/Users/jchavez/dev/java.net/jgsl.dev.java.net/jgsl/dev/src/jgsl/JGSL.java

```java
    /*
 * Copyright (c) 2005 Perception Software. All Rights Reserved.
 */
package jgsl;


import org.apache.log4j.BasicConfigurator;
import org.apache.log4j.Logger;
import org.apache.log4j.PropertyConfigurator;

import java.io.File;
import java.net.URL;

import jgsl.controller.script.ScriptEngine;

/**
 * JGSL program main class. This is main entry point for executing the JGSL in both command line and GUI modes. This
 * class configures 2 loggers using the Log4J API. The Log4J properties file is bundled with the jgsl.jar
 * file and is located in the <pre>jgsl/resources/jgsl_log.prop</pre> file.
 * <p/>
 * After the loggers are configured the control flow is passed to the ScriptEngine class.
 *
 * @author zenarchitect
 * @version $Id: JGSL.java,v 1.8 2005/05/16 00:54:23 zenarchitect Exp $
 */
public class JGSL {
    static Logger jgslLogger;
    static Logger sysLogger;

    /**
     * @link aggregationByValue
     * @directed
     */
    /*# ScriptEngine lnkScriptEngine; */

            public static void main(String[] args) {
        try {
            File logFileDir = new File(System.getProperty("user.home") + File.separator + ".jgsl" + File.separator +
"logs");
            logFileDir.mkdirs();
            URL props = Thread.currentThread().getContextClassLoader().getResource("jgsl/resources/jgsl_log.prop");
            if (props != null) {
                PropertyConfigurator.configure(props);
            } else {
                BasicConfigurator.configure();
            }
            jgslLogger = Logger.getLogger("jgsl_log");
            sysLogger = Logger.getLogger("jgsl_sys_log");

            jgslLogger.info("Starting JGSL");
            ScriptEngine se = new ScriptEngine();
            se.processCommandLine(args);
            jgslLogger.info("Ending JGSL");

        } catch (Throwable t) {
            jgslLogger.error("JGSL Error\n", t);
        }

    }
}
```

/Users/jchavez/dev/java.net/jgsl.dev.java.net/jgsl/dev/src/jgsl/controller/script/ScriptEngine.java

```
/*
 * Copyright (c) 2005 Perception Software. All Rights Reserved.
 */
package jgsl.controller.script;

import org.apache.commons.cli.CommandLine;
import org.apache.commons.cli.CommandLineParser;
import org.apache.commons.cli.HelpFormatter;
import org.apache.commons.cli.Option;
import org.apache.commons.cli.Options;
import org.apache.commons.cli.ParseException;
import org.apache.commons.cli.PosixParser;
import org.apache.log4j.Level;
import org.apache.log4j.Logger;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Arrays;
import java.util.ResourceBundle;

import jgsl.io.ScriptParser;
import jgsl.io.ScriptParserException;
import jgsl.model.JGSLScript;
import jgsl.util.JarPackager;
import jgsl.util.JarPackagerException;
import jgsl.view.swing.JGSLSwingFrame;
import jgsl.view.swing.SwingScriptViewer;

/**
 * The ScriptEngine class is the controller for the JGSL application. It contains the command line processor for the
 * command console interface. The *Interactive methods are the controller interfaced for the Interactive GUI.
 *
 * @author zenarchitect
 * @version $Id: ScriptEngine.java,v 1.8 2005/05/21 01:42:06 zenarchitect Exp $
 */
public class ScriptEngine {
    static Logger jgslLogger = Logger.getLogger("jgsl_log");
    static Logger sysLogger = Logger.getLogger("jgsl_sys_log");

    // create Options object
    private Options options = new Options();

    /** @link dependency */
    /*# JGSLScript lnkJGSLScript; */

    /** @link dependency */
    /*# ScriptEngineException lnkScriptEngineException; */

    /** @link aggregationByValue
     * @directed*/
    /*# ScriptParser lnkScriptParser; */

    /** @link aggregationByValue
     * @directed*/
    /*# SwingScriptViewer lnkSwingScriptViewer; */

    /** @directed
     * @link aggregation*/
    /*# JGSLScript lnkJGSLScript1; */

    /**
     * Parse the script supplied in fileName and return the JGSLScript containing the JGSL object model for the script.
     *
     * @param fileName Name of the JGSL script file
     * @return JGSLScript object containing the script object model
     * @throws ScriptParserException If a problem is encountered during parsing a ScriptParser exception will be
     *                               thrown.
     * @see jgsl.model.JGSLScript
     */
    public JGSLScript parseInteractive(File fileName) throws ScriptParserException {
        ScriptParser sp = new ScriptParser();
        JGSLScript script = sp.execScript(fileName);
        return script;
    }

    /**
     * This method will parse the script contained in fileName and then display the result in the JGSL viewer.
     *
     * @param fileName Name of the JGSL script file
     * @return JGSLScript object containing the script object model
     * @throws ScriptParserException If a problem is encountered during parsing a ScriptParser exception will be
     *                               thrown.
     * @see jgsl.model.JGSLScript
     */
    public JGSLScript viewInteractive(File fileName, String saveToFileName) throws ScriptParserException {
        JGSLScript script = parseInteractive(fileName);
        String fullClassName = script.generateImplementation();
        if (fullClassName != null) {
            SwingScriptViewer ssv = new SwingScriptViewer();
            ssv.renderScript(fullClassName, saveToFileName);
        } else {
            sysLogger.debug("Unable to generated implementation, full class name for file: " + fileName.getAbsolutePath
());
            throw new ScriptParserException("Unable to generated implementation, full class name for file: " + fileName.
getAbsolutePath());
        }
        return script;
```

```java
    }

    /**
     * The method will parse the script contained in scriptFileName and then create an executable JAR file with name of
     * jarFileName containing the Java class for the JGSL script.
     *
     * @param scriptFileName Name of JGSL script to create the JAR from.
     * @param jarFileName    Name of JAR file to generate
     * @return JGSLScript object containing the script object model
     * @throws ScriptParserException If a problem is encountered during parsing a ScriptParser exception will be
     *                               thrown.
     * @throws ScriptEngineException If a problem occurs during the creation of the JAR file.
     * @see jgsl.model.JGSLScript
     */
    public JGSLScript jarInteractive(File scriptFileName, File jarFileName) throws ScriptEngineException,
ScriptParserException {
        JGSLScript script = parseInteractive(scriptFileName);
        script.generateImplementation();
        try {
            JarPackager.makeJar(jarFileName, script.getClassFileName(), script.getFullClassName());
        } catch (JarPackagerException e) {
            sysLogger.error(e.getMessage(), e);
            throw new ScriptEngineException("Unable to generated JAR: " + scriptFileName.getAbsolutePath());
        }
        return script;
    }

    /**
     * Process the command line arguments and perform the requested actions. The set of available options is listed
     * below.
     * <p/>
     * <pre>
     * usage: jgsl.JGSL
     *      -d,--doc jgsl script filt script doc file   Generate script documenation
     *      -e,--exec script file                       Execute the script file
     *      -h,--help                                   Print this message
     *      -j,--jar jgsl script file JAR file          Generate JAR file for script
     *      -l,--logLevel user log level                Set user logging level
     *                                                  to one of: LOG, DEBUG, ERROR, WARNING
     *      -p,--parse script file                      Parse the script file and print the results
     *      -s,--sysLogLevel system log level           Set system logging
     *                                                  level to one of: LOG, DEBUG, ERROR, WARNIN
     *      -v,--view Type of viewer, supports: swing   Parse, execute and view the script
     * </pre>
     *
     * @param args Array of String references to valid arguments
     * @throws ScriptParserException If parsing is requested and problem is found this script this exception will be
     *                               thrown.
     * @throws ScriptEngineException If execution or JAR is requested and a problem occurs this exception will be
     *                               thrown.
     */
    public void processCommandLine(String[] args) throws ScriptParserException, ScriptEngineException {
        ResourceBundle res = ResourceBundle.getBundle("jgsl.resources.JGSL");

        sysLogger.info("JGSL arguments: " + Arrays.toString(args));

        // create the command line parser
        CommandLineParser parser = new PosixParser();

        // create the Options

        // help option
        Option help = new Option(res.getString("app.option.help.short"), res.getString("app.option.help.long"), false, res.getString("app.option.help.message"));
        options.addOption(help);

        // log level script option
        Option logLevel = new Option(res.getString("app.option.loglevel.short"), res.getString("app.option.loglevel.long"), true, res.getString("app.option.loglevel.message"));
        logLevel.setArgName(res.getString("app.option.loglevel.level"));
        options.addOption(logLevel);

        // log level script option
        Option sysLogLevel = new Option(res.getString("app.option.sysloglevel.short"), res.getString("app.option.sysloglevel.long"), true, res.getString("app.option.sysloglevel.message"));
        sysLogLevel.setArgName(res.getString("app.option.sysloglevel.level"));
        options.addOption(sysLogLevel);

        // parse script option
        Option parseScript = new Option(res.getString("app.option.parsescript.short"), res.getString("app.option.parsescript.long"), true, res.getString("app.option.parsescript.message"));
        parseScript.setArgName(res.getString("app.option.parsescript.filearg"));
        options.addOption(parseScript);

        // execute script option
        Option execScript = new Option(res.getString("app.option.execscript.short"), res.getString("app.option.execscript.long"), true, res.getString("app.option.execscript.message"));
        execScript.setArgName(res.getString("app.option.execscript.filearg"));
        options.addOption(execScript);

        // view script option
        Option viewScript = new Option(res.getString("app.option.viewscript.short"), res.getString("app.option.viewscript.long"), true, res.getString("app.option.viewscript.message"));
        viewScript.setArgName(res.getString("app.option.viewscript.viewertype"));
        viewScript.setOptionalArg(true);
        options.addOption(viewScript);

        // save to file
        Option saveToFile = new Option(res.getString("app.option.savetofiletype.short"), res.getString("app.option.savetofiletype.long"), true, res.getString("app.option.savetofiletype.message"));
        saveToFile.setArgName(res.getString("app.option.savetofiletype.type"));
        options.addOption(saveToFile);

        // generate docs script option
        Option genDoc = new Option(res.getString("app.option.gendoc.short"), res.getString("app.option.gendoc.long"), 
true, res.getString("app.option.gendoc.message"));
        genDoc.setArgs(2);
        genDoc.setArgName(res.getString("app.option.gendoc.files"));
        options.addOption(genDoc);

        // generate JAR option
        Option genJar = new Option(res.getString("app.option.genjar.short"), res.getString("app.option.genjar.long"), 
true, res.getString("app.option.genjar.message"));
        genJar.setArgs(2);
        genJar.setArgName(res.getString("app.option.genjar.files"));
        options.addOption(genJar);

        String saveToFileName = null;
        // parse the command line arguments
        CommandLine line = null;
        try {
            line = parser.parse(options, args);
        } catch (ParseException e) {
            throw new ScriptEngineException(new StringBuffer().append("app.exception.program.args\n").append(e.
getMessage()).toString());
        }

        if (line.hasOption(help.getOpt())) {
            // automatically generate the help statement
            HelpFormatter formatter = new HelpFormatter();
            formatter.printHelp(res.getString("app.command.line.name"), options);
            return;
        }

        if (line.hasOption(logLevel.getOpt())) {
            String level = line.getOptionValue(logLevel.getOpt());
            jgslLogger.setLevel(Level.toLevel(level));
        }

        if (line.hasOption(sysLogLevel.getOpt())) {
            String level = line.getOptionValue(sysLogLevel.getOpt());
            sysLogger.setLevel(Level.toLevel(level));
        }

        if (line.hasOption(parseScript.getOpt())) {
            ScriptParser sp = new ScriptParser();
            String fileName = line.getOptionValue(parseScript.getOpt());
            String result = sp.parseScript(new File(fileName));
            jgslLogger.info(result);
            return;
        }

        if (line.hasOption(execScript.getOpt())) {
            ScriptParser sp = new ScriptParser();
            String fileName = line.getOptionValue(execScript.getOpt());
            JGSLScript script = sp.execScript(new File(fileName));

            jgslLogger.info(script.getParseStatus());

            if (script.hasErrors()) {
                return;
            }

            String fullClassName = script.generateImplementation();
            if (line.hasOption(viewScript.getOpt())) {
// TODO: currently only swing is supported so the type is not checked here
                SwingScriptViewer ssv = new SwingScriptViewer();
                if (line.hasOption(saveToFile.getOpt())) {
                    saveToFileName = line.getOptionValue(saveToFile.getOpt());
                }
                ssv.renderScript(fullClassName, saveToFileName);
            }
            return;
        }

        if (line.hasOption(genDoc.getOpt())) {
            ScriptParser sp = new ScriptParser();
            String filenames[] = line.getOptionValues(genDoc.getOpt());
            JGSLScript script = sp.execScript(new File(filenames[0]));
            jgslLogger.info(script.getParseStatus());

            if (script.hasErrors()) {
                System.exit(1);
            }
            String docStr = script.getDocumentation();
            File docFile = new File(filenames[1]);
            try {
                FileWriter fw = new FileWriter(docFile);
                fw.write(docStr);
                fw.close();
            } catch (FileNotFoundException e) {
                jgslLogger.error(e.getMessage());
                sysLogger.error(e.getMessage(), e);
                System.exit(1);
            } catch (IOException e) {
                jgslLogger.error(e.getMessage());
                sysLogger.error(e.getMessage(), e);
                System.exit(1);
            }
            return;
        }


        if (line.hasOption(genJar.getOpt())) {
            ScriptParser sp = new ScriptParser();
            String filenames[] = line.getOptionValues(genJar.getOpt());
            JGSLScript script = sp.execScript(new File(filenames[0]));
            jgslLogger.info(script.getParseStatus());

            if (script.hasErrors()) {
                System.exit(1);
            }
            script.generateImplementation();
            try {
                JarPackager.makeJar(new File(filenames[1]), script.getClassFileName(), script.getFullClassName());
            } catch (JarPackagerException e) {
                jgslLogger.error(e.getMessage());
                sysLogger.error(e.getMessage(), e);
                System.exit(1);
            }
            return;
        }

        // if we get here then show the GUI
        JGSLSwingFrame.startJGSL(args);
    }

}

