Abstract syntax

Int	::=	...
String	::=	...
id	::=	... any lowercase-starting java id
Tid	::=	... any uppercase-starting java id
literal	::=	*Int* | *String* | `true` | `false` | *type* | *path*
type	::=	`<>` | `outer`$_1$ `....outer`$_n$`.` *Tid*$_1$ `....` *Tid*$_k$ | *predefinedTypes* | *type*`[ ]`
predefinedTypes	::=	`Object` | `RuntimeException` | `CompositionException`
		| `MissingInternalMemberException` | `MissingNestedClassException`
		| `MissingDefinedInternalMemberException` | `IncompatibleConstructorsException`
		| `EmptyPathProvidedException` | `NonCompatibleInternalMembersException`
		| `ImplementsRelationBrokenException` | `NoNRedirectableToDestinationException`
		| `InvalidStringProvidedException` | `NoNestedRequiredException`
		| `MemberRedirectedOnItselfException`
		| `String` | `int` | `void` | `class` | `path` | `iname` | `cpath` | `type` | `boolean`
path	::=	`.<>` | `.`*id*$_1$ `....` *id*$_n$
paramListUse	::=	`(`e_1`,...,`e_n`)`
initListArray	::=	`{`e_1`,...,`e_n`}`
paramDec	::=	`(`*type*$_1$ *id*$_1$`,...,`*type*$_n$ *id*$_n$`)`
baseE	::=	`(` *e* `)` | *e*`.`*id* | *id* | `new` *type paramListUse* | *e*`.`*id paramListUse* | *literal* | *bc*
		| `new` *type* `[ ]` *initListArray* | `new` *type* `[` e_1`]` `{`e_2`}`
unOpE	::=	*baseE* | `!` *unOpE* | `copy` *unOpE* | *unOpE* `[` *e* `]` | *unOpE* `[ members ]`
		| *unOpE* `[ restrict` e_1 `in` e_2`]`
		| *unOpE* `[ alias` e_1 `to` e_2 `in`e_3`]`
		| *unOpE* `[redirect` e_1 `of` e_2`to` e_3`]`
		| *unOpE* `[ alias` e_1 `to` e_2`]`
		| *unOpE* `[redirect` e_1 `to` e_2`]`
e	::=	*unOpE* | *e binOp e*
binOp	::=	`&&` | `||` | `==` | `<` | `<=` | `>` | `>=` | `[+]` | `+` | `-` | `*` | `/`
stm	::=	*type id* `=` *e*`;` | `if (` *e* `)` *stm* `else` *stm* | `if (` *e* `)` *stm* | `while (` *e* `)` *stm* | *block* |
		`return` *e*`;` | `throw` *e*`;` | `print (` *e* `);` | e_1 `=` e_2`;` | `for (` *type id* `:` *e* `)` *stm* |
		`try` *block*`catch (` *type*$_1$ *id*$_1$ `)` *block*$_1$ ... `catch (` *type*$_n$ *id*$_n$ `)` *block*$_n$ | *validStmExp*`;`
validStmExp	::=	`new` *type paramListUse* | *e*`.`*id paramListUse*
block	::=	`{` *stm*$_1$ ... *stm*$_n$ `}`
bc	::=	*ch*`{` *member*$_1$ ... *member*$_n$ `}`
ch	::=	`abstract implements` *type*$_1$`,...,`*type*$_n$
		| `abstract` | `implements`*type*$_1$`,...,`*type*$_n$ | Λ
member	::=	`abstract` *type id*`;` | *type id*`;`
		| `abstract` *type id paramDec*`;` | *type id paramDec block*
		| `constructor` *paramDec* `{ this.`*id*$_1$ `=` e_1`; ...this.`*id*$_n$ `=` e_n`; }`
		| *Tid* `=` *e*