCount, setDecoration, setLabel

C.10.19 Class ReturnTree

Declaration public class ReturnTree
extends org.openmrs.module.dssmodule.ast.AST (in C.10.3, page 160)

Constructor summary

ReturnTree()

Method summary

accept(ASTVisitor)

Constructors

- **ReturnTree**
  `public` **ReturnTree**`()`

Methods

- **accept**
  `public abstract java.lang.Object` **accept**`(org.openmrs.module.dssmodule.visitor.ASTVis`
  `v``)`
  - **Description copied from AST** (**in C.10.3, page 160**)
    accept the visitor for this node - this method must be defined in each of the subclasses of AST
  - **Parameters**
    - `v` – is the ASTVisitor visiting this node (currently, a printer, constrainer and code generator)
  - **Returns** – the desired Object, as determined by the visitor

Members inherited from class AST `org.openmrs.module.dssmodule.ast.AST` (in C.10.3, page 160)

accept, addKid, getDecoration, getKid, getKids, getLabel, getNodeNum, kidCount, setDecoration, setLabel

C.10.20 Class WhileTree

Declaration public class WhileTree
extends org.openmrs.module.dssmodule.ast.AST (in C.10.3, page 160)

Constructor summary

WhileTree()

Method summary

accept(ASTVisitor)

Constructors

- **WhileTree**
  `public` **WhileTree**`()`

Methods

- **accept**
  public abstract java.lang.Object **accept**(org.openmrs.module.dssmodule.visitor.ASTVis **v**)
  - **Description copied from AST** (**in C.10.3, page 160**)
    accept the visitor for this node - this method must be defined in each of the subclasses of AST
  - **Parameters**
    * v – is the ASTVisitor visiting this node (currently, a printer, constrainer and code generator)
  - **Returns** – the desired Object, as determined by the visitor

Members inherited from class AST org.openmrs.module.dssmodule.ast.AST (in C.10.3, page 160)

accept, addKid, getDecoration, getKid, getKids, getLabel, getNodeNum, kidCount, setDecoration, setLabel

C.11 Package org.openmrs.module.dssmodule.value

Package Contents *Page*

Classes

C.11.1 Class DSSValue

Declaration public abstract class DSSValue
extends java.lang.Object
implements java.util.Comparator, java.lang.Comparable

All known subclasses DSSValueFloat (in C.11.5, page 189), DSSValueNull (in C.11.8, page 196), DSSValueObject (in C.11.9, page 197), DSSValueBool (in C.11.2, page 184), DSSValueString (in C.11.10, page 200), DSSValueInt (in C.11.6, page 191), DSSValueDate (in C.11.3, page 186), DSSValueList (in C.11.7, page 194)

Constructor summary

DSSValue()

Method summary

add(DSSValue)
and(DSSValue)
compare(DSSValue, DSSValue)
compareTo(DSSValue)
concat(DSSValue)
div(DSSValue)
equal(DSSValue)
getDSSValueTimeStamp()
getTimeStamp()

greaterthan(DSSValue)
greaterthanequal(DSSValue)
isBoolean()
isDate()
isFloat()
isInt()
isList()
isNull()
isNumeric()
isObject()
isString()
length()
lessthan(DSSValue)
lessthanequal(DSSValue)
mult(DSSValue)
not(DSSValue)
notequal(DSSValue)
or(DSSValue)
power(DSSValue)
setTimeStamp(Date)
setTimeStamp(DSSValueDate)
setTimeStamp(Long)
sort()
sub(DSSValue)
toFloat()
toInt()
toLong()