```

/Users/jchavez/dev/java.net/jgsl.dev.java.net/jgsl/dev/src/jgsl/io/ScriptParser.java

```java
    /*
 * Copyright (c) 2005 Perception Software. All Rights Reserved.
 */
package jgsl.io;

import java.io.File;

import jgsl.model.JGSLScript;
import jgsl.parser.JGSL_Parser;
import jgsl.parser.ParseException;

/**
 * Parse the specified script file using the JGSL_Parser and report the status of the parse.
 *
 * @author zenarchitect
 * @version $Id: ScriptParser.java,v 1.6 2005/05/16 00:54:16 zenarchitect Exp $
 */

public class ScriptParser {
    /**
     * @link aggregationByValue
     */
    JGSL_Parser parser;

    /** @link dependency */
    /*# ScriptParserException lnkScriptParserException; */

    /** @link dependency */
    /*# Message lnkMessage; */

    /**
     * @directed
     * @link aggregation
     */
    /*# JGSLScript lnkJGSLScript; */

        public JGSLScript execScript(File scriptFile) throws ScriptParserException {
        parseScript(scriptFile);
        if (parser == null) {
            throw new ScriptParserException("jgsl parser: Execution failed, unable to create script parser.");
        }
        JGSLScript script = parser.getScript();
        script.setScriptName(scriptFile.getName());
        return script;
    }

    public String parseScript(File scriptFile) throws ScriptParserException {
        String result = "jgsl parser:  Reading from file " + scriptFile + " . . .\n";
        double initTime = 0;
        double parseTime = 0;
        double startTime = 0;
        double stopTime = 0;
        try {
            startTime = (double) System.currentTimeMillis();
            parser = new JGSL_Parser(new java.io.FileInputStream(scriptFile));
            stopTime = (double) System.currentTimeMillis();
            initTime = (double) stopTime - startTime;
        } catch (java.io.FileNotFoundException e) {
            if (parser != null) {
                JGSLScript script = parser.getScript();
                script.addError(new ScriptError(result + "\njgsl parser:  File " + scriptFile + " not found.\n"));
            } else {
                throw new ScriptParserException("jgsl parser: Execution failed, unable to create script 
parser.");
            }
        }
        try {
            startTime = (double) System.currentTimeMillis();
            parser.parseScript();
            stopTime = (double) System.currentTimeMillis();
            parseTime = stopTime - startTime;
            result += "jgsl parser:\n";
            result += "\tparsed " + scriptFile + " in " + (initTime + parseTime) / 1000.0 + " sec\n";
            result += "\tinitialization time = " + initTime / 1000.0 + " sec\n";
            result += "\tparse time = " + parseTime / 1000.0 + " sec\n";
            JGSLScript script = parser.getScript();
            script.addMessage(new ScriptMessage(result));
        } catch (ParseException e) {
            String ex = "jgsl parser:  Reading from file " + scriptFile + " . . .\n"
                    + "jgsl parser:  Encountered errors during parse...\n";
            ex += e.getMessage();
            if (parser != null) {
                JGSLScript script = parser.getScript();
                script.addError(new ScriptError(ex));
            } else {
                throw new ScriptParserException(ex);
            }
        }
        return result;
    }
```

```

}


```

/Users/jchavez/dev/java.net/jgsl.dev.java.net/jgsl/dev/src/jgsl/io/ScriptParserException.java

```
/*
 * Copyright (c) 2005 Perception Software. All Rights Reserved.
 */
package jgsl.io;

/**
 * Report script parsing exceptions.
 *
 * @author Joe Chavez
 * @version $Id: ScriptParserException.java,v 1.2 2005/05/16 00:54:16 zenarchitect Exp $
 */
public class ScriptParserException extends Exception {
    /**
     * Constructs a new exception with the specified detail message.  The cause is not initialized, and may subsequently
     * be initialized by a call to {@link #initCause}.
     *
     * @param message the detail message. The detail message is saved for later retrieval by the {@link #getMessage()}
     *                method.
     */
    public ScriptParserException(String message) {
        super(message);
    }

}


```

/Users/jchavez/dev/java.net/jgsl.dev.java.net/jgsl/dev/src/jgsl/model/JGSLScript.java

```
/*
 * Copyright (c) 2005 Perception Software. All Rights Reserved.
 */

package jgsl.model;
import javassist.CannotCompileException;
import javassist.ClassClassPath;
import javassist.ClassPool;
import javassist.CtClass;
import javassist.CtConstructor;
import javassist.CtMethod;
import javassist.NotFoundException;
import org.apache.log4j.Logger;

import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.Serializable;
import java.util.ArrayList;
import java.util.LinkedList;
import java.util.ResourceBundle;

import jgsl.io.Message;
import jgsl.io.ParseStatus;
import jgsl.io.ScriptError;
import jgsl.io.ScriptMessage;
import jgsl.io.ScriptWarning;

/**
 * A JGSLScript contains an ordered colllection of objects that implement the statement interface.
 *
 * @author zenarchitect
 * @version $Id: JGSLScript.java,v 1.6 2005/05/16 00:54:18 zenarchitect Exp $
 */
public class JGSLScript implements Serializable, Script, ParseStatus {
    static Logger jgslLogger = Logger.getLogger("jgsl_log");
    static Logger sysLogger = Logger.getLogger("jgsl_sys_log");

    ResourceBundle res = ResourceBundle.getBundle("jgsl.view.swing.resources.BaseFrame");

    /**
     * @link aggregationByValue
     * @supplierCardinality 1..*
     * @clientCardinality 1
     * @label is composed of
     */
    /*# Statement lnkStatement; */

    private LinkedList<Statement> statements = new LinkedList<Statement>();

    private ArrayList<Message> messages = new ArrayList<Message>();
    int errorCount = 0;
    int warningCount = 0;
    int messageCount = 0;

    private String scriptName;
    private Documentation doc;

    private String className;
    private String fullClassName;
    private String classFileName;


    public JGSLScript() {
    }

    public String getClassName() {
        return className;
    }

    public String getFullClassName() {
        return fullClassName;
    }

    public String getClassFileName() {
        return classFileName;
    }

    /**
     * Get the script name
     *
     * @return String containing the script name
     */
    public String getScriptName() {
        return scriptName;
    }

    /**
     * Set the scipt name
     *
     * @param scriptName name of the script file
     */
    public void setScriptName(String scriptName) {
        this.scriptName = scriptName;
    }
```

```java

    public String getJavaForInit() {
        StringBuffer strBuff = new StringBuffer(1024);

        for (Statement s : statements) {
            if (s.getType().equals(Commands.CANVAS.getName())) {
                strBuff.append(s.getJava());
            }
        }
        String title = res.getString("jgsl.title") + " - " + scriptName;
        strBuff.append("setTitle(\"" + title + "\");");

        return strBuff.toString();
    }

    /**
     * Return the Java implementation of this script
     *
     * @return the Java language implementation of this script
     */
    public String getJava() {
        StringBuffer strBuff = new StringBuffer(1024);

        strBuff.append("super.paint(g);");
        strBuff.append("java.awt.Graphics2D g2 = (java.awt.Graphics2D) g;");
        strBuff.append("java.awt.Container canvas = this.getContentPane();");

        for (Statement s : statements) {
            if (s.getType().equals(Commands.CANVAS.getName())) {
                continue;
            }
            strBuff.append(s.getJava());
        }

        return strBuff.toString();
    }

    public void addDocumentation(String d) {
        if (doc == null) {
            doc = new Documentation();
        }
        if (d.startsWith("\"")) {
            d = d.substring(1);
        }
        if (d.endsWith("\"")) {
            d = d.substring(0, d.length() - 1);
        }
        if (!d.endsWith("\n")) {
            d += "\n";
        }
        doc.addDoc(d);
    }

    /**
     * Returns the JGSL script documentation as specified in the DOC keyword by the script author.
     *
     * @return The script documentation
     */
    public String getDocumentation() {
        return doc.getJava();
    }

    /**
     *
     */
    public void add(Statement s) {
        statements.add(s);
    }

    /**
     * Generate the implementation class and return the name of the class
     *
     * @return returns a String containing the full name of the implementation class
     */
    public String generateImplementation() {

        className = scriptName.substring(scriptName.lastIndexOf("/") + 1, scriptName.lastIndexOf("."));
        fullClassName = "jgsl.generated." + className;
//TODO: get destination dir from command line or config file
        String dirName = System.getProperty("user.home") + File.separator + ".jgsl" + File.separator + "cache";

        File dir = new File(dirName);
        if (!dir.exists()) {
            dir.mkdirs();
        }
        // now check for the jar files
        // if not there then write them from the resources
        File jgslJar = new File(dir, "jgsl_rt.jar");
        try {
            InputStream is = Thread.currentThread().getContextClassLoader().getResourceAsStream("lib/jgsl_rt.jar");
            BufferedInputStream bis = new BufferedInputStream(is);
            FileOutputStream fos = new FileOutputStream(jgslJar);
            BufferedOutputStream bos = new BufferedOutputStream(fos);
            byte buff[] = new byte[1024];
            int bytesRead = 0;
            while ((bytesRead = bis.read(buff)) != -1) {
                bos.write(buff, 0, bytesRead);
            }
            bis.close();
            bos.close();
        } catch (IOException e) {
            e.printStackTrace();  //TODO handle exception
        }

        File log4Jar = new File(dir, "log4j-1.2.9.jar");
        try {
            InputStream is = Thread.currentThread().getContextClassLoader().getResourceAsStream("lib/log4j-1.2.9.
jar");
            BufferedInputStream bis = new BufferedInputStream(is);
            FileOutputStream fos = new FileOutputStream(log4Jar);
            BufferedOutputStream bos = new BufferedOutputStream(fos);
            byte buff[] = new byte[1024];
            int bytesRead = 0;
            while ((bytesRead = bis.read(buff)) != -1) {
                bos.write(buff, 0, bytesRead);
            }
            bis.close();
            bos.close();
        } catch (IOException e) {
            e.printStackTrace();  //TODO handle exception
        }

        File f = new File(dirName + File.separator + fullClassName);
        if (f.exists()) {
            f.delete();
        }

        ClassPool pool = ClassPool.getDefault();
        pool.insertClassPath(new ClassClassPath(this.getClass()));
        CtClass cc = null;
        try {
            cc = pool.get("jgsl.view.swing.BaseFrame");
            cc.setName(fullClassName);

            // modify the constructor
            CtConstructor cd[] = cc.getDeclaredConstructors();

            String initStr = getJavaForInit();
            cd[0].insertBeforeBody(initStr);

            // modify the paint method
            CtMethod m = cc.getDeclaredMethod("paint");

            String paintStr = getJava();
            m.insertAfter("{" +
                    paintStr +
                    "}");

            cc.writeFile(dirName);
            cc.defrost();
            classFileName = dirName + File.separator + fullClassName.replace('.', '/');
            return fullClassName;

