7.1 LiveOak-0: Expressions, Assignment Statements, Sequential Control Flow

Id	Production	Comments
P1		
P2		
P3	⟨*Program*⟩ → ⟨*Body*⟩	Start here.
P4		
P5		
P6	⟨*Body*⟩ → (⟨*VarDecl*⟩)* ⟨*Block*⟩	
P7		
P8		
P9	⟨*VarDecl*⟩ → ⟨*Type*⟩ ⟨*Identifier*⟩ (`,` ⟨*Identifier*⟩)* `;`	Only declaration; no initialization.
P10	⟨*Block*⟩ → `{` (⟨*Stmt*⟩)+ `}`	Extraneous braces to anticipate later levels.
P11		
P12		
P13		
P14		
P15	⟨*Stmt*⟩ → ⟨*Var*⟩ `=` ⟨*Expr*⟩ `;`	Basic assignment statement.
P16	⟨*Stmt*⟩ → `;`	Null statement.
P17		
P18		
P19		
P20		
P21		
P22	⟨*Expr*⟩ → `(` ⟨*Expr*⟩ `?` ⟨*Expr*⟩ `:` ⟨*Expr*⟩ `)`	Ternary choice operation.
P23	⟨*Expr*⟩ → `(` ⟨*Expr*⟩ ⟨*Binop*⟩ ⟨*Expr*⟩ `)`	Binary operations.
P24	⟨*Expr*⟩ → `(` ⟨*Unop*⟩ ⟨*Expr*⟩ `)`	Unary operations.
P25	⟨*Expr*⟩ → `(` ⟨*Expr*⟩ `)`	Extra parentheses.
P26	⟨*Expr*⟩ → ⟨*Var*⟩	A named variable.
P27	⟨*Expr*⟩ → ⟨*Literal*⟩	A literal value.
P28	⟨*Binop*⟩ → `[+-*/%&|<>=]`	Valid binary operators.
P29	⟨*Unop*⟩ → `[~!]`	Valid unary operators.
P30		
P31		
P32	⟨*Type*⟩ → `int`	
P33	⟨*Type*⟩ → `bool`	
P34	⟨*Type*⟩ → `String`	
P35	⟨*Literal*⟩ → ⟨*Num*⟩	Integer literals. Non-negative only.
P36	⟨*Literal*⟩ → `true`	Boolean literal `true`.
P37	⟨*Literal*⟩ → `false`	Boolean literal `false`.
P38	⟨*Literal*⟩ → ⟨*String*⟩	A string literal.
P39		
P40		
P41	⟨*Var*⟩ → ⟨*Identifier*⟩	
P42	⟨*Num*⟩ → (`[0-9]`)+	Regular expression for integer literals.
P43	⟨*String*⟩ → `"` (*ASCII character*)* `"`	Regular expression for string literals.
P44	⟨*Identifier*⟩ → `[a-zA-Z]([a-zA-Z0-9'_'])*`	Regular expression for identifiers.

7.2 LiveOak-1: Imperative Programming with Structured Control Flow

Id	Production	Comments
P1		
P2		
P3	⟨*Program*⟩ → ⟨*Body*⟩	
P4		
P5		
P6	⟨*Body*⟩ → (⟨*VarDecl*⟩)* ⟨*Block*⟩	
P7		
P8		
P9	⟨*VarDecl*⟩ → ⟨*Type*⟩ ⟨*Identifier*⟩ (`,` ⟨*Identifier*⟩)* `;`	
P10	⟨*Block*⟩ → `{` (⟨*Stmt*⟩)+ `}`	
P11		
P12	⟨*Stmt*⟩ → `if` `(` ⟨*Expr*⟩ `)` ⟨*Block*⟩ `else` ⟨*Block*⟩	Conditional statement.
P13	⟨*Stmt*⟩ → `while` `(` ⟨*Expr*⟩ `)` ⟨*Block*⟩	Loop statement.
P14	⟨*Stmt*⟩ → `break` `;`	Break out of enclosing loop.
P15	⟨*Stmt*⟩ → ⟨*Var*⟩ `=` ⟨*Expr*⟩ `;`	
P16	⟨*Stmt*⟩ → `;`	
P17		
P18		
P19		
P20		
P21		
P22	⟨*Expr*⟩ → `(` ⟨*Expr*⟩ `?` ⟨*Expr*⟩ `:` ⟨*Expr*⟩ `)`	
P23	⟨*Expr*⟩ → `(` ⟨*Expr*⟩ ⟨*Binop*⟩ ⟨*Expr*⟩ `)`	
P24	⟨*Expr*⟩ → `(` ⟨*Unop*⟩ ⟨*Expr*⟩ `)`	
P25	⟨*Expr*⟩ → `(` ⟨*Expr*⟩ `)`	
P26	⟨*Expr*⟩ → ⟨*Var*⟩	
P27	⟨*Expr*⟩ → ⟨*Literal*⟩	
P28	⟨*Binop*⟩ → `[+-*/%&|<>=]`	
P29	⟨*Unop*⟩ → `[~!]`	
P30		
P31		
P32	⟨*Type*⟩ → `int`	
P33	⟨*Type*⟩ → `bool`	
P34	⟨*Type*⟩ → `String`	
P35	⟨*Literal*⟩ → ⟨*Num*⟩	
P36	⟨*Literal*⟩ → `true`	
P37	⟨*Literal*⟩ → `false`	
P38	⟨*Literal*⟩ → ⟨*String*⟩	
P39		
P40		
P41	⟨*Var*⟩ → ⟨*Identifier*⟩	
P42	⟨*Num*⟩ → `([0-9])`+	
P43	⟨*String*⟩ → `"` `([' '-'~'])*` `"`	
P44	⟨*Identifier*⟩ → `[a-zA-Z]([a-zA-Z0-9'_'])*`	

