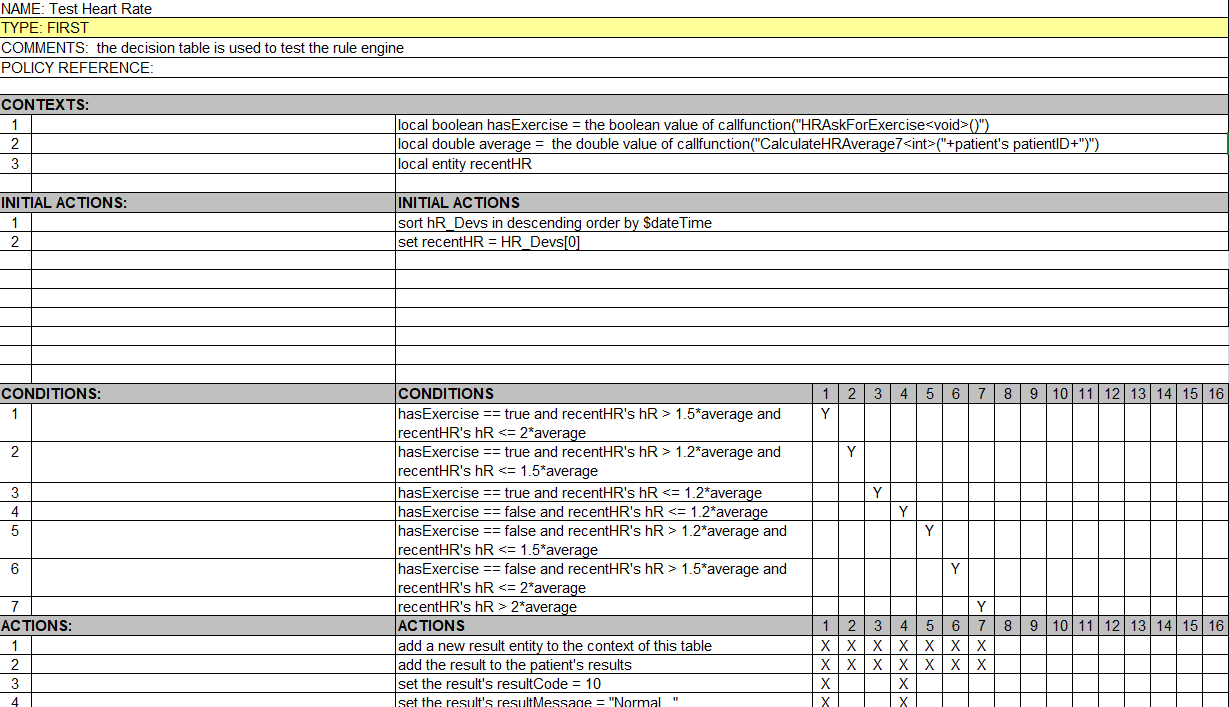
# Architecture



The figure above is a sample of a decision table, then we will make a description from top to bottom for each details.

1. NAME: the name of a decision table, the sample above is “Test Heart Rate”.

PS: When we reference a table name, we replace space with “\_”, as showed in the figure followed as *perform Evaluate\_zipcode*, means to perform the table *Evaluate zipcode*（the table can also be named as *Evaluate\_zipcode*）:



1. TYPE: used to make it clear how the rules will be performed. The types include *FIRST* and *ALL*.

PS: *FIRST* means to perform the first rule which meets the conditions from left to right.

*ALL* means to perform all the rules that meet the conditions from left to right..

1. COMMENTS: the description of the decision table.
2. POLICY REFERENCE: no real use, you can just take it as COMMENTS.
3. CONTEXTS: stands for the context, this section is optional.
4. INITIAL ACTIONS: to do some operations before the CONDITIONS and ACTIONS, this section is also optional.
5. CONDITIONS: the conditions which are all bool expressions.
6. ACTIONS: the actions to do of a rule.

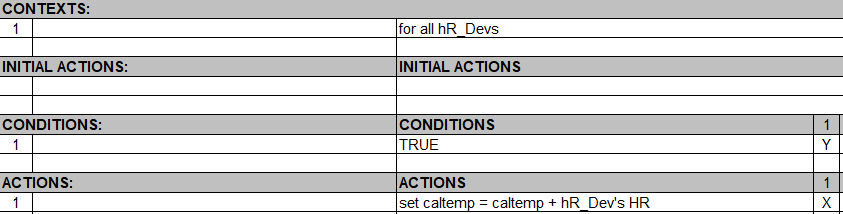
# CONTEXTS

Section Contexts is mainly used to make it clear what context will be used followed. The following statements are only used in CONTEXTS.

## for all

*for all* means to loop each entity in an array, this statement only exists in section CONTEXTS, and it could be nested by other *for all* loops. In each loop, the rules will be judged and executed. The statement is: *for all <Array>*.

How to access the data entity in each loop: you can just use the entity name to access the data entity in each loop, or you can straightly use the name of attributes in the data entity. As is showed followed:



Or you can use *set caltemp = caltemp + HR* to replace the action above.

## for all whose…

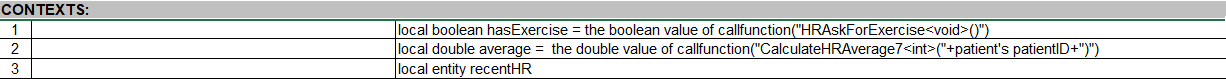
*for all whose* statement is used to help filter the data in the array. For example: *for all HR\_Devs whose HR > 0*. The statement is: *for all <Array> whose <BoolExpression>*.

