



Fables FAQ

Prepared by:
László Gulyás,
Richárd Legéndi and
Attila Szabó



August 2008, Budapest

1 Contents

1	Contents	2
2	General Issues	3
3	Syntax problems	5
4	Scheduling	8
5	Randomization	9
6	Charting Wizard	10
7	Linux problems	11
8	Appendix	12
8.1	The Fables EBNF grammar	12

2 General Issues

1. ***Q: What do I need to run the Fables Integrated Environment (IME)?***

A: Nothing. For the previous versions of the Fables IME it was necessary to have installed a Java(TM) SE Runtime Environment 6 or later. But for the new version we've integrated the JRE into the IME, so it is not necessary.

Repast and any other additional libraries are all included in the installation package.

In short, nothing is needed in order to run the Fables IME.

Moreover, if you'd like to run MEME from the IME, you have to separately install it, since it is not bundled into the application. Make sure you have installed the latest version. You can download it from either <http://mass.aitia.ai> or <http://meme.aitia.ai>.

2. ***Q: What is the latest version of the Fables IME?***

A: The latest version is v1.5, you can find the latest updates and weekly builds of the IME at <http://mass.aitia.ai>.

3. ***Q: Is the Fables IME written in Eclipse?***

A: No, the Fables IME is an Eclipse product with several Eclipse Plug-ins included, like the Fables plug-in and other plug-ins required to compile, run and handle generated Java sources. The current version of the IME is built on Eclipse 3.4, Ganymede.

4. ***Q: Can I download Fables as a plug-in for my Eclipse installation?***

A: Unfortunately no, it is temporary impossible to download the Fables plug-in, and integrate it into an existing Eclipse installation.

5. ***Q: How do I report a bug/request a feature in the Fables IME?***

A: Please post a bug if you see something wrong (even if you are in doubt whether it really is a bug), or feature requests to mass@aitia.ai.

6. ***Q: Where can I find more information about the Fables Compiler/IME?***

A: The official Fables site is <http://fables.aitia.ai>. Another site is used for announcements and upgrades, which is <http://mass.aitia.ai>. If you have further questions, either you can contact us via e-mail at mass@aitia.ai, or subscribe to the MASS mailing list at <http://mass.aitia.ai>.

7. ***Q: What is the easiest way to learn Fables?***

A: We advise to:

- Start with the Fables tutorial included with the software to understand the basics of both the language and the IME.
- Take a look at the Fables user guide to get familiar with the language syntax and the use of the IME.
- The Fables IME has many built-in example applications, examine them, and explore the possibilities of the language.

8. ***Q: The Fables compiler is written in Java, isn't that too slow?***

A: No, Java is often used for writing compilers. The best example is Sun's `javac` compiler, which is written in Java. Sun's Java JVM interpreter is mainly written in Java too (except some platform-specific and low level routines). Sun's Java Web Server is also written in Java.

9. ***Q: Which Repast version is currently supported by the Fables IME?***

A: The Fables compiler generates source for the latest stable Repast J version (3.1). *Repast Symphony and Fables are not compatible yet.*

10. ***Q: When I click either on the Fables icons at the toolbar (e.g. the "Run Fables code" button), it says "Please select and open a valid fabfile.". Which model is going to be processed?***

A: The quick launch icons process the Fables model in the currently active editor. You only have to open the model you wish to process.

11. ***Q: Can I contribute to the Fables demos? What do I have to do?***

A: Yes, we would be happy to include any Fables models you've written in the sample models! You only have to send us the Fables sources, the created chart xml file, a short description and a screenshot of your simulation. Please send them to mass@aitia.ai.

12. ***Q: How is the Fables IME licensed?***

A: Fables IME is free to use for everyone but is not an open source application. See the license on our website at <http://mass.aitia.ai> or *license.txt* in the software's installation directory.