Constructors

- **DSSValue**
  `public` **DSSValue**`()`

Methods

- **add**
  `public DSSValue` **add**`(DSSValue b)`

- **and**
  `public DSSValue` **and**`(DSSValue b)`

- **compare**
  `public int` **compare**`(DSSValue` **t**`, DSSValue` **t1**`)`

- **compareTo**
  `public int` **compareTo**`(DSSValue` **t**`)`

- **concat**
  `public DSSValue` **concat**`(DSSValue` **b**`)`

- **div**
  `public DSSValue` **div**`(DSSValue` **b**`)`

- **equal**
  `public abstract boolean` **equal**`(DSSValue` **b**`)`

- **getDSSValueTimeStamp**
  `public DSSValue` **getDSSValueTimeStamp**`()`

- **getTimeStamp**
  `public java.util.Date` **getTimeStamp**`()`

- **greaterthan**
  `public abstract boolean` **greaterthan**`(DSSValue` **b**`)`

- **greaterthanequal**
  `public abstract boolean` **greaterthanequal**`(DSSValue` **b**`)`

- **isBoolean**
  `public boolean` **isBoolean**`()`

- **isDate**
  `public boolean` **isDate**`()`

- **isFloat**
  `public boolean` **isFloat**`()`

- **isInt**
  `public boolean` **isInt**`()`

- **isList**
  `public boolean` **isList**`()`

- **isNull**
  `public boolean` **isNull**`()`
- **isNumeric**
  `public boolean` **isNumeric**`()`
- **isObject**
  `public boolean` **isObject**`()`
- **isString**
  `public boolean` **isString**`()`
- **length**
  `public int` **length**`()`
- **lessthan**
  `public abstract boolean` **lessthan**`(DSSValue` **b**`)`
- **lessthanequal**
  `public abstract boolean` **lessthanequal**`(DSSValue` **b**`)`
- **mult**
  `public DSSValue` **mult**`(DSSValue` **b**`)`
- **not**
  `public DSSValue` **not**`(DSSValue` **b**`)`
- **notequal**
  `public abstract boolean` **notequal**`(DSSValue` **b**`)`
- **or**
  `public DSSValue` **or**`(DSSValue` **b**`)`
- **power**
  `public DSSValue` **power**`(DSSValue` **b**`)`
- **setTimeStamp**
  `public void` **setTimeStamp**`(java.util.Date` **d**`)`
- **setTimeStamp**
  `public void` **setTimeStamp**`(DSSValueDate` **d**`)`

- **setTimeStamp**
  `public void` **setTimeStamp**`(java.lang.Long` **d**`)`

- **sort**
  `public DSSValue` **sort**`()`

- **sub**
  `public DSSValue` **sub**`(DSSValue` **b**`)`

- **toFloat**
  `public abstract double` **toFloat**`()`

- **toInt**
  `public abstract int` **toInt**`()`

- **toLong**
  `public abstract long` **toLong**`()`

C.11.2 Class DSSValueBool

Declaration public class DSSValueBool
extends org.openmrs.module.dssmodule.value.DSSValue (in C.11.1, page 180)

Method summary

and(DSSValue)
equal(DSSValue)
greaterthan(DSSValue)
greaterthanequal(DSSValue)
isBoolean()
lessthan(DSSValue)
lessthanequal(DSSValue)
not(DSSValue)
notequal(DSSValue)
or(DSSValue)
toFloat()
toInt()
toLong()
toString()

Methods

- **and**
  `public DSSValue` **and**`(DSSValue` **b**`)`
- **equal**
  `public abstract boolean` **equal**`(DSSValue` **b**`)`
- **greaterthan**
  `public abstract boolean` **greaterthan**`(DSSValue` **b**`)`
- **greaterthanequal**
  `public abstract boolean` **greaterthanequal**`(DSSValue` **b**`)`
- **isBoolean**
  `public boolean` **isBoolean**`()`
- **lessthan**
  `public abstract boolean` **lessthan**`(DSSValue` **b**`)`
- **lessthanequal**
  `public abstract boolean` **lessthanequal**`(DSSValue` **b**`)`
- **not**
  `public DSSValue` **not**`(DSSValue` **b**`)`
- **notequal**
  `public abstract boolean` **notequal**`(DSSValue` **b**`)`
- **or**
  `public DSSValue` **or**`(DSSValue` **b**`)`
- **toFloat**
  `public abstract double` **toFloat**`()`
- **toInt**
  `public abstract int` **toInt**`()`
- **toLong**
  `public abstract long` **toLong**`()`
- **toString**
  `public java.lang.String` **toString**`()`

