

Cedar Style Sheet

<p>No common suffix on the names of DEFINITIONS modules</p>	<pre>-- FILE: CedarSample.mesa -- Last edited by Mitchell, April 23, 1980 12:05 PM CedarSample: DEFINITIONS = BEGIN upperLimit: INTEGER = 32; -- name the upper limit of an interval type IntervalType: TYPE = [0 .. upperLimit]; -- preferred form for interval type Sample: TYPE = REF SampleRec; SampleRec: TYPE = RECORD[val: Value, next: Sample]; SampleSet: TYPE = REF SampleSetRec; SampleSetRec: TYPE = RECORD[head: Sample, count: Value]; Problem: ERROR [reason: ErrorCode]; ErrorCode: TYPE = {damagedSample, callingError, programError}; NewSample: PROCEDURE [val: Value] RETURNS [Sample]; NewSampleSet: PROCEDURE RETURNS [nilSet: SampleSet]; RemoveSample: PROCEDURE [toBeRemoved: Sample, from: SampleSet] RETURNS [inSet: BOOLEAN, setMinus: SampleSet]; -- ERRORS: Problem[damagedSample] -- SIGNALS: ResumableCondition, SomeSignal emptySet: SampleSet = NIL; END.</pre>	<p>Standard prelude for a module</p> <p>Using a type is better than using anonymous type constructor.</p> <p>Use a small set of ERRORS with an error code parameter.</p> <p>attach stylized comments to procedures that generate ERRORS or SIGNALS.</p>
<p>Capitalize the first letter of module procedure, signal, or type names</p>	<pre>This is where a global description of the module goes CHANGE LOG Changed by: YourName: DateTime DescriptionOfChange</pre>	<p>Standard postlude for a module includes a history log of (non-trivial) changes to the module</p>
<p>Capitalize the first letter of each imbedded word of a multi-word name</p>	<pre>-- FILE: CedarSampleImpl.mesa -- Last edited by Mitchell, April 23, 1980 3:19 PM DIRECTORY SomeInterface USING [SomeProc, SomeType], CedarSample; PROGRAM CedarSampleImpl PROGRAM IMPORTS SI: SomeInterface EXPORTS CedarSample = BEGIN OPEN CedarSample; Problem: ERROR[reason: ErrorCode] = CODE; ResumableCondition: PUBLIC SIGNAL = CODE; SomeSignal: PUBLIC SIGNAL = CODE; NewSample: PUBLIC PROCEDURE [val: Value] RETURNS [Sample] = BEGIN sample: Sample; x: SI.SomeType = 0; IF val = 0 THEN x := SomeProc(x); RETURN [sample]; END; NewSampleSet: PUBLIC PROCEDURE RETURNS [nilSet: SampleSet] = BEGIN nilSet _ AllocatesSampleSet[allocFault => ERROR Problem[programError]] nilSet [head: NIL, count: 0]; END; Bug: ERROR = CODE; RemoveSample: PUBLIC PROCEDURE [toBeRemoved: Sample, from: SampleSet] RETURNS [inSet: BOOLEAN, setMinus: SampleSet] = BEGIN OPEN SI; IF . . . THEN ERROR Problem[damagedSample]; WITH refAnyVar SELECT FROM r: REF REAL => r^ - r^ + 1.0; i: REF INT => { . SomeProc[y]; . . }; b: REF BOOLEAN => SIGNAL ResumableCondition; ENDCASE => ERROR Bug; . . . END; AllocFault: ERROR CODE; -- error for local use in this module AllocateSampleSet : PROCEDURE RETURNS [unitedSet: SampleSet] = BEGIN . . . END; END.</pre>	<p>OK to use unnamed OPEN of interface if DIRECTORY entry has a USING list</p> <p>OK to OPEN interface that module implements</p> <p>Only let signals that are part of abstraction escape out of it.</p> <p>OK to OPEN an interface in a local scope where it is heavily used</p> <p>Only raise SIGNALS using SIGNAL, and ERROR using ERROR</p> <p>NO using ENDCASE to handle a single remaining case: use as "OTHERWISE" or to generate an ERROR</p> <p>OK to have locally defined ERRORS and SIGNALS</p>
<p>PROGRAM module name is normally formed by suffixing interface name with "Impl". Alternatively, name may be totally different than interface name</p>	<pre>CHANGE LOG Changed by: YourName: DateTime DescriptionOfChange</pre>	<p>Standard postlude for a module includes a history log of (non-trivial) changes to the module</p>
<p>NO names in same scope differing only by letter case distinctions except a value with same name as its type but with lowercase first letter</p>	<pre>CHANGE LOG Changed by: YourName: DateTime DescriptionOfChange</pre>	<p>Standard postlude for a module includes a history log of (non-trivial) changes to the module</p>
<p>Qualify identifiers from interfaces</p>	<pre>CHANGE LOG Changed by: YourName: DateTime DescriptionOfChange</pre>	<p>Standard postlude for a module includes a history log of (non-trivial) changes to the module</p>
<p>Keyword constructors, argument lists and extractors preferred for multiple component constructors</p>	<pre>CHANGE LOG Changed by: YourName: DateTime DescriptionOfChange</pre>	<p>Standard postlude for a module includes a history log of (non-trivial) changes to the module</p>
<p>Use REF ANY instead of variant records and discriminate</p>	<pre>CHANGE LOG Changed by: YourName: DateTime DescriptionOfChange</pre>	<p>Standard postlude for a module includes a history log of (non-trivial) changes to the module</p>
<p>Calls on single-argument procedures don't have to use keyword notation</p>	<pre>CHANGE LOG Changed by: YourName: DateTime DescriptionOfChange</pre>	<p>Standard postlude for a module includes a history log of (non-trivial) changes to the module</p>