13. ***Q: I had a previous installation of the Fables IME. Is it possible to import my previous workspace?***

A: Importing the previous workspace could be no problem, just select it at the first start of the IME at the workspace selection dialog or use the File → Switch Workspace → Other... menu item to achieve this.

Importing the charts however, could lead to some minor problems. Since both the Fables Compiler and the Charting Wizard is under heavy development, we try our best to uphold backward compatibility with the previous versions. But sometimes the newly introduced features requires major modifications, and they could lead to incompatibility issues.

In these cases, we'd suggest renaming the chart descriptor XML with the F2 function key in the IME to back it up, and creating a new chart.

14. ***Q: I've imported my previous workspace and my previous Fables projects successfully. I can edit my chart, so it seems there's no problem with the Chart descriptor file, but when I run the simulation, I cannot see my charts! I get no errors, what could happened?***

A: We've introduced a new feature in the MassGUI. From now on, it is able to remember the chart positions if they're replaced on the screen, and automatically saves/updates them.

If you have used one of the weekly builds, your GUI settings could have been corrupted. Please locate the *flexdock* directory in your home directory (that's usually under C:\Documents and Settings\flexdock directory and delete all files under the *perspectives* directory. They are always regenerated when a simulation is started and its descriptor hasn't been found by the IME.

15. ***Q: I cannot start MEME from the Fables IME!***

First of all, make sure you already have the latest version of **MEME** installed. It's not bundled into the IME, it **has to be installed separately**.

If you're still not able to run MEME, that's because the IME cannot locate it at its default installation directory. To set it, please click at the Window → Preferences → MEME, and set the location of the **MEME.jar** file via the Browse... button.

16. **Q: There's a Run PET build button in the Fables IDE. What it is used for?**

A: Currently it generates a PET archive in the root directory of your project. It's used to create participatory simulations from Fables models, with the PET framework, that is another member of the MASS suite.

For the details, please visit <http://pet.aitia.ai/>.

3 Syntax problems

17. **Q: What is the syntax definition of the Fables programming language?**

A: You can find the EBNF grammar description of the language attached in appendix 8.1.

18. **Q: I'm getting "Unexpected token: ..." errors. What does that mean?**

A: The most likely reason is that the compiler fails to parse an expression due to syntactical problems. Check if there's a ';' character at the end of each line, whether the bracketing is correct and the time values are defined for the schedules. These are the most common reasons.

19. **Q: What's the difference between the many = operators?**

A: We have 3 different operators similar to =, the =, = = and the := operators. These are similar in their forms, but they do have their own meaning and places where they are usable.

- *The definition symbol (=)* can be used only for defining functions or constants. You can pronounce it as "it is always true, that ...";

```
worldSize = 50;
```

- *The assignment operator (:=)* can be used for assigning a value to a variable you have declared. It is mostly used when you create a new instance of an agent to set up its initial parameters/behavior. Read it as "let x be equal ...". It is also used to set the value of the variables of newly created agents;

```
new Agent[ age := 0, id := 10 ];
```

- *The equals operator (= =)* is a logical operator used to determine whether two items are equal or not (this is the opposite of the <> operator). Read it as "is the same/equal?";

```
println( 1 == 2 );
```

```
→ false
```

20. **Q: How do I make a model parameter I can set before the simulation start, or change during runtime?**

A: Create a constant. Its value will be the parameter's default value. Then just put it in the parameter list of the model's `startUp()` function:

```
model Params {
    myParam = 5; // Parameter's default value

    // Parameter list
    startUp( myParam ) {
        println( "My parameter is currently: " ++ myParam );
    }
}
```

```
}

```

Another – and perhaps more comfortable – way to do it is to use the `param` keyword before the constant definition.

```
model Params {
    param myParam = 5; // Parameter's default value

    // Parameter list
    startup {
        println( "My parameter is currently: " ++ myParam );
    }
}

```

These definitions are semantically equal and it is possible to use both of them at the same time.

Please note parameters always have to have a default value.