7.3 LiveOak-2: Procedural Programming

Id	Production	Comments
P1		
P2	⟨*Program*⟩ → (⟨*MethodDecl*⟩)*	Start here.
P3		Not here.
P4		
P5	⟨*MethodDecl*⟩ → ⟨*Type*⟩ ⟨*MethodName*⟩ `(` (⟨*Formals*⟩)? `)` `{` ⟨*Body*⟩ `}`	Method declaration.
P6	⟨*Body*⟩ → (⟨*VarDecl*⟩)* ⟨*Block*⟩	
P7	⟨*Formals*⟩ → ⟨*Type*⟩ ⟨*Identifier*⟩ (`,` ⟨*Type*⟩ ⟨*Identifier*⟩)*	Formal parameters in a method declaration.
P8	⟨*Actuals*⟩ → ⟨*Expr*⟩ (`,` ⟨*Expr*⟩)*	Actual parameters in a method invocation.
P9	⟨*VarDecl*⟩ → ⟨*Type*⟩ ⟨*Identifier*⟩ (`,` ⟨*Identifier*⟩)* `;`	
P10	⟨*Block*⟩ → `{` (⟨*Stmt*⟩)+ `}`	
P11	⟨*Stmt*⟩ → `return` ⟨*Expr*⟩ `;`	Return a result value from a method.
P12	⟨*Stmt*⟩ → `if` `(` ⟨*Expr*⟩ `)` ⟨*Block*⟩ `else` ⟨*Block*⟩	
P13	⟨*Stmt*⟩ → `while` `(` ⟨*Expr*⟩ `)` ⟨*Block*⟩	
P14	⟨*Stmt*⟩ → `break` `;`	
P15	⟨*Stmt*⟩ → ⟨*Var*⟩ `=` ⟨*Expr*⟩ `;`	
P16	⟨*Stmt*⟩ → `;`	
P17		
P18		
P19		
P20		
P21	⟨*Expr*⟩ → ⟨*MethodName*⟩ `(` (⟨*Actuals*⟩)? `)`	Invoke a method.
P22	⟨*Expr*⟩ → `(` ⟨*Expr*⟩ `?` ⟨*Expr*⟩ `:` ⟨*Expr*⟩ `)`	
P23	⟨*Expr*⟩ → `(` ⟨*Expr*⟩ ⟨*Binop*⟩ ⟨*Expr*⟩ `)`	
P24	⟨*Expr*⟩ → `(` ⟨*Unop*⟩ ⟨*Expr*⟩ `)`	
P25	⟨*Expr*⟩ → `(` ⟨*Expr*⟩ `)`	
P26	⟨*Expr*⟩ → ⟨*Var*⟩	
P27	⟨*Expr*⟩ → ⟨*Literal*⟩	
P28	⟨*Binop*⟩ → `[+-*/%&|<>=]`	
P29	⟨*Unop*⟩ → `[~!]`	
P30		
P31		
P32	⟨*Type*⟩ → `int`	
P33	⟨*Type*⟩ → `bool`	
P34	⟨*Type*⟩ → `String`	
P35	⟨*Literal*⟩ → ⟨*Num*⟩	
P36	⟨*Literal*⟩ → `true`	
P37	⟨*Literal*⟩ → `false`	
P38	⟨*Literal*⟩ → ⟨*String*⟩	
P39		
P40	⟨*MethodName*⟩ → ⟨*Identifier*⟩	
P41	⟨*Var*⟩ → ⟨*Identifier*⟩	
P42	⟨*Num*⟩ → `([0-9])+`	
P43	⟨*String*⟩ → `"` `([' '-'~'])*` `"`	
P44	⟨*Identifier*⟩ → `[a-zA-Z]([a-zA-Z0-9'_'])*`	