```

```java
        } catch (NotFoundException e) {
            e.printStackTrace();  //TODO: To change body of catch statement use File | Settings | File Templates.
        } catch (IOException e) {
            e.printStackTrace();  //TODO: To change body of catch statement use File | Settings | File Templates.
        } catch (CannotCompileException e) {
            e.printStackTrace();  //TODO: To change body of catch statement use File | Settings | File Templates.
        }
        return null;
    }


    public String toString() {
        return "JGSLScript{" +
                "statements=" + statements +
                ", scriptName='" + scriptName + "'" +
                "}";
    }

    /**
     * Add a ScriptError to the parse status
     *
     * @param se
     */
    public void addError(ScriptError se) {
        messages.add(se);
        errorCount++;
    }

    /**
     * Add a ScriptWarning to the parse status
     *
     * @param sw
     */
    public void addWarning(ScriptWarning sw) {
        messages.add(sw);
        warningCount++;
    }

    /**
     * Add a ScriptMessage to the parse status
     *
     * @param sm
     */
    public void addMessage(ScriptMessage sm) {
        messages.add(sm);
        messageCount++;
    }


    /**
     * Return the error state of the script
     *
     * @return true of the script contains errors or false otherwise
     */
    public boolean hasErrors() {
        return errorCount > 0;
    }

    /**
     * Return the warning state of the script
     *
     * @return true of the script contains warnings or false otherwise
     */
    public boolean hasWarnings() {
        return warningCount > 0;
    }

    /**
     * Return the message state of the script * @return true of the script contains messages or false otherwise
     */
    public boolean hasMessages() {
        return messageCount > 0;
    }

    public int getErrorCount() {
        return errorCount;
    }

    public int getWarningCount() {
        return warningCount;
    }

    public int getMessageCount() {
        return messageCount;
    }

    public String getParseStatus() {
        String status = "Parse Status\n";

        for (Message m : messages) {
            status += m.getMessage();
        }

        String messageString = "messages";
        String warningString = "warnings";
        String errorString = "errors";
        if (messageCount == 1) {
            messageString = "message";
        }
        if (warningCount == 1) {
            warningString = "warning";
        }
        if (errorCount == 1) {
            errorString = "error";
        }

        status += String.format("%d %s, %d %s, %d %s encountered during script processing.\n", messageCount, messageString, warningCount, warningString, errorCount, errorString);
        if (hasErrors()) {
            status += "Please examine and correct any errors.";
        }
        return status;
    }
}

```

/Users/jchavez/dev/java.net/jgsl.dev.java.net/jgsl/dev/src/jgsl/util/JarPackager.java

```java
/*
 * Copyright (c) 2005 Perception Software. All Rights Reserved.
 */
package jgsl.util;

import org.apache.log4j.Logger;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.ResourceBundle;
import java.util.jar.Attributes;
import java.util.jar.JarEntry;
import java.util.jar.JarInputStream;
import java.util.jar.JarOutputStream;
import java.util.jar.Manifest;

/**
 * Create an executable JAR file for a JGSL script Java class.
 *
 * @author jchavez
 */
public class JarPackager {
    static Logger jgslLogger = Logger.getLogger("jgsl_log");
    static Logger sysLogger = Logger.getLogger("jgsl_sys_log");

    /**
     * Create a jar file for JGSL distribution. The className parameter will be used to set the main class attribute.
     * <p/>
     * Main-Class: className
     *
     * @param jarFileName   Name of JAR to create
     * @param classFileName Full path to the class file to add to the jar
     * @param className     Name of the class with full package specification. The "." will be replaced with "/".
     * @throws JarPackagerException
     */
    public static void makeJar(File jarFileName, String classFileName, String className) throws JarPackagerException
{
        jgslLogger.info("Creating " + jarFileName.getAbsolutePath() + " for JGSL script " + className);
        JarOutputStream targetJar;
        FileOutputStream fos;
        Manifest manifest = new Manifest();
        Attributes attrs = manifest.getMainAttributes();

        ResourceBundle res = ResourceBundle.getBundle("jgsl.resources.JGSL");
        String version = res.getString("jgsl.app.name");

        attrs.putValue("Manifest-Version", "1.0");
        attrs.putValue("Created-By", version);
        attrs.putValue("Main-Class", className);


        // Need the following files in the JAR
        //      the JGSL class
        //      jgsl_rt.jar
        //      log4j-1.2.9.jar

        try {
            fos = new FileOutputStream(jarFileName);
            targetJar = new JarOutputStream(fos, manifest);
            mergeJar(targetJar, "jgsl_rt.jar");
            mergeJar(targetJar, "log4j-1.2.9.jar");

            addEntry(targetJar, classFileName, className);

            targetJar.flush();
            targetJar.close();
            jgslLogger.info("JAR creation completed.");
        } catch (FileNotFoundException e) {
            jgslLogger.error(e.getMessage(), e);
            throw new JarPackagerException("Unable to create JAR file: " + jarFileName.getAbsolutePath());
        } catch (IOException e) {
            jgslLogger.error(e.getMessage(), e);
            throw new JarPackagerException("Unable to create JAR file: " + jarFileName.getAbsolutePath());
        }

    }

    /**
     * Add an entry to a jar file
     *
     * @param targetJar     Output stream of the jar to add the entry to
     * @param classFileName Full path to the class file to add to the jar
     * @param className     Name of the class with full package specification. The "." will be replaced with "/".
     * @throws IOException If reading/writing encounters an error
     */
    private static void addEntry(JarOutputStream targetJar, String classFileName, String className) throws IOException
{
        sysLogger.debug("BEGIN - JarPackager.addEntry");
        String jarEntryName = className.replace('.', '/') + ".class";
        JarEntry entry = new JarEntry(jarEntryName);

        targetJar.putNextEntry(entry);

        FileInputStream fis = new FileInputStream(classFileName + ".class");

        byte[] buf = new byte[4096];
        int bytesRead = 0;
```

```
          while ((bytesRead = fis.read(buf)) != -1) {
              targetJar.write(buf, 0, bytesRead);
          }
          targetJar.closeEntry();
          sysLogger.debug("END - JarPackager.addEntry");
      }

      /**
       * Merge the contents of a JAR file into another
       * <p/>
       * Looks for jarName in the JGSL cache: System.getProperty("user.home") + File.separator + ".jgsl" + File.
separator
       * + "cache";
       *
       * @param jarOut  Output stream of the jar to merge into
       * @param jarName Name of the JAR file to merge will taken from the JGSL cache
       * @throws IOException Thrown if reading/writing fails.
       */
      private static void mergeJar(JarOutputStream jarOut, String jarName) throws IOException {
          sysLogger.debug("BEGIN - JarPackager.mergeJar");
          JarInputStream jarIn;
          String jgslCache = System.getProperty("user.home") + File.separator + ".jgsl" + File.separator + "cache";
          File jarFile = new File(jgslCache + File.separator + jarName);

          jarIn = new JarInputStream(new FileInputStream(jarFile));
          // Create a read buffer to be used for transferring data from the input

          byte[] buf = new byte[4096];

          // Iterate the entries

          JarEntry entry;
          while ((entry = jarIn.getNextJarEntry()) != null) {
              // Exclude the Manifest file from the old JAR

              if (entry.getName().equals("META-INF/MANIFEST.MF")) {
                  continue;
              }

              // Write out the entry to the output JAR

              jarOut.putNextEntry(entry);
              int read;
              while ((read = jarIn.read(buf)) != -1) {
                  jarOut.write(buf, 0, read);
              }

              jarOut.closeEntry();
          }

          // Flush and close all the streams

```

```java

        jarIn.close();
        sysLogger.debug("END - JarPackager.mergeJar");
    }

}

```

/Users/jchavez/dev/java.net/jgsl.dev.java.net/jgsl/dev/src/jgsl/util/JarPackagerException.java

```
/*
 * Copyright (c) 2005 Perception Software. All Rights Reserved.
 */
package jgsl.util;

/**
 * Report JarPackageer exceptions.
 *
 * @author jchavez
 */
public class JarPackagerException extends Exception {
    public JarPackagerException(String string) {
        super(string);
    }
}
```

/Users/jchavez/dev/java.net/jgsl.dev.java.net/jgsl/dev/src/jgsl/view/swing/JGSLSwingFrame.java

```
/*
 * Copyright (c) 2005 Perception Software. All Rights Reserved.
 */
package jgsl.view.swing;

import com.intellij.uiDesigner.core.GridConstraints;
import com.intellij.uiDesigner.core.GridLayoutManager;
import org.apache.log4j.Logger;

import javax.swing.BorderFactory;
import javax.swing.DefaultListModel;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JList;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;
import java.awt.Dimension;
import java.awt.HeadlessException;
import java.awt.Insets;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.Locale;
import java.util.ResourceBundle;

import jgsl.controller.script.ScriptEngine;
import jgsl.controller.script.ScriptEngineException;
import jgsl.io.ImageFileFilter;
import jgsl.io.JARFileFilter;
```

```java
import jgsl.io.JGSLFileFilter;
import jgsl.io.ScriptParserException;
import jgsl.model.JGSLScript;

/**
 * The JGSLSwingFrame class is the main class for the interactive GUI.
 *
 * @author zenarchitect
 * @version $Id: JGSLSwingFrame.java,v 1.5 2005/05/21 01:42:11 zenarchitect Exp $
 */
public class JGSLSwingFrame implements ActionListener, ItemListener, ListSelectionListener {
    static Logger jgslLogger = Logger.getLogger("jgsl_log");
    static Logger sysLogger = Logger.getLogger("jgsl_sys_log");


    ResourceBundle res = ResourceBundle.getBundle("jgsl.view.swing.resources.JGSLSwingFrame", new Locale("en", 
"US"), Thread.currentThread().getContextClassLoader());

    private File currentFileName = new File("newfile.jgsl");
    private File newFileName;
    private boolean isNew = false;

    private JFrame frame;

    private JPanel mainPanel;
    private JTextArea scriptTextArea;
    private JPanel scriptPanel;
    private JButton viewButton;
    private JButton preferencesButton;
    private JButton quitButton;
    private JPanel actionPanel;
    private JPanel quitPanel;


    private JPanel statusPanel;
    private JList statusList;
    private JScrollPane statusScrollPane;

    JMenuBar menuBar;

    JMenu fileMenu;
    JMenuItem newMenuItem;
    JMenuItem openMenuItem;
    JMenuItem saveMenuItem;
    JMenuItem saveAsMenuItem;
    JMenuItem exitMenuItem;

    JMenu viewMenu;

    JMenu helpMenu;
    JMenuItem helpMenuItem;
    JMenuItem aboutMenuItem;

    DefaultListModel statusListModel = new DefaultListModel();
    private int MAX_STATUS_LIST_SIZE = 250;
    private JScrollPane scriptScrollPane;
    private String frameTitle;
    private JButton jarButton;
    private JTextField scriptOutputFileName;
    private JButton selectScriptOutputButton;
    private JCheckBox saveSciptOutputCheckBox;
    private boolean saveScriptOuput;

    /**
     * Constructs a new frame that is initially invisible.
     * <p/>
     * This constructor sets the component's locale property to the value returned by
     * <code>JComponent.getDefaultLocale</code>.
     *
     * @throws java.awt.HeadlessException if GraphicsEnvironment.isHeadless() returns true.
     * @see java.awt.GraphicsEnvironment#isHeadless
     * @see java.awt.Component#setSize
     * @see java.awt.Component#setVisible
     * @see javax.swing.JComponent#getDefaultLocale
     */
    public JGSLSwingFrame(JFrame frame) throws HeadlessException {
        sysLogger.debug("BEGIN: JGSLSwingFrame");

        this.frame = frame;

        //Center frame
        Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
        Dimension size = frame.getSize();
        screenSize.height = screenSize.height / 2;
        screenSize.width = screenSize.width / 2;
        size.height = size.height / 2;
        size.width = size.width / 2;
        int y = screenSize.height - size.height;
        int x = screenSize.width - size.width;
        frame.setLocation(x, y);
        sysLogger.debug("JGSLSwingFrame: frame setup completed...");

        if (viewButton == null) {
            sysLogger.debug("JGSLSwingFrame: goButton == null");
        }

        viewButton.addActionListener(this);

        jarButton.addActionListener(this);
        selectScriptOutputButton.addActionListener(this);
//        quitButton.addActionListener(this);

        saveSciptOutputCheckBox.addItemListener(this);

        statusList.addListSelectionListener(this);
        statusList.setModel(statusListModel);
        sysLogger.debug("JGSLSwingFrame: status list setup completed...");


        menuBar = new JMenuBar();
        fileMenu = new JMenu("File");

        newMenuItem = new JMenuItem("New");
        newMenuItem.addActionListener(this);
        openMenuItem = new JMenuItem("Open");
        openMenuItem.addActionListener(this);
        saveMenuItem = new JMenuItem("Save");
        saveMenuItem.addActionListener(this);
        saveAsMenuItem = new JMenuItem("Save As");
        saveAsMenuItem.addActionListener(this);

        exitMenuItem = new JMenuItem("Exit");
        exitMenuItem.addActionListener(this);

        fileMenu.add(newMenuItem);
        fileMenu.addSeparator();
        fileMenu.add(openMenuItem);
        fileMenu.add(saveMenuItem);
        fileMenu.add(saveAsMenuItem);
        fileMenu.addSeparator();
        fileMenu.add(exitMenuItem);


        viewMenu = new JMenu();


        helpMenu = new JMenu("Help");
        helpMenuItem = new JMenuItem("Help Topics");
        helpMenuItem.addActionListener(this);
        aboutMenuItem = new JMenuItem("About");
        aboutMenuItem.addActionListener(this);

        helpMenu.add(helpMenuItem);
        helpMenu.add(aboutMenuItem);

        menuBar.add(fileMenu);
//        menuBar.add(viewMenu);
        menuBar.add(helpMenu);
        sysLogger.debug("JGSLSwingFrame: menu bar setup completed...");


        frame.setJMenuBar(menuBar);
        sysLogger.debug("JGSLSwingFrame: menu bar added to main window...");


        frameTitle = res.getString("jgsl.i-gui.title");
```

```java
        if (frameTitle != null) {
            frame.setTitle(frameTitle);
        } else {
            frame.setTitle("JGSL 1.0 - Java Web Start");
        }
        sysLogger.debug("JGSLSwingFrame: window title set...");

        addStatusItem("JGSL 1.0 ready...");

        sysLogger.debug("END: JGSLSwingFrame");

    }

    /**
     * Return reference to the main panel.
     */
    public JPanel getMainPanel() {
        return mainPanel;
    }

    /**
     * Start the JGSL. This method is needed to work around the strange startup requirements by the IntelliJ IDEA GUI
     * builder.
     */
    public static void startJGSL(String[] args) {
        sysLogger.debug("BEGIN: startJGSL");
        JFrame f = new JFrame();

        f.setSize(640, 480);

        JGSLSwingFrame mainFrame = new JGSLSwingFrame(f);

        f.setContentPane(mainFrame.getMainPanel());

        f.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent ev) {
                System.exit(0);
            }
        });

        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);


//        f.pack();
        f.setVisible(true);
        sysLogger.debug("END: startJGSL");

    }

    /**
     * Program entry point.
     */
    public static void main(String[] args) {
        startJGSL(args);
    }

    public void actionPerformed(ActionEvent actionEvent) {

        if (actionEvent.getSource() == quitButton || actionEvent.getSource() == exitMenuItem) {
            frame.setVisible(false);
            System.exit(0);
        } else if (actionEvent.getSource() == newMenuItem) {
            newScript();
        } else if (actionEvent.getSource() == openMenuItem) {
            openScript();
        } else if (actionEvent.getSource() == saveMenuItem) {
            storeScript();
        } else if (actionEvent.getSource() == viewButton) {
            viewScript();
        } else if (actionEvent.getSource() == jarButton) {
            jarScript();
        } else if (actionEvent.getSource() == selectScriptOutputButton) {
            selectScriptOutputFile();
        } else if (actionEvent.getSource() == saveAsMenuItem) {
            saveAsScript();
        }
    }

    private void selectScriptOutputFile() {
        JFileChooser chooser = new JFileChooser();
        ImageFileFilter filter = new ImageFileFilter();
        chooser.setFileFilter(filter);
        int returnVal = chooser.showSaveDialog(getMainPanel());
        if (returnVal == JFileChooser.APPROVE_OPTION) {
            File outputFileName = chooser.getSelectedFile();
            this.scriptOutputFileName.setText(outputFileName.getAbsolutePath());
        }
    }

    private void jarScript() {
        JFileChooser chooser = new JFileChooser();
        JARFileFilter filter = new JARFileFilter();
        chooser.setFileFilter(filter);
        int returnVal = chooser.showSaveDialog(getMainPanel());
        if (returnVal == JFileChooser.APPROVE_OPTION) {
            File jarFileName = chooser.getSelectedFile();
            if (!jarFileName.getName().endsWith(".jar")) {
                jarFileName = new File(jarFileName.getAbsolutePath() + ".jar");
            }
            // parse and validate
            // generate implementation class
            // execute viewer
            ScriptEngine se = new ScriptEngine();
            try {
                JGSLScript script = se.jarInteractive(currentFileName, jarFileName);
                if (script.hasErrors()) {
// TODO - Get the errors as an object and add to status list
// TODO - Make status list context aware and select error to show user in script
                    addStatusItem(script.getParseStatus());
                }
            } catch (ScriptEngineException e) {
                e.printStackTrace();
                addStatusItem("An error was encontered creating a JAR for your script, details are provided below");
                addStatusItem(e.getMessage());
            } catch (ScriptParserException e) {
                e.printStackTrace();
                addStatusItem("An error was encontered parsing your script, details are provided below");
                addStatusItem(e.getMessage());
            }
        }
    }

    private void storeScript() {
        if (isNew) {
            saveAsScript();
        } else {
            newFileName = currentFileName;
            saveScript();
        }

    }

    private void viewScript() {
        // save the script
        storeScript();

        // parse and validate
        // generate implementation class
        // execute viewer
        ScriptEngine se = new ScriptEngine();
        try {
            String scriptOutputFileName = null;
            if(saveScriptOuput) {
                scriptOutputFileName = this.scriptOutputFileName.getText();
            }
            JGSLScript script = se.viewInteractive(currentFileName, scriptOutputFileName); // TODO add file name to GUI
            if (script.hasErrors()) {
// TODO - Make status list context aware and select error to show user in script
                addStatusItem(script.getParseStatus());
            }
        } catch (ScriptParserException e) {
            e.printStackTrace();
            addStatusItem("An error was encontered parsing your script, details are provided below");
            addStatusItem(e.getMessage());
        }