21. **Q: What model members can be used as parameters? Can I also assign parameters to classes?**

A: In the current version it is only permitted to use logical, numeric and string parameters. Classes' `startup()` sections cannot have parameter lists.

22. **Q: I've declared a numeric parameter, but when I set its value to real, it doesn't seem it changes. What's the reason?**

A: The most common reason is that the parameter is evaluated as integer by the compiler. Thus, it cannot have a real value (the compiler rounds every assignment downwards, i.e. 0.9 is rounded down to 0). Let's say you have the following model:

```
model Params {
    param = 0 ;

    startup( param ) {
        println( param ) ;
    }
}

```

When you run the model above, and try to set the value of `param` to 0.9, you'll see that its value remains 0. To avoid these errors, declare `param` as a real constant like this:

```
param = 0.0 ;

```

23. **Q: Where do I use the `where` and the `when` keywords? Aren't they the same?**

A: No, they have different meanings:

- The `where` keyword is used for defining a complete expression (a model or class member with local constant definitions). For instance, here's a valid usage of `where`:

```
x = [ 1, 2, 3, 4, 5 ] ;
meanAverage = sumX / size(x) where sumX = sum(x) ;

```

- The `when` keyword is used for defining a logical condition for a set or sequence type after the iterations:

```
rndset = { i : i is [1..10] when uniform(0,1) < 0.5 } ;

```

The example set above contains each element of the interval [1,10] with 50% probability.

24. ***Q: I want to print a sequence or set, but instead of the elements I see only something like [I@9ab0. What's that?***

A: The `print` and `println` functions can only handle two dimensional sequences. When using these functions on higher order sequences, it will simply call the default Java `print` command, which results in this strange String (the `[I` prefix means that this is an array, and the substring after the `@` character is the most unique hash value of the object).

To prevent this, please iterate through the elements and try to print them. The elements have lower dimensions, so the `print` or `println` function can handle them.

25. ***Q: How to acquire the (i,j)th element of my 2 dimensional matrix?***

A: The correct form of indexing is `mx(i)(j)`, not `mx(i,j)` – where `mx` is a two dimensional matrix. The indexing operator is a function too, that has exactly one parameter.

26. ***Q: I want to use a function from the StdLib, but it says it accepts only statements as parameters. What is a statement?***

A: A statement is an expression that has no return value, like the assignment operator (`:=`) or the `println()` function. For the keywords and functions their description contains if they can be used as statements. Please refer to the User Guide or press CTRL + Space in the IME's editor for the full list of available identifiers and their descriptions.

27. ***Q: The descriptions in the content assistant (CTRL + Space) have some strange notations. What is their meaning?***

A: We tried to make function parameters self explanatory. Some of the functions accept only arguments of a special type: if their type is not obvious, these types are shown in `()` brackets after the name of the parameter. For instance, the function `cos()` accepts only a number parameter:

```
cos( x (number) )
```

If there's no notation for a parameter, then there's no restriction for the type of it.

Some functions have optional parameters, they are shown within `[]` brackets. For instance, when you're creating a new discrete uniform random number generator, you could assign a seed value for it. If no seed value is specified, then a default seed value is used:

```
addDiscreteUniform( name (string), [ seed (number) ] )
```

If a function can be called with different signatures, each of the possible formats are noted in the Syntax section of the description. For instance, the `max()` function has two forms:

```
max( collection of numbers )
max( number1, number2, ...)
```

28. ***Q: I've defined a conditional statement with a local constant. It seems it's working somehow wrong...***

A: The problem could be with the local definition. Local definitions are **always evaluated before** the function. For instance let's examine the following code:

```
myFunction() = ( size( Agent ) > 0 ) => println( randomAgent )
              where ( randomAgent = discreteUniform( Agent ) );
```

In the code above, the order of the expressions could cause some misunderstandings. First of all, the `randomAgent` local variable is created, **without** checking if there's any agent instances already. The result of the code above leads to a runtime error in case of an empty collection.

