Pseodocode AST to LLVM:

createVtable(classDecl):

1. res <- “@.classDecl_vtable = global [“ + numberOfMethods + ” x i8*] [\n”
2. allMethods <- classDecl.methods() ∪ inheritedMethods
3. **for** i <- 0 to allMethods.size() - 2:
4. res <- res.concat(“i8* bitcast (“ + methodReturnType+ ” (i8*” + methodargs + ”)* @classDecl.” + allMethods[i] + ” to i8*),\n”)
5. res <- res.concat(“i8* bitcast (“ + methodReturnType+ ” (i8*” + methodargs + ”)* @classDecl.” + allMethods[allMethods.size() - 1] + ” to i8*),\n”)
6. **return** res

implementMethod(methodDecl):

1. res <- “define “ + methodReturnType + ” @classDecl.methodDecl (i8* %this” + methodargs +”) {\n”
2. for statement in methodDecl.body():
3. res <- res.concat(“\t” + implementStatement(statement) + “\n”)
4. res <- res.concat(implementReturnExpr(methodDecl.ret() + “\n}”)

Program to llvm:

1. res <- init string
2. **for** classDecl in classDecls
3. res <- res.concat(createVtable(classDecl) + “\n”)
4. res <- res.concat(calloc etc.)
5. **for** classDecl in classDecls ∪ [MainClass]
6. **for** methodDecl in classDecl
7. res <- res.concat(implementMethod(methodDecl) + “\n”)
8. **return** res