```

```java
    }

    public void itemStateChanged(ItemEvent itemEvent) {

        if(itemEvent.getSource() == saveSciptOutputCheckBox) {
            saveScriptOuput = !saveScriptOuput;
        }
    }

    public void valueChanged(ListSelectionEvent listSelectionEvent) {
        //TODO: implement method
    }

    private void newScript() {
        String basicScript = "/*\n" +
                "JGSL Script\n" +
                "*/\n" +
                "\n" +
                "begin\n" +
                "\n" +
                "// Initialize the canvas\n" +
                "canvas (640, 480, BLACK, WHITE, \"jgsl script\");\n" +
                "\n" +
                "end\n";
        try {
            InputStreamReader isr = new InputStreamReader(Thread.currentThread().getContextClassLoader().
getResourceAsStream("jgsl/resources/template.jgsl"));
            BufferedReader br = new BufferedReader(isr);
            String line = "";
            StringBuffer sb = new StringBuffer(1024);
            while ((line = br.readLine()) != null) {
                sb.append(line);
                sb.append("\n");
            }
            isr.close();
            scriptTextArea.setText(sb.toString());
            addStatusItem("Template script loaded, use the script editor write your script.");
            frame.setTitle(frameTitle + " - (new)");
        } catch (IOException e1) {
            e1.printStackTrace();
            addStatusItem("Unable to read file template file, creating basic script.");
            scriptTextArea.setText(basicScript);
        }
        isNew = true;
    }

    private void saveScript() {
        addStatusItem("Saving " + newFileName);
        try {
            FileWriter fw = new FileWriter(newFileName);
            fw.write(scriptTextArea.getText(), 0, scriptTextArea.getText().length());
            fw.close();
            currentFileName = newFileName;
            addStatusItem("Saved file " + newFileName);
            isNew = false;
            frame.setTitle(frameTitle + " - " + currentFileName);
        } catch (IOException e1) {
            e1.printStackTrace();
            addStatusItem("Unable to write file: " + newFileName);
        }
    }

    private void saveAsScript() {
        JFileChooser chooser = new JFileChooser();
        JGSLFileFilter filter = new JGSLFileFilter();
        chooser.setFileFilter(filter);
        int returnVal = chooser.showSaveDialog(getMainPanel());
        if (returnVal == JFileChooser.APPROVE_OPTION) {
            newFileName = chooser.getSelectedFile();
            if (!newFileName.getName().endsWith(".jgsl")) {
                newFileName = new File(newFileName.getAbsolutePath() + ".jgsl");
            }
            saveScript();
        }
    }

    private void openScript() {
        JFileChooser chooser = new JFileChooser();
        JGSLFileFilter filter = new JGSLFileFilter();
        chooser.setFileFilter(filter);
        int returnVal = chooser.showOpenDialog(getMainPanel());
        if (returnVal == JFileChooser.APPROVE_OPTION) {
            File newFileName = chooser.getSelectedFile();
            addStatusItem("Opening " + newFileName);
            try {
                FileReader fr = new FileReader(newFileName);
                BufferedReader br = new BufferedReader(fr);
                String line = "";
                StringBuffer sb = new StringBuffer(1024);
                while ((line = br.readLine()) != null) {
                    sb.append(line);
                    sb.append("\n");
                }
                fr.close();
                scriptTextArea.setText(sb.toString());
                scriptTextArea.setCaretPosition(0);
                currentFileName = newFileName;
                addStatusItem("Opened " + newFileName);
                isNew = false;
                frame.setTitle(frameTitle + " - " + currentFileName);
            } catch (IOException e1) {
                e1.printStackTrace();
                addStatusItem("Unable to read file: " + newFileName);
            }
        }
    }

    private void addStatusItem(String item) {
        if (statusListModel.getSize() > MAX_STATUS_LIST_SIZE * 1.25) {
            statusListModel.removeRange(MAX_STATUS_LIST_SIZE, (int) ((MAX_STATUS_LIST_SIZE * 1.25) - 1));
        }
        statusListModel.addElement(item);
        statusList.ensureIndexIsVisible(statusListModel.size() - 1);

    }


    {
// GUI initializer generated by IntelliJ IDEA GUI Designer
// >>> IMPORTANT!! <<<
// DO NOT EDIT OR ADD ANY CODE HERE!
        $$$setupUI$$$();
    }

    /**
     * Method generated by IntelliJ IDEA GUI Designer >>> IMPORTANT!! <<< DO NOT edit this method OR call it in your
     * code!
     */
    private void $$$setupUI$$$() {
        mainPanel = new JPanel();
        mainPanel.setLayout(new GridLayoutManager(3, 3, new Insets(0, 0, 0, 0), -1, -1));
        scriptPanel = new JPanel();
        scriptPanel.setLayout(new GridLayoutManager(1, 1, new Insets(0, 0, 0, 0), -1, -1));
        mainPanel.add(scriptPanel, new GridConstraints(0, 0, 1, 3, GridConstraints.ANCHOR_CENTER, GridConstraints.FILL_BOTH, GridConstraints.SIZEPOLICY_CAN_SHRINK | GridConstraints.SIZEPOLICY_CAN_GROW, GridConstraints.SIZEPOLICY_CAN_SHRINK | GridConstraints.SIZEPOLICY_CAN_GROW, new Dimension(320, 200), new Dimension(640, 480), null));
        scriptPanel.setBorder(BorderFactory.createTitledBorder(BorderFactory.createLoweredBevelBorder(), "Script Editor"));
        scriptScrollPane = new JScrollPane();
        scriptPanel.add(scriptScrollPane, new GridConstraints(0, 0, 1, 1, GridConstraints.ANCHOR_CENTER, GridConstraints.FILL_BOTH, GridConstraints.SIZEPOLICY_CAN_SHRINK | GridConstraints.SIZEPOLICY_WANT_GROW, GridConstraints.SIZEPOLICY_CAN_SHRINK | GridConstraints.SIZEPOLICY_WANT_GROW, null, null, null));
        scriptTextArea = new JTextArea();
        scriptScrollPane.setViewportView(scriptTextArea);
        actionPanel = new JPanel();
        actionPanel.setLayout(new GridLayoutManager(1, 4, new Insets(0, 0, 0, 0), -1, -1));
        mainPanel.add(actionPanel, new GridConstraints(2, 0, 1, 2, GridConstraints.ANCHOR_WEST, GridConstraints.FILL_NONE, GridConstraints.SIZEPOLICY_FIXED, GridConstraints.SIZEPOLICY_FIXED, null, null, null));
        actionPanel.setBorder(BorderFactory.createTitledBorder(BorderFactory.createEtchedBorder(), "Scipt Actions"));
        viewButton = new JButton();
        viewButton.setText("View");
        viewButton.setToolTipText("View your JGLS script.");
        actionPanel.add(viewButton, new GridConstraints(0, 0, 1, 1, GridConstraints.ANCHOR_CENTER, GridConstraints.
FILL_HORIZONTAL, GridConstraints.SIZEPOLICY_FIXED, GridConstraints.SIZEPOLICY_FIXED, null, null, null));
        jarButton = new JButton();
        jarButton.setText("Create JAR");
        jarButton.setToolTipText("Create an executable JAR file from your JGSL script.");
        actionPanel.add(jarButton, new GridConstraints(0, 1, 1, 1, GridConstraints.ANCHOR_CENTER, GridConstraints.
FILL_HORIZONTAL, GridConstraints.SIZEPOLICY_FIXED, GridConstraints.SIZEPOLICY_FIXED, null, null, null));
        final JPanel panel0 = new JPanel();
        panel0.setLayout(new GridLayoutManager(2, 2, new Insets(0, 0, 0, 0), -1, -1));
        actionPanel.add(panel0, new GridConstraints(0, 2, 1, 1, GridConstraints.ANCHOR_CENTER, GridConstraints.
FILL_BOTH, GridConstraints.SIZEPOLICY_CAN_SHRINK | GridConstraints.SIZEPOLICY_CAN_GROW, GridConstraints.
SIZEPOLICY_CAN_SHRINK | GridConstraints.SIZEPOLICY_CAN_GROW, null, null, null));
        selectScriptOutputButton = new JButton();
        selectScriptOutputButton.setText("Select");
        selectScriptOutputButton.setToolTipText("Press this button to enter a file name.");
        panel0.add(selectScriptOutputButton, new GridConstraints(1, 1, 1, 1, GridConstraints.ANCHOR_CENTER,
GridConstraints.FILL_HORIZONTAL, GridConstraints.SIZEPOLICY_CAN_SHRINK | GridConstraints.
SIZEPOLICY_CAN_GROW, GridConstraints.SIZEPOLICY_FIXED, null, null, null));
        scriptOutputFileName = new JTextField();
        scriptOutputFileName.setEditable(false);
        scriptOutputFileName.setInheritsPopupMenu(false);
        scriptOutputFileName.setText("jgsl_image.jpg");
        scriptOutputFileName.setToolTipText("File name to which your JGSL script image will be saved to.");
        panel0.add(scriptOutputFileName, new GridConstraints(1, 0, 1, 1, GridConstraints.ANCHOR_WEST,
GridConstraints.FILL_HORIZONTAL, GridConstraints.SIZEPOLICY_WANT_GROW, GridConstraints.
SIZEPOLICY_FIXED, null, new Dimension(150, -1), null));
        saveSciptOutputCheckBox = new JCheckBox();
        saveSciptOutputCheckBox.setText("Save script output");
        saveSciptOutputCheckBox.setToolTipText("Check this to generate an image from your JGSL script.");
        panel0.add(saveSciptOutputCheckBox, new GridConstraints(0, 0, 1, 1, GridConstraints.ANCHOR_WEST,
GridConstraints.FILL_NONE, GridConstraints.SIZEPOLICY_CAN_SHRINK | GridConstraints.
SIZEPOLICY_CAN_GROW, GridConstraints.SIZEPOLICY_FIXED, null, null, null));
        statusPanel = new JPanel();
        statusPanel.setLayout(new GridLayoutManager(1, 1, new Insets(0, 0, 0, 0), -1, -1));
        mainPanel.add(statusPanel, new GridConstraints(1, 0, 1, 3, GridConstraints.ANCHOR_CENTER, GridConstraints.
FILL_BOTH, GridConstraints.SIZEPOLICY_CAN_SHRINK | GridConstraints.SIZEPOLICY_CAN_GROW, GridConstraints.
SIZEPOLICY_CAN_SHRINK | GridConstraints.SIZEPOLICY_CAN_GROW, null, null, null));
        statusPanel.setBorder(BorderFactory.createTitledBorder(BorderFactory.createLoweredBevelBorder(), "Status"));
        statusScrollPane = new JScrollPane();
        statusPanel.add(statusScrollPane, new GridConstraints(0, 0, 1, 1, GridConstraints.ANCHOR_CENTER,
GridConstraints.FILL_BOTH, GridConstraints.SIZEPOLICY_CAN_SHRINK | GridConstraints.
SIZEPOLICY_WANT_GROW, GridConstraints.SIZEPOLICY_CAN_SHRINK | GridConstraints.
SIZEPOLICY_WANT_GROW, null, null, null));
        statusList = new JList();
        statusList.setToolTipText("Status of JGSL actions.");
        statusScrollPane.setViewportView(statusList);
    }
}
```