The correct way to implement the above function is to split the function into two segments: perform the check in the first one and call the second if the requirements are met:

```
performFunction = println( randomAgent )
                  where ( randomAgent = discreteUniform( Agent ) );

myFunction() = ( size( Agent ) > 0 ) => performFunction();
```

4 Scheduling

29. **Q:** *I've removed every schedule from my simulation with the `removeSchedule()` function, but the simulation hasn't stopped. Why?*

A: In the generated *Repast* code the declared schedules are only **BasicActions**. We use one global scheduler to merge and process each action.

The simulation doesn't stop running because it can't be determined whether a **Schedule's ActionQueue** is empty or not.

Please, consider using the `pause()` and `stop()` functions to interrupt the simulation.

30. **Q:** *I've examined the generated Java model, but I see only one *Repast* Schedule object, although I've defined several ones! Is that correct?*

A: Yes, that's absolutely correct. The compiler hides the real schedules you've defined, and merges them into one schedule. Both named, unnamed, cyclic and acyclic schedules are handled differently, but in general their first activation turn, cycle and actions are all stored in this general schedule.

31. **Q:** *Where's the `startUp()` section I defined in the generated code? I can't see any functions like that.*

A: The `startUp()` section of a Fables model is called `begin()` in the generated *Repast* code. All statements you've written in the `startUp()` section have their generated code there. Additionally, you can spot some other calls you haven't written in the Fables source, like variable and schedule initializations. These are all necessary to make the model work properly.

For classes, their `startUp()` code is generated directly into their constructors.

32. **Q:** *Is there any difference between the named and unnamed schedules?*

A: Basically there's no difference between them. The benefit of using named schedules is that they can be re-scheduled or stopped dynamically during the simulation with StdLib functions like `addSchedule()` and `removeSchedule()`.

33. **Q:** *There's a function called `addEvent()`. Does it have any relation to schedules? Can I add an event to a running schedule with it?*

A: Yes, this feature has been added in the new version. Function `addEvent()` can process a statement at the given simulation turn, and it can also be used to add a statement to a specified schedule at the given time. The event

inherits the schedule's properties: if the schedule is cyclic, the event is going to be executed periodically as all of the schedule's other events.

34. ***Q: Scheduling functions in the StdLib are working with relative time. What does it mean?***

A: Some of the StdLib functions related to scheduling need a specified time for their execution (like `stop()`, `addEvent()`, etc.). Relative time means the specified time is interpreted as `time + the specified time` (where `time` is the current simulation time). For instance, if the statement `addEvent(3, println("SomeText"))` is evaluated in the first turn, it is going to be activated in the 4th turn; if it is evaluated in the 7th turn, it is going to be activated in the 10th turn.

35. ***Q: Is it possible to make sure that a schedule's events are always executed before another schedule?***

A: Yes. Let's say we have the following schedule definitions:

```
schedule A cyclic 1 {
    1 : println( "A" ) ;
}
schedule B cyclic 1 {
    1 : println( "B" ) ;
}
```

If the events of schedule `A` always has to be executed before those of `B`, simply modify the event definitions of schedule `A`. Since Fables' schedules has been improved to accept real values for event definitions, replace the time 1 of the event definition of schedule `A` to a number below 1, i.e. 0.9:

```
schedule A cyclic 1 {
    0.9 : println( "A" ) ;
}
schedule B cyclic 1 {
    1 : println( "B" ) ;
}
```

5 Randomization

36. ***Q: Function discreteUniform(a,b) doesn't return any values between a and b!***

A: Yes, that is the standard behavior of `discreteUniform()`. For instance, `discreteUniform(1,3,4,5,8,10)` will return one of the values 1, 3, 4, 5, 8 or 10. To make it choose a number randomly from an interval from `a` to `b` use the expression `discreteUniform([a..b])` or `discreteUniformFromTo(a, b)`.