Members inherited from class DSSValue `org.openmrs.module.dssmodule.value.DSSValue` (in C.11.1, page 180)

add, and, compare, compareTo, concat, div, equal, getDSSValueTimeStamp, getTimeStamp, greaterthan, greaterthanequal, isBoolean, isDate, isFloat, isInt, isList, isNull, isNumeric, isObject, isString, length, lessthan, lessthanequal, mult, not, notequal, or, power, setTimeStamp, setTimeStamp, setTimeStamp, sort, sub, toFloat, toInt, toLong

C.11.3 Class DSSValueDate

Declaration public class DSSValueDate
extends org.openmrs.module.dssmodule.value.DSSValue (in C.11.1, page 180)

Constructor summary

DSSValueDate(Date)

Method summary

add(DSSValue)
equal(DSSValue)
greaterthan(DSSValue)
greaterthanequal(DSSValue)
isDate()
lessthan(DSSValue)
lessthanequal(DSSValue)
notequal(DSSValue)
sub(DSSValue)
toFloat()
toInt()
toLong()
toString()

Constructors

- **DSSValueDate**
  `public` **DSSValueDate**`(java.util.Date x)`

Methods

- **add**
  public DSSValue **add**(DSSValue **b**)

- **equal**
  public abstract boolean **equal**(DSSValue **b**)

- **greaterthan**
  public abstract boolean **greaterthan**(DSSValue **b**)

- **greaterthanequal**
  public abstract boolean **greaterthanequal**(DSSValue **b**)

- **isDate**
  public boolean **isDate**()

- **lessthan**
  public abstract boolean **lessthan**(DSSValue **b**)

- **lessthanequal**
  public abstract boolean **lessthanequal**(DSSValue **b**)

- **notequal**
  public abstract boolean **notequal**(DSSValue **b**)

- **sub**
  public DSSValue **sub**(DSSValue **b**)

- **toFloat**
  public abstract double **toFloat**()

- **toInt**
  public abstract int **toInt**()

- **toLong**
  public abstract long **toLong**()

- **toString**
  public java.lang.String **toString**()

Members inherited from class DSSValue `org.openmrs.module.dssmodule.value.DSSValue` (in C.11.1, page 180)

add, and, compare, compareTo, concat, div, equal, getDSSValueTimeStamp, getTimeStamp, greaterthan, greaterthanequal, isBoolean, isDate, isFloat, isInt, isList, isNull, isNumeric, isObject, isString, length, lessthan, lessthanequal, mult, not, notequal, or, power, setTimeStamp, setTimeStamp, setTimeStamp, sort, sub, toFloat, toInt, toLong

C.11.4 Class DSSValueFactory

Declaration public class DSSValueFactory
extends java.lang.Object

Constructor summary

DSSValueFactory()

Method summary

getDSSValue()
getDSSValue(boolean)
getDSSValue(Date)
getDSSValue(double)
getDSSValue(float)
getDSSValue(int)
getDSSValue(long)
getDSSValue(Map)
getDSSValue(String)
getDSSValue(Vector)
getDSSValueList()

Constructors

- **DSSValueFactory**
  `public` **DSSValueFactory**`()`

Methods

- **getDSSValue**
  `public static DSSValue` **getDSSValue**`()`

- **getDSSValue**
  `public static DSSValue` **getDSSValue**`(boolean x)`

- **getDSSValue**
  `public static DSSValue` **getDSSValue**`(java.util.Date x)`
- **getDSSValue**
  `public static DSSValue` **getDSSValue**`(double x)`
- **getDSSValue**
  `public static DSSValue` **getDSSValue**`(float x)`
- **getDSSValue**
  `public static DSSValue` **getDSSValue**`(int x)`
- **getDSSValue**
  `public static DSSValue` **getDSSValue**`(long x)`
- **getDSSValue**
  `public static DSSValue` **getDSSValue**`(java.util.Map` **map**`)`
- **getDSSValue**
  `public static DSSValue` **getDSSValue**`(java.lang.String x)`
- **getDSSValue**
  `public static DSSValue` **getDSSValue**`(java.util.Vector x)`
- **getDSSValueList**
  `public static DSSValue` **getDSSValueList**`()`

