## Loose Ideas

Conditions,

2020-05-24

Some information about this notation might already be found in some of these places in the Circle Language Spec:

* "Inheritance \ Enforcing & Preventing Specialization" seems to use the Conditions notation e.g. in the section "Requirements for Other Side of Connection", "Abstract Class", "Sealed / Final Class", "Optional/Required".
* "Parameters \ Imported Parameter Concepts \ Required & Optional" seems to demonstrate the notation for one of the most basic conditions: required/optional. This is a condition, but not the generic condition notation, though, that other conditions might use.

JJ

-----

"Nice" programming language conditions

class CoffeeMachine{

CoffeeCup brew(Coffee cfe)

requires cfe.beanAge < 10 : "Beans are too old to brew"

ensures result.temp > 155 : "Coffee isn't hot enough to serve" {

return new CoffeeCup(coffee: cfe, temp:160, isFull:true);

}

}

class Coffee{

int beanAge;

}

class CoffeeCup{

int temp;

boolean isFull;

Coffee coffee;

}

-----

Automatically imposing conditions

-----

< Optimization issue >

Checking conditions takes up small quantities of time. It is a balance between the time required to check conditions, time required to rollback the method’s data mutations, and the likeliness that an error will occur.

You might want to automatically regulate checking the conditions or not, but this regulation would probably add a considerable performance penalty too. You might also do a periodic automatic tuning of the methods, so that conditions will be added or removed based on how many times an error occurs, and if it’s averagely faster to do a rollback every now and then, or a conditional check all the time.

Maybe the conditional checks will be checked under the surface anyway, only you could do some of the checks earlier at the beginning of the method.

< / Optimization issue >