37. ***Q: How do I use the same random number repeatedly in the same definition?***

A: You can use the same random value in a function definition with local definitions like the following:

```
rnd(i) = (act <> i) => act otherwise rnd(i)
        where ( act = discreteUniform([a..b]) );
```

Please note that local definitions cannot have parameters at the moment, so a definition like the following one are illegal and causes compilation errors:

```
wrnd_def = localFunc(1) where (
    localFunc(x) = 2*x; // This is illegal definition
);
```

6 Charting Wizard

38. **Q:** *I can zoom in on several charts (i.e. the TimeSeries chart) if I select a rectangle area on the chart with the mouse while pressing the left mouse button. How do I zoom back?*

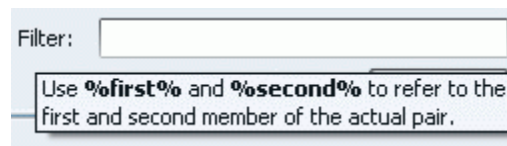
A: Hold down the right mouse button while the mouse cursor is above the zoomed chart, move the mouse to the left and upwards and release the right mouse button. This will restore the default view of the chart.

39. **Q:** *In the Datasource Editor, I would like to create a filter for the elements used in a Series or a Series of pairs. How do I do that?*

A: For the Series, you can refer to the actual element with `%value%`. For instance, if you'd like to include only the positive values from the specified elements to the series, add the following filter expression: `%value% > 0`. The filters has to be valid Java conditions.

For the Series of pairs, you can refer to the current element's first element with `%first%`, and to the second element with `%second%`.

If you're in trouble remembering how to refer to these values, have a look at the tooltip messages of the filter components:



Tooltip message of the Series of pairs Filter section

40. **Q:** *I've found a menu while running my simulation that says I can disable my charts during the run. However, when I uncheck one of my charts, it's still updating. On the other hand, I'm able to disable all of my charts. What's going on, isn't this feature working?*

A: No, this is normal behavior. Charts can be disabled to lower the overhead of the visualization of the simulation (i.e. in order to speed it up). Charts display datasources, whose creation takes most of the time of updating a chart.

When several charts display the same datasource(s), and one of them is disabled, the time consuming task still has to be performed for the other charts. Hence, there's no advantage gained by disabling the chart, and it takes almost no time to redraw it.

On the other hand, when all of the charts that using the same datasource(s) have been disabled, the redrawing stops.

7 Linux problems

37. *Q: When I try to start the IME on Linux (especially on Gentoo-based distributions), I get errors like "xcb_xlib.c:50: xcb_xlib_unlock: Assertion 'c->xlib.lock' failed." What can I do about it?*

A: This is a known compatibility problem between `xcb` and Java. You have two options to resolve this:

- Run the following command as root (Gentoo-based distributions only):

```
CFLAGS="${CFLAGS} -DNDEBUG" emerge -alv xcb libxcb
```

- Locate your Java installation path, and execute the following command as root, replacing the path with yours:

```
sed -i 's/XINERAMA/FAKEEXTN/g' /usr/lib/jvm/java-6-sun-1.6.0.00/jre/lib/i386/xawt/libmawt.so
```

Note: for more information, please visit the following page: http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=6532373

38. *Q: I get a dialog box with a huge error message when I try to start the IME on my Linux distribution saying the virtual machine is cannot be found. What's the problem?*

A: You need to start the IME with the following command (or you can create a script that does this for you):

```
#!/bin/bash
./fables -vm `java-config --java`
```

Replace `./fables` with the proper path.