## for the first….

*for the first* statement is used to choose the first data entity of the array which meet the requirements. For example: *for the first of the HR\_Devs whose HR > 0*. The Statement is: *for the first of the <Array> whose <BoolExpression>*.

## local

*local* statement is used to define a local variable or entity used to store data or do some calculation, you can straightly assign some value when declare the local variable or just declare and assign the value for it in section INITIAL ACTIONS or do some calculation. The statement is: *local <Type> <VarName> = <Value>* or just *local <Type> <VarName>*. For example: *local double average = 0* or *local entity recentHR*.



# INITIAL ACTIONS & ACTIONS

Section ACTIONS is used to declare the operations of actions. The difference between INITIAL ACTIONS and ACTIONS is that INITIAL ACTIONS are performed before CONDITIONS and ACTIONS. The statements of this two sections are the same.

## Set

*set* statement is used to assign value to a variable. The statement is: *set <Variable> = <Value>*. For example: *set recentHR = HR\_Devs[0]* or *set average = 0*.

## Add

*add* statement is used to add variables into a certain array. The statement is: add <Variable/Vaule> to <Array/Number>. For example: *add recentHR to HR\_Devs* or *add a new HR\_Dev entity to HR\_Devs*. And we can also add a variable into the context: *add a new HR\_Dev entity to the context of the table*.



In addition, *add* statement can also stands for addition as is mentioned above, for example *add 1 to the counter*. But we recommend this statement instead of using add statement: *set counter = counter + 1*.

## Subtract

*subtract* statement is used for subtraction. The statement is: *subtract <Value> from <Value/Variable>*. For example: *subtract 1 from counter*.

## Remove

*remove* statement is used to remove an element from an array. The statement is: *remove the <IntNumber> element from the <Array> array* or *remove <Variable> from the <Array> array*. For example: *remove the 10 element from the HR\_Devs array* or *remove HR\_Dev from the HR\_Devs array* in a loop context.

## Randomize

*randomize* statement is used to shuffle the array. The statement is: *randomize <Array>.* For example: *randomize HR\_Devs*.

## Clear

*clear* statement is used to clear an array and make it empty. The statement is: *clear <Array>*. For example: *clear HR\_Devs*.

## Sort

*sort* statement is used to sort an array. The statement is: *sort <Array> in ascending/descending order by $<AttributeName>*. For example: *sort HR\_Devs in descending order by $DateTime*.

## New

*new* statement is used to declare a new entity. The statement is: *new <ClassName> entity* for declaring a new entity, or *new <ClassName> array* for declaring a new array. For example: *set recentHR = new HR\_Dev entity*.

## Something about Date

**Date Transform**: We can use *(Date)”MM/DD/YYYY”* to convert the string to date, returns a date. For example: *set DateTime = (Date)”10/3/2007”*.

**Get Current Date**: *current date*, and this statement can be used straightly, stands for the current date.

**Some operations**: *add <IntNumber> days/months/years to <Date>*. For example: *add 20 days to cdate*, the cdate is a date entity defined before. PS: The date entity here must be a straight date entity, not a nested expression.

*subtract <IntNumber> days/months/years from <Date>*, the same as above.

*<Date> +/- <IntNumber> days/months/years*, return a date. For example: *set cdate = cdate + 10 days*.

We can use >, <, == straightly to judge the two date which is earlier or later. Return a bool value.

## String相关

**String Transform**: *(String)<Number>, return a string*.

**String Connection**: *<String>+<Number>+……return a string*.

**String Case Conversion**: *change <String> to lower/upper case*, return a string. For example: *set str = change str to lower case*

**Substring**: *substring of <String> from <IntNumber> to <IntNumber>*, return a string. For example: *set str = substring of str from 0 to 5.*

**Trim**: *trim(<String>)*, return a string.

**Get the Start Index of Substring**: *index of <String> in <String>*, return an integer. For example: *set indexer = index of “asdf” in str*

**RegEX Match**: *<String> matches <String>*, return a boolean. This statement means like the former String.matches(the later String) in java.

# CONDITIONS

You can write any bool expressions here, normally you can use: >, <, ==.

## Custom Functions

If you can't use the statements described above to complete the complex logic you need, we also provide a call to custom functions to help you accomplish this complex logic.

The statement is: *callfunction(“FunctionName<Type, Type……>(Para1, Para2……)”)*. The types here are all simple basic types, including int, double, string and char. The string type must use single quotes to wrap the string. And then you can implement your custom function in edu.dhu.lib.Utils class.



We also provide another *callfunctionwithobj(“FunctionName<Type, Type……>”, Para1, Para2……)*. The type here could be an entity, but could not be an array for now, but you can push the patient straightly and we provide IRObjConverter class to transform the patient data into java class.

For example: local double average = the double value of callfunctionwithobj(“CalculateHRAverage<entity>”, patient). The patient here is the whole data entity of the context we defined.

You can do some test after your finishing your custom functions.

# The Data Entities and Data Classes

The data classes or data entities in this project are fixed, but if the corresponding design in database is changed, we also need to modify some certain files. 具Including Dev\_edd.xls file in DevEDD directory, Dev\_map.xml file in xml directory, and all the data class in the java project. The location is edu.dhu.DTRules.entities. And we also need to modify the transform of data in .net project.

# Some Examples

If you still have no idea about how to write your decision tables, you can reference to the samples we give. The location is in the samples directory.