1 Bytecode Abstract Syntax

b	::=	$v\ P\ \overline{i}\ \overline{M}$	*bytecodefile*
v	::=	*magic major.minor*	*magic+version number*
P	::=	*fully qualified path*	*path to module*
i	::=	*import* μ *URI* : τ *as* x	*module import*
μ	::=	[`metadata`] [`type`]	
M	::=	`module` $P : \tau = e$	*top level modules*
	|	`type` $P = T\ [\delta]$	
e	::=	x	*expressions*
	|	`new` $\tau\ \{x \Rightarrow \overline{d}\}$	
	|	$e.m(\overline{e})$	
	|	$e.f$	
	|	$e.f = e$	
	|	`let` $x = e$ `in` e	
	|	$\mathscr{L}$	
	|	e.`match` $\overline{x : p.L \Rightarrow e}$ [`else` e]	
	|	$\overline{e}$	
$\mathscr{L}$	::=	*string*	*literals*
	|	*integer*	

d	::=	`val` $f : \tau = e$	*declarations*
	|	`var` $f : \tau = e$	
	|	`def` $m(\overline{x : \tau}) : \tau = e$	
	|	`type` $L = T\ [\delta]$	
T	::=	c	*type desc.*
	|	`extag` c	
	|	`datatag` $\overline{p.L}\ c$	
c	::=	τ	*case desc.*
	|	`extends` $p.L\ \tau$	
τ	::=	$\tau\ \{\mathtt{x} \Rightarrow \overline{\sigma}\}_s$	*type*
	|	$p.L$	
	|	$\top$	
	|	$\bot$	
	|	?	
p	::=	x	*paths*
	|	$p.f$	
s	::=	`stateful` | `pure`	
σ	::=	`val` $f : \tau$	*decl type*
	|	`var` $f : \tau$	
	|	`def` $m : \Pi\overline{x{:}\tau}.\tau$	
	|	`type` $L = T\ [\delta]$	
	|	`type`$_s$ $L\ [\delta]$	
δ	::=	`metadata` e	*metadata*

Notation: overbar means a list of elements, as in Java