/Users/jchavez/dev/java.net/jgsl.dev.java.net/jgsl/dev/src/jgsl/view/swing/SwingScriptViewer.java

```java
/*
 * Copyright (c) 2005 Perception Software. All Rights Reserved.
 */
package jgsl.view.swing;


import org.apache.log4j.Logger;

import java.io.BufferedReader;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.Map;

import jgsl.view.ScriptViewer;


/**
 * The SwingScriptViewer class creates an JVM that executes the JGSLViewer class with an argument of the compiled JGSL
 * script Java class.
 *
 * @author zenarchitect
 * @version $Id: SwingScriptViewer.java,v 1.6 2005/05/21 01:42:11 zenarchitect Exp $
 */

public class SwingScriptViewer implements ScriptViewer {
    static Logger jgslLogger = Logger.getLogger("jgsl_log");
    static Logger sysLogger = Logger.getLogger("jgsl_sys_log");
    static final boolean DEBUG = false;

    /**
     * Reder the script by creating a Process object with the properly JGSL runtime class path. The runtime classpath
     * includes the compile JGSL script in the form of a Java class. Also required on the classpath are the jgsl_rt.jar
     * and log4j-1.2.9.jar files.
     *
     * @param fullClassName
     */
    public void renderScript(String fullClassName, String saveToFileName) {
        sysLogger.debug("BEGIN - renderScript");
        // The script is rendered in a swing gui by creating a new class from the script
        // that is a subclass of a base JFrame that overrides the paint method
        try {
            sysLogger.debug("fullClassName = " + fullClassName);
            ProcessBuilder pb;
            if (saveToFileName != null) {
                pb = new ProcessBuilder(System.getProperty("java.home") +
                        File.separator + "bin" +
                        File.separator + "java", "" +
                        "jgsl.view.swing.JGSLViewer",
                        fullClassName,
                        saveToFileName);
            } else {
                pb = new ProcessBuilder(System.getProperty("java.home") +
                        File.separator + "bin" +
                        File.separator + "java", "" +
                        "jgsl.view.swing.JGSLViewer",
                        fullClassName);
            }
            if (DEBUG) {
                pb.redirectErrorStream(true);
            }
            String jgslCache = System.getProperty("user.home") + File.separator + ".jgsl" + File.separator + "cache";
            String classPath = System.getProperty("java.class.path");
            classPath += File.pathSeparator + jgslCache;
            classPath += File.pathSeparator + jgslCache + File.separator + "jgsl_rt.jar";
            classPath += File.pathSeparator + jgslCache + File.separator + "log4j-1.2.9.jar";
            sysLogger.debug("Class path = " + classPath);
            Map<String, String> env = pb.environment();
            env.put("CLASSPATH", classPath);

            Process p = pb.start();

            if (DEBUG) {
                InputStream is = p.getInputStream();
                BufferedReader br = new BufferedReader(new InputStreamReader(is));
                String line;
                while ((line = br.readLine()) != null) {
                    System.out.println(line);
                }
            }
            sysLogger.debug("END - renderScript");

        } catch (IOException e) {
            e.printStackTrace();  // TODO: Handle exception
            sysLogger.debug(e.getMessage());
        } catch (Exception e) {
            e.printStackTrace();  // TODO: Handle exception
            sysLogger.debug(e.getMessage());
        }
    }

    /**
     * @directed
     */
    /*# JGSLViewer lnkJGSLViewer; */
}

```

/Users/jchavez/dev/java.net/jgsl.dev.java.net/jgsl/dev/src/jgsl/controller/script/ScriptEngineException.java

```
/*
 * Copyright (c) 2005 Perception Software. All Rights Reserved.
 */
package jgsl.controller.script;

/**
 * ScriptEngineException is thrown by the ScriptEngine class to report exception conditions.
 *
 * @author zenarchitect
 * @version $Id: ScriptEngineException.java,v 1.3 2005/05/16 00:54:15 zenarchitect Exp $
 */

public class ScriptEngineException extends Throwable {
    /**
     * Constructs a new throwable with <code>null</code> as its detail message. The cause is not initialized, and may
     * subsequently be initialized by a call to {@link #initCause}.
     * <p/>
     * <p>The {@link #fillInStackTrace()} method is called to initialize the stack trace data in the newly created
     * throwable.
     */
    public ScriptEngineException() {
        super();
    }

    /**
     * Constructs a new throwable with the specified detail message.  The cause is not initialized, and may subsequently
     * be initialized by a call to {@link #initCause}.
     * <p/>
     * <p>The {@link #fillInStackTrace()} method is called to initialize the stack trace data in the newly created
     * throwable.
     *
     * @param message the detail message. The detail message is saved for later retrieval by the {@link #getMessage()}
     *                method.
     */
    public ScriptEngineException(String message) {
        super(message);
    }

    /**
     * Constructs a new throwable with the specified detail message and cause.  <p>Note that the detail message
     * associated with <code>cause</code> is <i>not</i> automatically incorporated in this throwable's detail
message.
     * <p/>
     * <p>The {@link #fillInStackTrace()} method is called to initialize the stack trace data in the newly created
     * throwable.
     *
     * @param message the detail message (which is saved for later retrieval by the {@link #getMessage()} method).
     * @param cause   the cause (which is saved for later retrieval by the {@link #getCause()} method).  (A
```

```java
     *              <tt>null</tt> value is permitted, and indicates that the cause is nonexistent or unknown.)
     * @since 1.4
     */
    public ScriptEngineException(String message, Throwable cause) {
        super(message, cause);
    }

    /**
     * Constructs a new throwable with the specified cause and a detail message of <tt>(cause==null ? null :
     * cause.toString())</tt> (which typically contains the class and detail message of <tt>cause</tt>). This
     * constructor is useful for throwables that are little more than wrappers for other throwables (for example, {@link
     * java.security.PrivilegedActionException}).
     * <p/>
     * <p>The {@link #fillInStackTrace()} method is called to initialize the stack trace data in the newly created
     * throwable.
     *
     * @param cause the cause (which is saved for later retrieval by the {@link #getCause()} method).  (A <tt>null</tt>
     *              value is permitted, and indicates that the cause is nonexistent or unknown.)
     * @since 1.4
     */
    public ScriptEngineException(Throwable cause) {
        super(cause);
    }
}
```

