1 Syntax

create{IPLAYER} player creates an instance of IPLAYER and assigns it to player

player := create{IPLAYER} same as create{IPLAYER} player

iplayer.play_mp3(create{MP3_FILE}.make) initializes and passes an MP3-File to the feature

check not player.is_playing end Work just like pre-/postconditions, just that they're right in the body (assertion)

NATURAL from 0 up

modulo \\

integer division 5 // 2 = 2

v ?= e downcast, needs runtime checking

h@k dictionary access (stands for h.at(k))

attached a_node synonym to a_node /= Void

str ~ str same as str.is_equal(str), just void-safe

2 Semantics

Qualified call explicitly lists the target object, e.g. $x.f(args)$

Feature export
- Information hiding only applies to qualified calls.
- Features exported to $NONE$ (a class inheriting from all classes) can not be accessed by clients. It may only be accessed within the defining class itself or its descendants.
- Creation procedures exported to $NONE$ may only be used as creation procedures by clients.

3 Containers

Hash Tables
- Open Hashing: Every entry has a Linked List, collided keys are therefore just stored in the same list.
- Closed Hashing: In case of a collision, look for open spots somewhere around the index, where the collision took place (that's what Hash Tables in Eiffel use)

Tuples Tuple [A,B,C] conforms to [A,B] and [A]

4 Design by Contracts

Class invariant Must be satisfied after creation and after the execution of any feature by any client (so affects **qualified** calls only, **unqualified** calls and calls to **non-exported** features may break the invariant)
If inherited, may only be stronger or equally strong

Precondition If inherited, may only be weaker or equally strong

Postcondition If inherited, may only be stronger or equally strong

Loop Invariant Boolean expression to determine whether the loop achieves its purpose. Needs to be $TRUE$ after initialization of the loop and preserved by the loop body (so has to be true after the last step as well).

Loop Variant Integer expression to determine whether the loop will terminate. Has to be non-negative after initialization of the loop and decreased by at least one, while remaining non-negative, by any execution of the loop body.

5 Agents

LIST

- do_all(action: PROCEDURE)
  apply action to every item
- do_if(action: PROCEDURE, test: FUNCTION)
  apply action to every item that satisfies test
- there_exists(test: FUNCTION): BOOLEAN
  is test true for at least one element
- for_all(test: FUNCTION)
  is test true for all elements