Brian Luong Compiler

Scope:
My project is implemented with everything specified from PA1 – PA4, except for short circuit evaluation from PA4. I have also implemented two additional features: static field initialization and class constructors with parameters.

Summary of changes to AST:
Added declaration instance field in Identifiers
Added UnsupportedType AST class for unsupported types
Added ErrorType AST class for producing error types in Type Checking
Added ConstructorType AST class to support constructor methods
Added MethodDecl instance field in NewObjExpr AST to support constructors
Modified newObjExpr to add an ExprList (argument list) instance field to support constructor
Added ConstructorMethodDecl AST class (subclass of MethodDecl) to support constructors
Added FieldDeclStmt (subclass of FieldDecl) to support static field initialization

Static initialization special rules:
Static field initialization is done before the main method is called in Code Generation. My compiler runs through initialization before the main method is called.
Cannot refer to an instance variable (in the same class) before it is defined. Order matters. My compiler checks for this.

Constructor special rules:
Cannot have return types.
Cannot have more than one of the same number (and type) or arguments constructor per class.

Tests:
Passes the bubble sort test program. I added additional tests to test the additional static initialization and constructor features.

Static Initialization Tests:
Pass 1. Integer initialization
Pass 2. Qualified reference
Pass 3. Reference to previous defined static field in same class
Pass 4. Qualified reference mixed with Pass 3
Pass 5. Initialize a new object
Pass 6. Qualified reference to static variable in another class
Pass 7. Initialization from a static method

Fail 1. Reject instance field initialization (only allow static)
Fail 2. Referring to a field defined later in the class

Fail 3. Using non-static method to initialize variable

Constructor Tests:
Pass 1. Constructor with integer parameter
Pass 2. Constructor with Qualified expression parameter
Pass 3. Allow multiple constructors with same number of arguments but different type

Fail 1. Constructor called without the correct number of arguments
Fail 2. Multiple constructor with same name, same type.
Fail 3. Constructor with return type