/Users/jchavez/dev/java.net/jgsl.dev.java.net/jgsl/dev/src/jgsl/parser/JGSL_Parser.java

```java
    /* Generated By:JavaCC: Do not edit this line. JGSL_Parser.java */
package jgsl.parser;


import java.awt.Color;
import java.util.ArrayList;

import jgsl.io.ScriptError;
import jgsl.io.ScriptParserException;
import jgsl.io.ScriptParserUtil;
import jgsl.io.ScriptWarning;
import jgsl.model.Assignment;
import jgsl.model.Command;
import jgsl.model.Declaration;
import jgsl.model.JGSLColor;
import jgsl.model.JGSLDouble;
import jgsl.model.JGSLInteger;
import jgsl.model.JGSLScript;
import jgsl.model.JGSLString;

/**
 *
 * @author zenarchitect
 * @version $Id: JGSL_Parser.java,v 1.10 2005/05/21 01:42:08 zenarchitect Exp $
 */

public class JGSL_Parser implements JGSL_ParserConstants {

  private JGSLScript script = new JGSLScript();

  public JGSLScript getScript() {
      return script;
  }

  public static void main(String args[]) throws ParseException {
        JGSL_Parser parser;
        String filename = null;
        long initTime = 0;
        long parseTime = 0;
        long startTime = 0;
        long stopTime = 0;
        if (args.length == 0)
        {
            System.out.println("jgsl parser:  Reading from standard input . . .");
            parser = new JGSL_Parser(System.in);
        } else if (args.length == 1)
        {
            filename = args[0];
            System.out.println("jgsl parser:  Reading from file " + filename + " . . .");
            try
            {
                startTime = System.currentTimeMillis();
                parser = new JGSL_Parser(new java.io.FileInputStream(filename));
                stopTime = System.currentTimeMillis();
                initTime = stopTime - startTime;
            } catch (java.io.FileNotFoundException e)
            {
                System.out.println("jgsl parser:  File " + filename + " not found.");
                return;
            }
        } else
        {
            System.out.println("jgsl parser:  Usage is one of:");
            System.out.println("         java jgsl.JSGL < <stdin>");
            System.out.println("OR");
            System.out.println("         java jgsl.JSGL inputfile");
            return;
        }
        try
        {
            startTime = System.currentTimeMillis();
            parser.parseScript();
            stopTime = System.currentTimeMillis();
            parseTime = stopTime - startTime;
            System.out.println("jgsl parser: ");
            System.out.println("   JGSL file parsed " + filename + " successfully in " + (initTime + parseTime) + " ms.");
            System.out.println("      initialization time = " + initTime + " ms.");
            System.out.println("      parse time = " + parseTime + " ms.");
        } catch (ParseException e)
        {
            System.out.println(e.getMessage());
            System.out.println("jgsl parser:  Encountered errors during parse.");
        }
  }

/*
<comments | documentation>*

<BEGIN>

<attributes | commands | comments | documentation>

<END>

*/

/*******************************************
 * THE JGSL GRAMMAR STARTS HERE          *
 *******************************************/

 /*
  * Program structuring syntax follows.
  */
  final public void parseScript() throws ParseException {
    Script();
  }

  final public void Script() throws ParseException {
    label_1:
    while (true) {
      switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
      case DOC:
        ;
        break;
      default:
        jj_la1[0] = jj_gen;
        break label_1;
      }
      Documentation();
    }
    switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
    case BEGIN:
      ScriptBody();
      break;
    default:
      jj_la1[1] = jj_gen;
      ;
    }
    jj_consume_token(0);
  }

  final public void Documentation() throws ParseException {
    Token doc;
    jj_consume_token(DOC);
    doc = jj_consume_token(STRING_LITERAL);
    script.addDocumentation(doc.image);
  }

  final public void ScriptBody() throws ParseException {
    jj_consume_token(BEGIN);
    label_2:
    while (true) {
      switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
      case CLEAR:
      case CANVAS:
      case DRAW:
      case TEXT:
      case RECTANGLE:
      case SQUARE:
      case CIRCLE:
      case ELIPSE:
      case ARC:
      case POLYGON:
      case LINE:
      case WAIT:
      case LOG:
      case DEBUG:
      case ERROR:
      case WARNING:
      case DECLARE:
      case DOC:
      case IDENTIFIER:
        ;
        break;
      default:
        jj_la1[2] = jj_gen;
        break label_2;
      }
      switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
      case CLEAR:
      case CANVAS:
      case DRAW:
      case TEXT:
      case RECTANGLE:
      case SQUARE:
      case CIRCLE:
      case ELIPSE:
      case ARC:
      case POLYGON:
      case LINE:
      case WAIT:
      case LOG:
      case DEBUG:
      case ERROR:
      case WARNING:
        Command();
        break;
      case DECLARE:
        Declaration();
        break;
      case IDENTIFIER:
        Assignment();
        break;
      case DOC:
        Documentation();
        break;
      default:
        jj_la1[3] = jj_gen;
        jj_consume_token(-1);
        throw new ParseException();
      }
    }
    jj_consume_token(END);
  }

  final public void Command() throws ParseException {
    try {
      switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
      case CANVAS:
        Canvas();
        break;
      case CLEAR:
        Clear();
        break;
      case WAIT:
        Wait();
        break;
      case DRAW:
        Draw();
        break;
```

```java
      case RECTANGLE:
      case SQUARE:
      case CIRCLE:
      case ELIPSE:
      case ARC:
      case POLYGON:
      case LINE:
        DrawShape();
        break;
      case TEXT:
        DrawText();
        break;
      case LOG:
      case DEBUG:
      case ERROR:
      case WARNING:
        Log();
        break;
      default:
        jj_la1[4] = jj_gen;
        jj_consume_token(-1);
        throw new ParseException();
      }
      jj_consume_token(SEMICOLON);
    } catch (ParseException e) {
         error_skipto(SEMICOLON);
    }
  }

  void error_skipto(int kind) throws ParseException {
    ParseException e = generateParseException();  // generate the exception object.

    ScriptError se = new ScriptError(e.getMessage());
    script.addError(se);

    Token t;
    do {
    t = getNextToken();
    } while (t.kind != kind);
  }

  final public void Clear() throws ParseException {
    Token name;
    name = jj_consume_token(CLEAR);
        Command c = new Command(name.image);
        script.add(c);
  }

  final public Color GetStandardColor() throws ParseException {
    Token value;
    try {
      switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
      case BLACK:
        value = jj_consume_token(BLACK);
        break;
      case BLUE:
        value = jj_consume_token(BLUE);
        break;
      case DARK_GRAY:
        value = jj_consume_token(DARK_GRAY);
        break;
      case GRAY:
        value = jj_consume_token(GRAY);
        break;
      case GREEN:
        value = jj_consume_token(GREEN);
        break;
      case LIGHT_GRAY:
        value = jj_consume_token(LIGHT_GRAY);
        break;
      case MAGENTA:
        value = jj_consume_token(MAGENTA);
        break;
      case ORANGE:
        value = jj_consume_token(ORANGE);
        break;
      case PINK:
        value = jj_consume_token(PINK);
        break;
      case RED:
        value = jj_consume_token(RED);
        break;
      case WHITE:
        value = jj_consume_token(WHITE);
        break;
      case YELLOW:
        value = jj_consume_token(YELLOW);
        break;
      default:
        jj_la1[5] = jj_gen;
        jj_consume_token(-1);
        throw new ParseException();
      }
    } catch (ParseException e) {
        ScriptError se = new ScriptError(e.getMessage());
        script.addError(se);
        {if (true) return null;}
    }
//        Color color = Color.RED;
        String colorName = value.image.toLowerCase();
        if(colorName.equals("black")) {
            {if (true) return Color.BLACK;}
        }
        else if(colorName.equals("blue")) {
            {if (true) return Color.BLUE;}
        }
        else if(colorName.equals("dark_gray")) {
            {if (true) return Color.DARK_GRAY;}
        }
        else if(colorName.equals("gray")) {
            {if (true) return Color.GRAY;}
        }
        else if(colorName.equals("green")) {
            {if (true) return Color.GREEN;}
        }
        else if(colorName.equals("light_gray")) {
            {if (true) return Color.LIGHT_GRAY;}
        }
        else if(colorName.equals("magenta")) {
            {if (true) return Color.MAGENTA;}
        }
        else if(colorName.equals("orange")) {
            {if (true) return Color.ORANGE;}
        }
        else if(colorName.equals("pink")) {
            {if (true) return Color.PINK;}
        }
        else if(colorName.equals("red")) {
            {if (true) return Color.RED;}
        }
        else if(colorName.equals("white")) {
            {if (true) return Color.WHITE;}
        }
        else if(colorName.equals("yellow")) {
            {if (true) return Color.YELLOW;}
        }
    throw new Error("Missing return statement in function");
  }

  final public Color GetRGB() throws ParseException {
    Token r;
    Token g;
    Token b;
    switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
    case INTEGER_LITERAL:
      r = jj_consume_token(INTEGER_LITERAL);
      jj_consume_token(COMMA);
      g = jj_consume_token(INTEGER_LITERAL);
      jj_consume_token(COMMA);
      b = jj_consume_token(INTEGER_LITERAL);
      break;
    case STRING_LITERAL:
      r = jj_consume_token(STRING_LITERAL);
      jj_consume_token(COMMA);
      g = jj_consume_token(STRING_LITERAL);
      jj_consume_token(COMMA);
      b = jj_consume_token(STRING_LITERAL);
      break;
    default:
      jj_la1[6] = jj_gen;
      jj_consume_token(-1);
      throw new ParseException();
    }
//        Color rgb = new Color(0,0,0);
        int rInt = -1;
        int gInt = -1;
        int bInt = -1;

        try {
            rInt = ScriptParserUtil.parseInt(r.image);
        }
        catch(ScriptParserException e) {
            ScriptError se = new ScriptError(e.getMessage(), r.beginLine, r.beginColumn);
            script.addError(se);
        }
        try {
            rInt = ScriptParserUtil.parseInt(g.image);
        }
        catch(ScriptParserException e) {
            ScriptError se = new ScriptError(e.getMessage(), g.beginLine, g.beginColumn);
            script.addError(se);
        }
        try {
            rInt = ScriptParserUtil.parseInt(b.image);
        }
        catch(ScriptParserException e) {
            ScriptError se = new ScriptError(e.getMessage(), b.beginLine, b.beginColumn);
            script.addError(se);
        }

        Color c = null;

        if(rInt != -1 && gInt != -1 && bInt != -1) {
            c = new Color(rInt, gInt, bInt);
        }

        {if (true) return c;}
    throw new Error("Missing return statement in function");
  }

  final public Color GetColor() throws ParseException {
    Color c;
    switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
    case BLACK:
    case BLUE:
    case DARK_GRAY:
    case GRAY:
    case GREEN:
    case LIGHT_GRAY:
    case MAGENTA:
    case ORANGE:
    case PINK:
    case RED:
    case WHITE:
    case YELLOW:
      c = GetStandardColor();
      break;
    case INTEGER_LITERAL:
    case STRING_LITERAL:
      c = GetRGB();
      break;
    default:
      jj_la1[7] = jj_gen;
      jj_consume_token(-1);
      throw new ParseException();
    }
        {if (true) return c;}
    throw new Error("Missing return statement in function");
  }

  final public void Canvas() throws ParseException {
    Token name = null;
    Token width = null;
    Token height = null;
    Color bgcolor = null;
    Color fgcolor = null;
    Token title = null;
    ArrayList attributes = new ArrayList();
    name = jj_consume_token(CANVAS);
    switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
    case COLON:
      jj_consume_token(COLON);
      attributes = CanvasAttributes();
      break;
    default:
      jj_la1[8] = jj_gen;
      ;
    }
    jj_consume_token(LPAREN);
    switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
    case INTEGER_LITERAL:
      width = jj_consume_token(INTEGER_LITERAL);
      break;
    case STRING_LITERAL:
      width = jj_consume_token(STRING_LITERAL);
      break;
    default:
      jj_la1[9] = jj_gen;
      jj_consume_token(-1);
      throw new ParseException();
    }
    jj_consume_token(COMMA);
    switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
    case INTEGER_LITERAL:
      height = jj_consume_token(INTEGER_LITERAL);
      break;
    case STRING_LITERAL:
      height = jj_consume_token(STRING_LITERAL);
      break;
    default:
      jj_la1[10] = jj_gen;
      jj_consume_token(-1);
      throw new ParseException();
    }
    jj_consume_token(COMMA);
    bgcolor = GetColor();
    jj_consume_token(COMMA);
    fgcolor = GetColor();
    switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
    case COMMA:
      jj_consume_token(COMMA);
      title = jj_consume_token(STRING_LITERAL);
      break;
    default:
      jj_la1[11] = jj_gen;
      ;
    }
    jj_consume_token(RPAREN);
            ArrayList parameters = new ArrayList(4);

            if(bgcolor != null) {
                parameters.add(new JGSLColor("background", bgcolor));
            }
            else {
                parameters.add(new JGSLColor("background", Color.WHITE));
                ScriptWarning se = new ScriptWarning("Setting canvas background to WHITE.");
                script.addWarning(se);
            }

            if(fgcolor != null) {
                parameters.add(new JGSLColor("foreground", fgcolor));
            }
            else {
                parameters.add(new JGSLColor("foreground", Color.BLACK));
                ScriptWarning se = new ScriptWarning("Setting canvas foreground to BLACK.");
                script.addWarning(se);
            }

            addIntParam(parameters, "width", width);
            addIntParam(parameters, "height", height);

            if(title != null) {
                parameters.add(new JGSLString("title", title.image));
            }
            Command c = new Command(name.image, attributes, parameters);
            script.add(c);
  }

  final public ArrayList CanvasAttributes() throws ParseException {
    Token attrib;
    attrib = jj_consume_token(RED);
        ArrayList attributes = new ArrayList(1);
        {if (true) return attributes;}
    throw new Error("Missing return statement in function");
  }

  final public void Wait() throws ParseException {
    Token name;
    Token durationSeconds;
    name = jj_consume_token(WAIT);
    jj_consume_token(LPAREN);
    switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
    case INTEGER_LITERAL:
      durationSeconds = jj_consume_token(INTEGER_LITERAL);
      break;
    case STRING_LITERAL:
      durationSeconds = jj_consume_token(STRING_LITERAL);
      break;
    default:
      jj_la1[12] = jj_gen;
      jj_consume_token(-1);
      throw new ParseException();
    }
    jj_consume_token(RPAREN);
        ArrayList parameters = new ArrayList(1);
        addIntParam(parameters, "duration", durationSeconds);
        Command c = new Command(name.image, parameters);
        script.add(c);
  }

  final public ArrayList DrawAttributes() throws ParseException {
    Token attrib;
    attrib = jj_consume_token(RED);
        ArrayList attributes = new ArrayList(1);
        {if (true) return attributes;}
    throw new Error("Missing return statement in function");
  }

  final public void Draw() throws ParseException {
    Token name;
    Token x;
    Token y;
    ArrayList attributes = new ArrayList();
    name = jj_consume_token(DRAW);
    switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
    case COLON:
      jj_consume_token(COLON);
      attributes = DrawAttributes();
      break;
    default:
      jj_la1[13] = jj_gen;
      ;
    }
    jj_consume_token(LPAREN);
    switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
    case INTEGER_LITERAL:
      x = jj_consume_token(INTEGER_LITERAL);
      break;
    case STRING_LITERAL:
      x = jj_consume_token(STRING_LITERAL);
      break;
    default:
      jj_la1[14] = jj_gen;
      jj_consume_token(-1);
      throw new ParseException();
    }
    jj_consume_token(COMMA);
    switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
    case INTEGER_LITERAL:
      y = jj_consume_token(INTEGER_LITERAL);
      break;
    case STRING_LITERAL:
      y = jj_consume_token(STRING_LITERAL);
      break;
    default:
      jj_la1[15] = jj_gen;
      jj_consume_token(-1);
      throw new ParseException();
    }
    jj_consume_token(RPAREN);
            ArrayList parameters = new ArrayList(2);
            addIntParam(parameters, "x", x);
            addIntParam(parameters, "y", y);
            addIntParam(parameters, "x", x);
            addIntParam(parameters, "y", y);
            Command c = new Command(name.image, attributes, parameters);
            script.add(c);
  }

//<COLON> (Attributes())? (<LPAREN> Parameters() <RPAREN>)?
  final public void Log() throws ParseException {
    switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
    case LOG:
      Message();
      break;
    case WARNING:
      Warning();
      break;
    case DEBUG:
      Debug();
      break;
    case ERROR:
      Error();
      break;
    default:
      jj_la1[16] = jj_gen;
      jj_consume_token(-1);
      throw new ParseException();
    }
  }

  final public void Message() throws ParseException {
    Token name;
    Token message;
    name = jj_consume_token(LOG);
    jj_consume_token(LPAREN);
    message = jj_consume_token(STRING_LITERAL);
    jj_consume_token(RPAREN);
        ArrayList parameters = new ArrayList(1);
        parameters.add(new JGSLString("message", message.image));
        Command c = new Command(name.image, parameters);
        script.add(c);
  }