C.11.5 Class DSSValueFloat

Declaration public class DSSValueFloat
extends org.openmrs.module.dssmodule.value.DSSValueNumeric

Method summary

add(DSSValue)
and(DSSValue)
div(DSSValue)
equal(DSSValue)
greaterthan(DSSValue)

greaterthanequal(DSSValue)
isFloat()
lessthan(DSSValue)
lessthanequal(DSSValue)
mult(DSSValue)
notequal(DSSValue)
or(DSSValue)
sub(DSSValue)
toFloat()
toInt()
toLong()
toString()

Methods

- **add**
  `public DSSValue` **add**`(DSSValue b)`
- **and**
  `public DSSValue` **and**`(DSSValue b)`
- **div**
  `public DSSValue` **div**`(DSSValue b)`
- **equal**
  `public abstract boolean` **equal**`(DSSValue b)`
- **greaterthan**
  `public abstract boolean` **greaterthan**`(DSSValue b)`
- **greaterthanequal**
  `public abstract boolean` **greaterthanequal**`(DSSValue b)`
- **isFloat**
  `public boolean` **isFloat**`()`
- **lessthan**
  `public abstract boolean` **lessthan**`(DSSValue b)`
- **lessthanequal**
  `public abstract boolean` **lessthanequal**`(DSSValue b)`

- **mult**
  `public DSSValue` **mult**`(DSSValue` **b**`)`

- **notequal**
  `public abstract boolean` **notequal**`(DSSValue` **b**`)`

- **or**
  `public DSSValue` **or**`(DSSValue` **b**`)`

- **sub**
  `public DSSValue` **sub**`(DSSValue` **b**`)`

- **toFloat**
  `public abstract double` **toFloat**`()`

- **toInt**
  `public abstract int` **toInt**`()`

- **toLong**
  `public abstract long` **toLong**`()`

- **toString**
  `public java.lang.String` **toString**`()`

Members inherited from class DSSValueNumeric
`org.openmrs.module.dssmodule.value.DSSValueNumeric`
concat, isNumeric, power

Members inherited from class DSSValue `org.openmrs.module.dssmodule.value.DSSValue` (in C.11.1, page 180)
add, and, compare, compareTo, concat, div, equal, getDSSValueTimeStamp, getTimeStamp, greaterthan, greaterthanequal, isBoolean, isDate, isFloat, isInt, isList, isNull, isNumeric, isObject, isString, length, lessthan, lessthanequal, mult, not, notequal, or, power, setTimeStamp, setTimeStamp, setTimeStamp, sort, sub, toFloat, toInt, toLong

C.11.6 Class DSSValueInt

Declaration public class DSSValueInt
extends org.openmrs.module.dssmodule.value.DSSValueNumeric

Method summary

add(DSSValue)
and(DSSValue)
div(DSSValue)
equal(DSSValue)
greaterthan(DSSValue)
greaterthanequal(DSSValue)
isInt()
lessthan(DSSValue)
lessthanequal(DSSValue)
mult(DSSValue)
notequal(DSSValue)
or(DSSValue)
sub(DSSValue)
toFloat()
toInt()
toLong()
toString()

Methods

- **add**
  `public DSSValue` **add**`(DSSValue b)`

- **and**
  `public DSSValue` **and**`(DSSValue b)`

- **div**
  `public DSSValue` **div**`(DSSValue b)`

- **equal**
  `public abstract boolean` **equal**`(DSSValue b)`

- **greaterthan**
  `public abstract boolean` **greaterthan**`(DSSValue b)`

- **greaterthanequal**
  `public abstract boolean` **greaterthanequal**`(DSSValue b)`