8 Appendix

8.1 The Fables EBNF grammar

```

Start ::= ( Import )* Model ( "\u001a" )? ( <UNEXPECTED_CHAR: ~[]> )? <EOF>
Import ::= <IMPORT> <IDENTIFIER> ";"
Model ::= ( <DOC_COMMENT> )? <MODEL> ModelName "{" ( ( <DOC_COMMENT> )? ( StartUp |
Schedule | ClassDefinition | Definition ) * "}" ( ";" )?
ModelName ::= Name
Definition ::= VariableDefinition
| ( <PARAM> ConstantDefinition )
| ConstantDefinition
| FunctionDefinition
VariableDefinition ::= <VAR> Reference ( "," Reference ) * ";"
Reference ::= Name ( ":" "=" ImperativeExpression )?
ConstantDefinition ::= Name "=" CompleteExpression ";"
FunctionDefinition ::= Name "(" ( " " | ArgumentList " " ) "=" CompleteExpression ";"
LocalDefinition ::= Name ( "(" ArgumentList " " )? "=" Expression
ClassDefinition ::= <CLASS> ( <PERSISTENT> )? ClassName "{" ( ( <DOC_COMMENT> )? (
ClassStartUp | Schedule | Definition ) * "}" ( ";" )?
ClassStartUp ::= <STARTUP> ( "(" " " )? "{" ( Statement ) * "}" ( ";" )?
StartUp ::= <STARTUP> ( "(" ( " " | ParameterList " " ) )? "{" ( Statement ) * "}" ( ";" )?
ParameterList ::= Identifier ( "," Identifier ) *
Schedule ::= <SCHEDULE> ( "{" | <CYCLIC> Number "{" | ScheduleName ( <CYCLIC>
Number )? "{" ) ( Event ) * "}" ( ";" )?
ScheduleName ::= Name
Event ::= Time ":" ( Statement ) +
Time ::= Number
ClassName ::= Name
ArgumentList ::= Variable ( "," Variable ) *
Variable ::= Name
CompleteExpression ::= Expression ( <WHERE> "(" LocalDefinition ( "," LocalDefinition ) * ( ","
)? ")" )?
BasicExpression ::= AtomExpression
| SetExpression
| SequenceExpression
| BracketedExpression
| ForEachExpression
AtomExpression ::= ( Identifier ( "(" " " ) )? )
| Number
| Character
| String
BracketedExpression ::= "(" ( TupleExpression | BinaryOperator ) ")"
ForEachExpression ::= <FOR> <EACH> Name <IN> Expression <DO> ( ( Expression ) | ( (
"{" ( Statement ) * "}" ) ) )
TupleExpression ::= Expression ( "," Expression ) *
BinaryOperator ::= MultiplicativeOperator
| AdditiveOperator
| LogicalOperator
| RelationalOperator
| InOperator

```

```

LogicalOperator ::= AndOperator
                | OrOperator
MemberSelector ::= Name ( "(" TupleExpression ")" )?
                | Integer
UnaryHead      ::= ( "(" ( AdditiveOperator | MultiplicativeOperator ) ")" )
                | ( AdditiveOperator | MultiplicativeOperator )
FunctionExpression ::= ( ( UnaryHead ( ( "(" TupleExpression ")" ) | ( TupleExpression ) ) )
| ( BasicExpression ( ( ( "(" TupleExpression ")" ) | ( MemberOperator MemberSelector ) ) ) ) )
MemberOperator ::= ","
UnaryExpression ::= UnaryMinusExpression
                | FunctionExpression
UnaryMinusExpression ::= "-" FunctionExpression
PowerExpression ::= UnaryExpression ( "^" UnaryExpression )*
MultiplicativeExpression ::= PowerExpression ( MultiplicativeOperator PowerExpression )*
MultiplicativeOperator ::= MulOperator
                        | PerOperator
                        | DivOperator
                        | ModOperator
                        | RemOperator
AdditiveExpression ::= MultiplicativeExpression ( AdditiveOperator MultiplicativeExpression )*
AdditiveOperator ::= PlusOperator
                  | MinusOperator
ConcatenateExpression ::= AdditiveExpression ( "+" AdditiveExpression )*
RelationalExpression ::= ConcatenateExpression ( ( EqualExpression | NotEqualExpression |
LesserExpression | LesserOrEqualExpression | GreaterExpression | GreaterOrEqualExpression | InOperator
ConcatenateExpression ) )?
EqualExpression ::= EqualOperator ConcatenateExpression ( EqualOperator ConcatenateExpression
)*
NotEqualExpression ::= NotEqualOperator ConcatenateExpression ( NotEqualOperator
ConcatenateExpression )*
LesserExpression ::= LesserOperator ConcatenateExpression ( LesserOperator
ConcatenateExpression )*
LesserOrEqualExpression ::= LesserOrEqualOperator ConcatenateExpression (
LesserOrEqualOperator ConcatenateExpression )*
GreaterExpression ::= GreaterOperator ConcatenateExpression ( GreaterOperator
ConcatenateExpression )*
GreaterOrEqualExpression ::= GreaterOrEqualOperator ConcatenateExpression (
GreaterOrEqualOperator ConcatenateExpression )*
RelationalOperator ::= EqualOperator
                  | NotEqualOperator
                  | LesserOperator
                  | GreaterOperator
                  | LesserOrEqualOperator
                  | GreaterOrEqualOperator
NotExpression ::= RelationalExpression
               | UnaryNotExpression
UnaryNotExpression ::= <NOT> RelationalExpression
LogicalExpression ::= NotExpression ( ( LogicalAndExpression | LogicalOrExpression ) )*
LogicalAndExpression ::= AndOperator NotExpression
LogicalOrExpression ::= OrOperator NotExpression
NonImperativeExpression ::= LogicalExpression
AssignmentExpression ::= NonImperativeExpression ( "=" Expression )?