```

```
 final public void Warning() throws ParseException {
   Token name;
   Token message;
   name = jj_consume_token(WARNING);
   jj_consume_token(LPAREN);
   message = jj_consume_token(STRING_LITERAL);
   jj_consume_token(RPAREN);
       ArrayList parameters = new ArrayList(1);
       parameters.add(new JGSLString("message", message.image));
       Command c = new Command(name.image, parameters);
       script.add(c);
 }

 final public void Error() throws ParseException {
   Token name;
   Token message;
   name = jj_consume_token(ERROR);
   jj_consume_token(LPAREN);
   message = jj_consume_token(STRING_LITERAL);
   jj_consume_token(RPAREN);
       ArrayList parameters = new ArrayList(1);
       parameters.add(new JGSLString("message", message.image));
       Command c = new Command(name.image, parameters);
       script.add(c);
 }

 final public void Debug() throws ParseException {
   Token name;
   Token message;
   name = jj_consume_token(DEBUG);
   jj_consume_token(LPAREN);
   message = jj_consume_token(STRING_LITERAL);
   jj_consume_token(RPAREN);
       ArrayList parameters = new ArrayList(1);
       parameters.add(new JGSLString("message", message.image));
       Command c = new Command(name.image, parameters);
       script.add(c);
 }

 final public void DrawShape() throws ParseException {
   switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
   case LINE:
     DrawLine();
     break;
   case RECTANGLE:
     DrawRectangle();
     break;
   case SQUARE:
     DrawSquare();
     break;
   case CIRCLE:
     DrawCircle();
     break;
   case ELIPSE:
     DrawElipse();
     break;
   case ARC:
     DrawArc();
     break;
   case POLYGON:
     DrawPolygon();
     break;
   default:
     jj_la1[17] = jj_gen;
     jj_consume_token(-1);
     throw new ParseException();
   }
 }

 final public void DrawLine() throws ParseException {
   Token name;
   Token x1;
   Token y1;
   Token x2;
   Token y2;
   ArrayList attributes = new ArrayList();
   name = jj_consume_token(LINE);
   switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
   case COLON:
     jj_consume_token(COLON);
     attributes = LineAttributes();
     break;
   default:
     jj_la1[18] = jj_gen;
     ;
   }
   jj_consume_token(LPAREN);
   switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
   case INTEGER_LITERAL:
     x1 = jj_consume_token(INTEGER_LITERAL);
     break;
   case STRING_LITERAL:
     x1 = jj_consume_token(STRING_LITERAL);
     break;
   default:
     jj_la1[19] = jj_gen;
     jj_consume_token(-1);
     throw new ParseException();
   }
   jj_consume_token(COMMA);
   switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
   case INTEGER_LITERAL:
     y1 = jj_consume_token(INTEGER_LITERAL);
     break;
   case STRING_LITERAL:
     y1 = jj_consume_token(STRING_LITERAL);
     break;
   default:
     jj_la1[20] = jj_gen;
     jj_consume_token(-1);
```

```
    throw new ParseException();
  }
  jj_consume_token(COMMA);
  switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
  case INTEGER_LITERAL:
    x2 = jj_consume_token(INTEGER_LITERAL);
    break;
  case STRING_LITERAL:
    x2 = jj_consume_token(STRING_LITERAL);
    break;
  default:
    jj_la1[21] = jj_gen;
    jj_consume_token(-1);
    throw new ParseException();
  }
  jj_consume_token(COMMA);
  switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
  case INTEGER_LITERAL:
    y2 = jj_consume_token(INTEGER_LITERAL);
    break;
  case STRING_LITERAL:
    y2 = jj_consume_token(STRING_LITERAL);
    break;
  default:
    jj_la1[22] = jj_gen;
    jj_consume_token(-1);
    throw new ParseException();
  }
  jj_consume_token(RPAREN);
          ArrayList parameters = new ArrayList(4);
          addIntParam(parameters, "x1", x1);
          addIntParam(parameters, "y1", y1);
          addIntParam(parameters, "x2", x2);
          addIntParam(parameters, "y2", y2);
          Command c = new Command(name.image, attributes, parameters);
          script.add(c);
}

final public ArrayList LineAttributes() throws ParseException {
  Color c;
  c = GetColor();
      ArrayList attributes = new ArrayList(1);
      attributes.add(new JGSLColor("c", c));
      {if (true) return attributes;}
  throw new Error("Missing return statement in function");
}

final public void DrawRectangle() throws ParseException {
  Token name;
  Token x1;
  Token y1;
  Token width;
  Token height;
  ArrayList attributes = new ArrayList();
  name = jj_consume_token(RECTANGLE);
  switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
  case COLON:
    jj_consume_token(COLON);
    attributes = RectangleAttributes();
    break;
  default:
    jj_la1[23] = jj_gen;
    ;
  }
  jj_consume_token(LPAREN);
  switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
  case INTEGER_LITERAL:
    x1 = jj_consume_token(INTEGER_LITERAL);
    break;
  case STRING_LITERAL:
    x1 = jj_consume_token(STRING_LITERAL);
    break;
  default:
    jj_la1[24] = jj_gen;
    jj_consume_token(-1);
    throw new ParseException();
  }
  jj_consume_token(COMMA);
  switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
  case INTEGER_LITERAL:
    y1 = jj_consume_token(INTEGER_LITERAL);
    break;
  case STRING_LITERAL:
    y1 = jj_consume_token(STRING_LITERAL);
    break;
  default:
    jj_la1[25] = jj_gen;
    jj_consume_token(-1);
    throw new ParseException();
  }
  jj_consume_token(COMMA);
  switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
  case INTEGER_LITERAL:
    width = jj_consume_token(INTEGER_LITERAL);
    break;
  case STRING_LITERAL:
    width = jj_consume_token(STRING_LITERAL);
    break;
  default:
    jj_la1[26] = jj_gen;
    jj_consume_token(-1);
    throw new ParseException();
  }
  jj_consume_token(COMMA);
  switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
  case INTEGER_LITERAL:
    height = jj_consume_token(INTEGER_LITERAL);
    break;
  case STRING_LITERAL:
    height = jj_consume_token(STRING_LITERAL);
    break;
  default:
    jj_la1[27] = jj_gen;
    jj_consume_token(-1);
    throw new ParseException();
  }
  jj_consume_token(RPAREN);
          ArrayList parameters = new ArrayList(4);
          addIntParam(parameters, "x1", x1);
          addIntParam(parameters, "y1", y1);
          addIntParam(parameters, "width", width);
          addIntParam(parameters, "height", height);
          Command c = new Command(name.image, attributes, parameters);
          script.add(c);
}

final public ArrayList RectangleAttributes() throws ParseException {
  Color c;
  c = GetColor();
      ArrayList attributes = new ArrayList(1);
      attributes.add(new JGSLColor("c", c));
      {if (true) return attributes;}
  throw new Error("Missing return statement in function");
}

final public void DrawSquare() throws ParseException {
  Token name;
  Token x1;
  Token y1;
  Token width;
  ArrayList attributes = new ArrayList();
  name = jj_consume_token(SQUARE);
  switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
  case COLON:
    jj_consume_token(COLON);
    attributes = SquareAttributes();
    break;
  default:
    jj_la1[28] = jj_gen;
    ;
  }
  jj_consume_token(LPAREN);
  switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
  case INTEGER_LITERAL:
    x1 = jj_consume_token(INTEGER_LITERAL);
    break;
  case STRING_LITERAL:
    x1 = jj_consume_token(STRING_LITERAL);
    break;
  default:
    jj_la1[29] = jj_gen;
    jj_consume_token(-1);
    throw new ParseException();
  }
  jj_consume_token(COMMA);
  switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
  case INTEGER_LITERAL:
    y1 = jj_consume_token(INTEGER_LITERAL);
    break;
  case STRING_LITERAL:
    y1 = jj_consume_token(STRING_LITERAL);
    break;
  default:
    jj_la1[30] = jj_gen;
    jj_consume_token(-1);
    throw new ParseException();
  }
  jj_consume_token(COMMA);
  switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
  case INTEGER_LITERAL:
    width = jj_consume_token(INTEGER_LITERAL);
    break;
  case STRING_LITERAL:
    width = jj_consume_token(STRING_LITERAL);
    break;
  default:
    jj_la1[31] = jj_gen;
    jj_consume_token(-1);
    throw new ParseException();
  }
  jj_consume_token(RPAREN);
          ArrayList parameters = new ArrayList(4);
          addIntParam(parameters, "x1", x1);
          addIntParam(parameters, "y1", y1);
          addIntParam(parameters, "width", width);
          Command c = new Command(name.image, attributes, parameters);
          script.add(c);
}

final public ArrayList SquareAttributes() throws ParseException {
  Color c;
  c = GetColor();
      ArrayList attributes = new ArrayList(1);
      attributes.add(new JGSLColor("c", c));
      {if (true) return attributes;}
  throw new Error("Missing return statement in function");
}

final public void DrawCircle() throws ParseException {
  Token name;
  Token x1;
  Token y1;
  Token radius;
  ArrayList attributes = new ArrayList();
  name = jj_consume_token(CIRCLE);
  switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
  case COLON:
    jj_consume_token(COLON);
    attributes = CircleAttributes();
    break;
  default:
    jj_la1[32] = jj_gen;
    ;
  }
  jj_consume_token(LPAREN);
  switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
  case INTEGER_LITERAL:
    x1 = jj_consume_token(INTEGER_LITERAL);
    break;
  case STRING_LITERAL:
    x1 = jj_consume_token(STRING_LITERAL);
    break;
  default:
    jj_la1[33] = jj_gen;
    jj_consume_token(-1);
    throw new ParseException();
    }
  jj_consume_token(COMMA);
  switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
  case INTEGER_LITERAL:
    y1 = jj_consume_token(INTEGER_LITERAL);
    break;
  case STRING_LITERAL:
    y1 = jj_consume_token(STRING_LITERAL);
    break;
  default:
    jj_la1[34] = jj_gen;
    jj_consume_token(-1);
    throw new ParseException();
    }
  jj_consume_token(COMMA);
  switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
  case FLOATING_POINT_LITERAL:
    radius = jj_consume_token(FLOATING_POINT_LITERAL);
    break;
  case STRING_LITERAL:
    radius = jj_consume_token(STRING_LITERAL);
    break;
  default:
    jj_la1[35] = jj_gen;
    jj_consume_token(-1);
    throw new ParseException();
    }
  jj_consume_token(RPAREN);
          ArrayList parameters = new ArrayList(4);
          addIntParam(parameters, "x1", x1);
          addIntParam(parameters, "y1", y1);
          addDecimalParam(parameters, "radius", radius);
          Command c = new Command(name.image, attributes, parameters);
          script.add(c);
  }

  final public ArrayList CircleAttributes() throws ParseException {
  Color c;
  c = GetColor();
      ArrayList attributes = new ArrayList(1);
      attributes.add(new JGSLColor("c", c));
      {if (true) return attributes;}
  throw new Error("Missing return statement in function");
  }

  final public void DrawElipse() throws ParseException {
  Token name;
  Token x1;
  Token y1;
  Token width;
  Token height;
  ArrayList attributes = new ArrayList();
  name = jj_consume_token(ELIPSE);
  switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
  case COLON:
    jj_consume_token(COLON);
    attributes = ElipseAttributes();
    break;
  default:
    jj_la1[36] = jj_gen;
    ;
    }
  jj_consume_token(LPAREN);
  switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
  case INTEGER_LITERAL:
    x1 = jj_consume_token(INTEGER_LITERAL);
    break;
  case STRING_LITERAL:
    x1 = jj_consume_token(STRING_LITERAL);
    break;
  default:
    jj_la1[37] = jj_gen;
    jj_consume_token(-1);
    throw new ParseException();
    }
  jj_consume_token(COMMA);
  switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
  case INTEGER_LITERAL:
    y1 = jj_consume_token(INTEGER_LITERAL);
    break;
  case STRING_LITERAL:
    y1 = jj_consume_token(STRING_LITERAL);
    break;
  default:
    jj_la1[38] = jj_gen;
    jj_consume_token(-1);
    throw new ParseException();
    }
  jj_consume_token(COMMA);
  switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
  case INTEGER_LITERAL:
    width = jj_consume_token(INTEGER_LITERAL);
    break;
  case STRING_LITERAL:
    width = jj_consume_token(STRING_LITERAL);