- **isInt**
  `public boolean` **isInt**`()`

- **lessthan**
  `public abstract boolean` **lessthan**`(DSSValue` **b**`)`

- **lessthanequal**
  `public abstract boolean` **lessthanequal**`(DSSValue` **b**`)`

- **mult**
  `public DSSValue` **mult**`(DSSValue` **b**`)`

- **notequal**
  `public abstract boolean` **notequal**`(DSSValue` **b**`)`

- **or**
  `public DSSValue` **or**`(DSSValue` **b**`)`

- **sub**
  `public DSSValue` **sub**`(DSSValue` **b**`)`

- **toFloat**
  `public abstract double` **toFloat**`()`

- **toInt**
  `public abstract int` **toInt**`()`

- **toLong**
  `public abstract long` **toLong**`()`

- **toString**
  `public java.lang.String` **toString**`()`

Members inherited from class DSSValueNumeric `org.openmrs.module.dssmodule.value.DSSValueNumeric`

concat, isNumeric, power

Members inherited from class DSSValue `org.openmrs.module.dssmodule.value.DSSValue` (in C.11.1, page 180)

add, and, compare, compareTo, concat, div, equal, getDSSValueTimeStamp, getTimeStamp, greaterthan, greaterthanequal, isBoolean, isDate, isFloat, isInt, isList, isNull, isNumeric, isObject, isString, length, lessthan, lessthanequal, mult, not, notequal, or, power, setTimeStamp, setTimeStamp, setTimeStamp, sort, sub, toFloat, toInt, toLong

C.11.7 Class DSSValueList

Declaration public class DSSValueList
extends org.openmrs.module.dssmodule.value.DSSValue (in C.11.1, page 180)

Method summary

- **add(DSSValue)**
- **clear()**
- **concat(DSSValue)**
- **equal(DSSValue)**
- **get(int)**
- **greaterthan(DSSValue)**
- **greaterthanequal(DSSValue)**
- **isList()**
- **length()**
- **lessthan(DSSValue)**
- **lessthanequal(DSSValue)**
- **notequal(DSSValue)**
- **sort()**
- **sub(DSSValue)**
- **toFloat()**
- **toInt()**
- **toLong()**
- **toString()**

Methods

- **add**
  `public DSSValue add(DSSValue b)`

- **clear**
  `public void clear()`

- **concat**
  `public DSSValue concat(DSSValue b)`

- **equal**
  `public abstract boolean equal(DSSValue b)`

- **get**
  `public DSSValue` **get**`(int` **i**`)`

- **greaterthan**
  `public abstract boolean` **greaterthan**`(DSSValue` **b**`)`

- **greaterthanequal**
  `public abstract boolean` **greaterthanequal**`(DSSValue` **b**`)`

- **isList**
  `public boolean` **isList**`()`

- **length**
  `public int` **length**`()`

- **lessthan**
  `public abstract boolean` **lessthan**`(DSSValue` **b**`)`

- **lessthanequal**
  `public abstract boolean` **lessthanequal**`(DSSValue` **b**`)`

- **notequal**
  `public abstract boolean` **notequal**`(DSSValue` **b**`)`

- **sort**
  `public DSSValue` **sort**`()`

- **sub**
  `public DSSValue` **sub**`(DSSValue` **b**`)`

- **toFloat**
  `public abstract double` **toFloat**`()`

- **toInt**
  `public abstract int` **toInt**`()`

- **toLong**
  `public abstract long` **toLong**`()`

- **toString**
  `public java.lang.String` **toString**`()`

Members inherited from class DSSValue `org.openmrs.module.dssmodule.value.DSSValue` (in C.11.1, page 180)

add, and, compare, compareTo, concat, div, equal, getDSSValueTimeStamp, getTimeStamp, greaterthan, greaterthanequal, isBoolean, isDate, isFloat, isInt, isList, isNull, isNumeric, isObject, isString, length, lessthan, lessthanequal, mult, not, notequal, or, power, setTimeStamp, setTimeStamp, setTimeStamp, sort, sub, toFloat, toInt, toLong