```

```

ConstructExpression ::= ( <NEW> | <CREATE> ) Identifier ( "[" ( ( "]" ) | ( Identifier ":" Expression ( "," Identifier ":" Expression ) * "]" ) ) )?
DestructExpression ::= <DELETE> FunctionExpression
ImperativeExpression ::= AssignmentExpression
                        | ConstructExpression
                        | DestructExpression
ConditionalExpression ::= ImperativeExpression ( ( "=" ImperativeExpression ( "|" ImperativeExpression "=" ImperativeExpression ) * ( <OTHERWISE> ( "=" )? ImperativeExpression ) ) )?
Statement ::= EmptyStatement
            | Expression ";"
EmptyStatement ::= ";"
Expression ::= ConditionalExpression
SetExpression ::= "{" ( ( ( Iterations ( IterationCondition )? ) | ( Expression ( IntervalExpression | ( MultiValueExpression )? ( "," )? ( ":" Iterations ( IterationCondition )? ) ) ) ) )? "}"
SequenceExpression ::= "[" ( ( ( Iteration ( IterationCondition )? ) | ( Expression ( IntervalExpression | ( MultiValueExpression )? ( "," )? ( ":" Iterations ( IterationCondition )? ) ) ) ) ) )? "]"
MultiValueExpression ::= "," Expression ( "," Expression ) *
IntervalExpression ::= ".." Expression ( "," Expression ".." Expression )? ( "|" Expression )?
Iterations ::= Iteration ( "," Iteration ) *
IterationCondition ::= <WHEN> Expression
Iteration ::= Name <IS> Expression
Number ::= Integer
           | Float
PlusOperator ::= "+"
MinusOperator ::= "-"
MulOperator ::= "*"
PerOperator ::= "/"
DivOperator ::= <DIV>
ModOperator ::= <MOD>
RemOperator ::= <REM>
PowOperator ::= "^"
AndOperator ::= <AND>
OrOperator ::= <OR>
InOperator ::= <IN>
EqualOperator ::= "=="
NotEqualOperator ::= "<>"
LesserOperator ::= "<"
GreaterOperator ::= ">"
LesserOrEqualOperator ::= "<="
GreaterOrEqualOperator ::= ">="
Name ::= <IDENTIFIER>
Identifier ::= ( <SELF> | <IDENTIFIER> )
Integer ::= <INTEGER_LITERAL>
Float ::= <FLOATING_POINT_LITERAL>
Character ::= <CHARACTER_LITERAL>
String ::= <STRING_LITERAL>

```