7.4 LiveOak-3: Objects and Classes, No Inheritance

Id	Production	Comments
P1	⟨*Program*⟩ → (⟨*ClassDecl*⟩)*	Start here.
P2		Not here.
P3		
P4	⟨*ClassDecl*⟩ → `class` ⟨*ClassName*⟩ `(` (⟨*VarDecl*⟩)* `)` `{` (⟨*MethodDecl*⟩)* `}`	Class declaration.
P5	⟨*MethodDecl*⟩ → ⟨*Type*⟩ ⟨*MethodName*⟩ `(` (⟨*Formals*⟩)? `)` `{` ⟨*Body*⟩ `}`	
P6	⟨*Body*⟩ → (⟨*VarDecl*⟩)* ⟨*Block*⟩	
P7	⟨*Formals*⟩ → ⟨*Type*⟩ ⟨*Identifier*⟩ (`,` ⟨*Type*⟩ ⟨*Identifier*⟩)*	
P8	⟨*Actuals*⟩ → ⟨*Expr*⟩ (`,` ⟨*Expr*⟩)*	
P9	⟨*VarDecl*⟩ → ⟨*Type*⟩ ⟨*Identifier*⟩ (`,` ⟨*Identifier*⟩)* `;`	
P10	⟨*Block*⟩ → `{` (⟨*Stmt*⟩)+ `}`	
P11	⟨*Stmt*⟩ → `return` ⟨*Expr*⟩ `;`	
P12	⟨*Stmt*⟩ → `if` `(` ⟨*Expr*⟩ `)` ⟨*Block*⟩ `else` ⟨*Block*⟩	
P13	⟨*Stmt*⟩ → `while` `(` ⟨*Expr*⟩ `)` ⟨*Block*⟩	
P14	⟨*Stmt*⟩ → `break` `;`	
P15	⟨*Stmt*⟩ → ⟨*Var*⟩ `=` ⟨*Expr*⟩ `;`	
P16	⟨*Stmt*⟩ → `;`	
P17	⟨*Expr*⟩ → `this`	Reference to current object.
P18	⟨*Expr*⟩ → `null`	A null object.
P19	⟨*Expr*⟩ → `new` ⟨*ClassName*⟩ `(` (⟨*Actuals*⟩)? `)`	Create a new object.
P20	⟨*Expr*⟩ → ⟨*Var*⟩`.` ⟨*MethodName*⟩ `(` (⟨*Actuals*⟩)? `)`	Invoke an instance method.
P21		No static methods.
P22	⟨*Expr*⟩ → `(` ⟨*Expr*⟩ `?` ⟨*Expr*⟩ `:` ⟨*Expr*⟩ `)`	
P23	⟨*Expr*⟩ → `(` ⟨*Expr*⟩ ⟨*Binop*⟩ ⟨*Expr*⟩ `)`	
P24	⟨*Expr*⟩ → `(` ⟨*Unop*⟩ ⟨*Expr*⟩ `)`	
P25	⟨*Expr*⟩ → `(` ⟨*Expr*⟩ `)`	
P26	⟨*Expr*⟩ → ⟨*Var*⟩	
P27	⟨*Expr*⟩ → ⟨*Literal*⟩	
P28	⟨*Binop*⟩ → `[+-*/%&|<>=]`	
P29	⟨*Unop*⟩ → `[~!]`	
P30	⟨*Type*⟩ → `void`	Return type of constructor.
P31	⟨*Type*⟩ → ⟨*ClassName*⟩	User-defined type.
P32	⟨*Type*⟩ → `int`	
P33	⟨*Type*⟩ → `bool`	
P34	⟨*Type*⟩ → `String`	
P35	⟨*Literal*⟩ → ⟨*Num*⟩	
P36	⟨*Literal*⟩ → `true`	
P37	⟨*Literal*⟩ → `false`	
P38	⟨*Literal*⟩ → ⟨*String*⟩	
P39	⟨*ClassName*⟩ → ⟨*Identifier*⟩	
P40	⟨*MethodName*⟩ → ⟨*Identifier*⟩	
P41	⟨*Var*⟩ → ⟨*Identifier*⟩	
P42	⟨*Num*⟩ → `([0-9])+`	
P43	⟨*String*⟩ → `"` `([' '-'~'])*` `"`	
P44	⟨*Identifier*⟩ → `[a-zA-Z]([a-zA-Z0-9'_'])*`	