C.11.8 Class DSSValueNull

Declaration public class DSSValueNull
extends org.openmrs.module.dssmodule.value.DSSValue (in C.11.1, page 180)

Constructor summary

DSSValueNull()

Method summary

equal(DSSValue)
greaterthan(DSSValue)
greaterthanequal(DSSValue)
isNull()
lessthan(DSSValue)
lessthanequal(DSSValue)
notequal(DSSValue)
toFloat()
toInt()
toLong()
toString()

Constructors

- **DSSValueNull**
  `public` **DSSValueNull**`()`

Methods

- **equal**
  `public abstract boolean` **equal**`(DSSValue` **b**`)`

- **greaterthan**
  `public abstract boolean` **greaterthan**`(DSSValue` **b**`)`

- **greaterthanequal**
  `public abstract boolean` **greaterthanequal**`(DSSValue` **b**`)`

- **isNull**
  `public boolean` **isNull**`()`

- **lessthan**
  `public abstract boolean` **lessthan**`(DSSValue` **b**`)`

- **lessthanequal**
  `public abstract boolean` **lessthanequal**`(DSSValue` **b**`)`

- **notequal**
  `public abstract boolean` **notequal**`(DSSValue` **b**`)`

- **toFloat**
  `public abstract double` **toFloat**`()`

- **toInt**
  `public abstract int` **toInt**`()`

- **toLong**
  `public abstract long` **toLong**`()`

- **toString**
  `public java.lang.String` **toString**`()`

Members inherited from class DSSValue `org.openmrs.module.dssmodule.value.DSSValue` (in C.11.1, page 180)

add, and, compare, compareTo, concat, div, equal, getDSSValueTimeStamp, getTimeStamp, greaterthan, greaterthanequal, isBoolean, isDate, isFloat, isInt, isList, isNull, isNumeric, isObject, isString, length, lessthan, lessthanequal, mult, not, notequal, or, power, setTimeStamp, setTimeStamp, setTimeStamp, sort, sub, toFloat, toInt, toLong

C.11.9 Class DSSValueObject

Represents the "object" type in DSS1. Essentially operates as a struct (contains named fields which can be read or written).

Declaration public class DSSValueObject
extends org.openmrs.module.dssmodule.value.DSSValue (in C.11.1, page 180)
implements org.openmrs.module.dssmodule.state.NamingContext

Constructor summary

DSSValueObject()

Method summary

equal(DSSValue)
get(String)
greaterthan(DSSValue)
greaterthanequal(DSSValue)
lessthan(DSSValue)
lessthanequal(DSSValue)
names()
notequal(DSSValue)
set(String, DSSValue)
toFloat()
toInt()
toLong()

Constructors

- **DSSValueObject**
  `public` **DSSValueObject**`()`

Methods

- **equal**
  `public abstract boolean` **equal**`(DSSValue` **b**`)`

- **get**
  `DSSValue` **get**`(java.lang.String` **name**`)`

  - **Description copied from org.openmrs.module.dssmodule.state.NamingContext** (**in C.2.2, page 63**)

    Get the value currently associated with the specified name in this context

  - **Parameters**

    - `name` –

  - **Returns** –

- **greaterthan**
  `public abstract boolean` **greaterthan**`(DSSValue` **b**`)`

- **greaterthanequal**
  `public abstract boolean` **greaterthanequal**`(DSSValue` **b**`)`

- **lessthan**
  `public abstract boolean` **lessthan**`(DSSValue` **b**`)`

- **lessthanequal**
  `public abstract boolean` **lessthanequal**`(DSSValue` **b**`)`

- **names**
  `java.lang.String[]` **names**`()`
  - **Description copied from org.openmrs.module.dssmodule.state.NamingContext** (**in C.2.2, page 63**)
    All names used by objects in this context
  - **Returns** –

- **notequal**
  `public abstract boolean` **notequal**`(DSSValue` **b**`)`

- **set**
  `void` **set**`(java.lang.String` **name**`, DSSValue` **value**`)`
  - **Description copied from org.openmrs.module.dssmodule.state.NamingContext** (**in C.2.2, page 63**)
    Set a name-value association in this context. This may overwrite any previous association.
  - **Parameters**
    - `name` –
    - `value` –

- **toFloat**
  `public abstract double` **toFloat**`()`

- **toInt**
  `public abstract int toInt()`

- **toLong**
  `public abstract long toLong()`

Members inherited from class DSSValue `org.openmrs.module.dssmodule.value.DSSValue` (in C.11.1, page 180)

add, and, compare, compareTo, concat, div, equal, getDSSValueTimeStamp, getTimeStamp, greaterthan, greaterthanequal, isBoolean, isDate, isFloat, isInt, isList, isNull, isNumeric, isObject, isString, length, lessthan, lessthanequal, mult, not, notequal, or, power, setTimeStamp, setTimeStamp, setTimeStamp, sort, sub, toFloat, toInt, toLong

C.11.10 Class DSSValueString

Declaration public class DSSValueString
extends org.openmrs.module.dssmodule.value.DSSValue (in C.11.1, page 180)

Method summary

add(DSSValue)
and(DSSValue)
concat(DSSValue)
div(DSSValue)
equal(DSSValue)
greaterthan(DSSValue)
greaterthanequal(DSSValue)
length()
lessthan(DSSValue)
lessthanequal(DSSValue)
mult(DSSValue)
notequal(DSSValue)
or(DSSValue)
power(DSSValue)
sub(DSSValue)
toDate()
toFloat()

toInt()
toLong()
toString()

Methods

- **add**

  ```
  public DSSValue add(DSSValue b)
  ```

- **and**

  ```
  public DSSValue and(DSSValue b)
  ```

- **concat**

  ```
  public DSSValue concat(DSSValue b)
  ```

- **div**

  ```
  public DSSValue div(DSSValue b)
  ```

- **equal**

  ```
  public abstract boolean equal(DSSValue b)
  ```

- **greaterthan**

  ```
  public abstract boolean greaterthan(DSSValue b)
  ```

- **greaterthanequal**

  ```
  public abstract boolean greaterthanequal(DSSValue b)
  ```

- **length**

  ```
  public int length()
  ```

- **lessthan**

  ```
  public abstract boolean lessthan(DSSValue b)
  ```

- **lessthanequal**

  ```
  public abstract boolean lessthanequal(DSSValue b)
  ```

- **mult**

  ```
  public DSSValue mult(DSSValue b)
  ```

- **notequal**

  ```
  public abstract boolean notequal(DSSValue b)
  ```

- **or**
  `public DSSValue or(DSSValue b)`

- **power**
  `public DSSValue power(DSSValue b)`

- **sub**
  `public DSSValue sub(DSSValue b)`

- **toDate**
  `public DSSValue toDate() throws java.text.ParseException`

- **toFloat**
  `public abstract double toFloat()`

- **toInt**
  `public abstract int toInt()`

- **toLong**
  `public abstract long toLong()`

- **toString**
  `public java.lang.String toString()`

Members inherited from class DSSValue `org.openmrs.module.dssmodule.value.DSSValue` (in C.11.1, page 180)

add, and, compare, compareTo, concat, div, equal, getDSSValueTimeStamp, getTimeStamp, greaterthan, greaterthanequal, isBoolean, isDate, isFloat, isInt, isList, isNull, isNumeric, isObject, isString, length, lessthan, lessthanequal, mult, not, notequal, or, power, setTimeStamp, setTimeStamp, setTimeStamp, sort, sub, toFloat, toInt, toLong

C.11.11 Class OpenmrsDSSValue

Declaration public class OpenmrsDSSValue
extends java.lang.Object

Constructor summary

OpenmrsDSSValue()

Method summary

main(String[])

Constructors

- **OpenmrsDSSValue**

  `public` **OpenmrsDSSValue**`()`

Methods

- **main**

  `public static void` **main**`(java.lang.String[]` **args**`)`

  - **Parameters**
    - `args` – the command line arguments