    break;
  default:
    jj_la1[39] = jj_gen;
    jj_consume_token(-1);
    throw new ParseException();
  }
  jj_consume_token(COMMA);
  switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
  case INTEGER_LITERAL:
    height = jj_consume_token(INTEGER_LITERAL);
    break;
  case STRING_LITERAL:
    height = jj_consume_token(STRING_LITERAL);
    break;
  default:
    jj_la1[40] = jj_gen;
    jj_consume_token(-1);
    throw new ParseException();
  }
  jj_consume_token(RPAREN);
          ArrayList parameters = new ArrayList(4);
          addIntParam(parameters, "x1", x1);
          addIntParam(parameters, "y1", y1);
          addIntParam(parameters, "width", width);
          addIntParam(parameters, "height", height);
          Command c = new Command(name.image, attributes, parameters);
          script.add(c);
}

final public ArrayList ElipseAttributes() throws ParseException {
  Color c;
  c = GetColor();
      ArrayList attributes = new ArrayList(1);
      attributes.add(new JGSLColor("c", c));
      {if (true) return attributes;}
  throw new Error("Missing return statement in function");
}

final public void DrawArc() throws ParseException {
  Token name;
  Token x1;
  Token y1;
  Token width;
  Token height;
  Token startAngle;
  Token arcAngle;
  ArrayList attributes = new ArrayList();
  name = jj_consume_token(ARC);
  switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
  case COLON:
    jj_consume_token(COLON);
    attributes = ArcAttributes();
    break;
  default:
    jj_la1[41] = jj_gen;
    ;
  }
  jj_consume_token(LPAREN);
  switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
  case INTEGER_LITERAL:
    x1 = jj_consume_token(INTEGER_LITERAL);
    break;
  case STRING_LITERAL:
    x1 = jj_consume_token(STRING_LITERAL);
    break;
  default:
    jj_la1[42] = jj_gen;
    jj_consume_token(-1);
    throw new ParseException();
  }
  jj_consume_token(COMMA);
  switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
  case INTEGER_LITERAL:
    y1 = jj_consume_token(INTEGER_LITERAL);
    break;
  case STRING_LITERAL:
    y1 = jj_consume_token(STRING_LITERAL);
    break;
  default:
    jj_la1[43] = jj_gen;
    jj_consume_token(-1);
    throw new ParseException();
  }
  jj_consume_token(COMMA);
  switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
  case INTEGER_LITERAL:
    width = jj_consume_token(INTEGER_LITERAL);
    break;
  case STRING_LITERAL:
    width = jj_consume_token(STRING_LITERAL);
    break;
  default:
    jj_la1[44] = jj_gen;
    jj_consume_token(-1);
    throw new ParseException();
  }
  jj_consume_token(COMMA);
  switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
  case INTEGER_LITERAL:
    height = jj_consume_token(INTEGER_LITERAL);
    break;
  case STRING_LITERAL:
    height = jj_consume_token(STRING_LITERAL);
    break;
  default:
    jj_la1[45] = jj_gen;
    jj_consume_token(-1);
    throw new ParseException();
  }
  jj_consume_token(COMMA);
  switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
  case INTEGER_LITERAL:
    startAngle = jj_consume_token(INTEGER_LITERAL);
    break;
  case STRING_LITERAL:
    startAngle = jj_consume_token(STRING_LITERAL);
    break;
  default:
    jj_la1[46] = jj_gen;
    jj_consume_token(-1);
    throw new ParseException();
  }
  jj_consume_token(COMMA);
  switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
  case INTEGER_LITERAL:
    arcAngle = jj_consume_token(INTEGER_LITERAL);
    break;
  case STRING_LITERAL:
    arcAngle = jj_consume_token(STRING_LITERAL);
    break;
  default:
    jj_la1[47] = jj_gen;
    jj_consume_token(-1);
    throw new ParseException();
  }
  jj_consume_token(RPAREN);
          ArrayList parameters = new ArrayList(4);
          addIntParam(parameters, "x1", x1);
          addIntParam(parameters, "y1", y1);
          addIntParam(parameters, "width", width);
          addIntParam(parameters, "height", height);
          addIntParam(parameters, "startAngle", startAngle);
          addIntParam(parameters, "arcAngle", arcAngle);
          Command c = new Command(name.image, attributes, parameters);
          script.add(c);
}

final public ArrayList ArcAttributes() throws ParseException {
  Color c;
  c = GetColor();
      ArrayList attributes = new ArrayList(1);
      attributes.add(new JGSLColor("c", c));
      {if (true) return attributes;}
  throw new Error("Missing return statement in function");
}

final public void DrawPolygon() throws ParseException {
  Token name;
  Token x1;
  Token y1;
  Token width;
  Token height;
  Token startAngle;
  Token arcAngle;
  ArrayList attributes = new ArrayList();
  ArrayList parameters = new ArrayList();
  name = jj_consume_token(POLYGON);
  switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
  case COLON:
    jj_consume_token(COLON);
    attributes = PolygonAttributes();
    break;
  default:
    jj_la1[48] = jj_gen;
    ;
  }
  jj_consume_token(LPAREN);
  PolygonParameters(parameters);
  jj_consume_token(RPAREN);
          Command c = new Command(name.image, attributes, parameters);
          script.add(c);
}

final public ArrayList PolygonAttributes() throws ParseException {
  Color c;
  c = GetColor();
      ArrayList attributes = new ArrayList(1);
      attributes.add(new JGSLColor("c", c));
      {if (true) return attributes;}
  throw new Error("Missing return statement in function");
}

final public void PolygonParameters(ArrayList parameters) throws ParseException {
  Token x1 = null;
  Token y1 = null;
  label_3:
  while (true) {
    switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
    case INTEGER_LITERAL:
    case STRING_LITERAL:
      ;
      break;
    default:
      jj_la1[49] = jj_gen;
      break label_3;
    }
    switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
    case INTEGER_LITERAL:
      x1 = jj_consume_token(INTEGER_LITERAL);
      break;
    case STRING_LITERAL:
      x1 = jj_consume_token(STRING_LITERAL);
      break;
    default:
      jj_la1[50] = jj_gen;
      jj_consume_token(-1);
      throw new ParseException();
    }
    jj_consume_token(COMMA);
    switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
    case INTEGER_LITERAL:
      y1 = jj_consume_token(INTEGER_LITERAL);
      break;
    case STRING_LITERAL:
      y1 = jj_consume_token(STRING_LITERAL);
      break;
    default:
      jj_la1[51] = jj_gen;
      jj_consume_token(-1);
      throw new ParseException();
    }
    switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
    case COMMA:
      jj_consume_token(COMMA);
      break;
    default:
      jj_la1[52] = jj_gen;
      ;
    }
  }
      addIntParam(parameters, "x", x1);
      addIntParam(parameters, "y", y1);
}

final public void DrawText() throws ParseException {
  Token name;
  Token x1;
  Token y1;
  Token text;
  ArrayList attributes = new ArrayList();
  name = jj_consume_token(TEXT);
  switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
  case COLON:
    jj_consume_token(COLON);
    attributes = TextAttributes();
    break;
  default:
    jj_la1[53] = jj_gen;
    ;
  }
  jj_consume_token(LPAREN);
  switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
  case INTEGER_LITERAL:
    x1 = jj_consume_token(INTEGER_LITERAL);
    break;
  case STRING_LITERAL:
    x1 = jj_consume_token(STRING_LITERAL);
    break;
  default:
    jj_la1[54] = jj_gen;
    jj_consume_token(-1);
    throw new ParseException();
  }
  jj_consume_token(COMMA);
  switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
  case INTEGER_LITERAL:
    y1 = jj_consume_token(INTEGER_LITERAL);
    break;
  case STRING_LITERAL:
    y1 = jj_consume_token(STRING_LITERAL);
    break;
  default:
    jj_la1[55] = jj_gen;
    jj_consume_token(-1);
    throw new ParseException();
  }
  jj_consume_token(COMMA);
  text = jj_consume_token(STRING_LITERAL);
  jj_consume_token(RPAREN);
          ArrayList parameters = new ArrayList(3);
          addIntParam(parameters, "x1", x1);
          addIntParam(parameters, "y1", y1);
          parameters.add(new JGSLString("text", text.image));

          Command c = new Command(name.image, attributes, parameters);
          script.add(c);
}

void addIntParam(ArrayList parameters, String paramName, Token t) throws ParseException {
  try {
      if(t == null) {
          String msg = "Unable to convert value to a number.";
          ScriptError se = new ScriptError(msg, t.beginLine, t.beginColumn);
          script.addError(se);
      }
      parameters.add(new JGSLInteger(paramName, t.image));
  }
  catch(NumberFormatException e) {
      String msg = "Unable to convert value " + t.image + " to a number.";
      ScriptError se = new ScriptError(msg, t.beginLine, t.beginColumn);
      script.addError(se);
  }
}

void addDecimalParam(ArrayList parameters, String paramName, Token t) throws ParseException {
  try {
      parameters.add(new JGSLDouble(paramName, t.image));
  }
  catch(NumberFormatException e) {
      String msg = "Unable to convert value " + t.image + " to a number.";
      ScriptError se = new ScriptError(msg, t.beginLine, t.beginColumn);
      script.addError(se);
  }
}

final public ArrayList TextAttributes() throws ParseException {
  Color c;
  c = GetColor();
      ArrayList attributes = new ArrayList(1);
      attributes.add(new JGSLColor("c", c));
      {if (true) return attributes;}
  throw new Error("Missing return statement in function");
}

final public void Declaration() throws ParseException {
  jj_consume_token(DECLARE);
  switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
  case COLOR:
    DeclareColor();
    break;
  case IDENTIFIER:
    jj_consume_token(IDENTIFIER);
    switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
    case ASSIGN:
      jj_consume_token(ASSIGN);
      switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
      case IDENTIFIER:
        jj_consume_token(IDENTIFIER);
        break;
      case INTEGER_LITERAL:
        jj_consume_token(INTEGER_LITERAL);
        break;
      case STRING_LITERAL:
        jj_consume_token(STRING_LITERAL);
        break;
      case FLOATING_POINT_LITERAL:
        jj_consume_token(FLOATING_POINT_LITERAL);
        break;
      default:
        jj_la1[56] = jj_gen;
        jj_consume_token(-1);
        throw new ParseException();
      }
      break;
    default:
      jj_la1[57] = jj_gen;
      ;
    }
    break;
  default:
    jj_la1[58] = jj_gen;
    jj_consume_token(-1);
    throw new ParseException();
  }
  jj_consume_token(SEMICOLON);
}

final public void DeclareCanvas() throws ParseException {
  Token type;
  Token id;
  Token tokenValue = null;
  type = jj_consume_token(CANVAS);
  id = jj_consume_token(IDENTIFIER);
  switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
  case ASSIGN:
    jj_consume_token(ASSIGN);
    tokenValue = jj_consume_token(CANVAS);
    break;
  default:
    jj_la1[59] = jj_gen;
    ;
  }
      String value = null;
      if(tokenValue != null) {
          value = tokenValue.image;
      }
      Declaration d = new Declaration(type.image, id.image, value);
      script.add(d);
}

final public void DeclareColor() throws ParseException {
  Token type;
  Token id;
  type = jj_consume_token(COLOR);
  id = jj_consume_token(IDENTIFIER);
  switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
  case ASSIGN:
    jj_consume_token(ASSIGN);
    switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
    case BLACK:
    case BLUE:
    case DARK_GRAY:
    case GRAY:
    case GREEN:
    case LIGHT_GRAY:
    case MAGENTA:
    case ORANGE:
    case PINK:
    case RED:
    case WHITE:
    case YELLOW:
      DeclareStandardColor(type, id);
      break;
    case INTEGER_LITERAL:
    case STRING_LITERAL:
      DeclareRGB(type, id);
      break;
    default:
      jj_la1[60] = jj_gen;
      jj_consume_token(-1);
      throw new ParseException();
    }
    break;
  default:
    jj_la1[61] = jj_gen;
    ;
  }
}

final public void DeclareStandardColor(Token type, Token id) throws ParseException {
  Token value;
  switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
  case BLACK:
    value = jj_consume_token(BLACK);
    break;
  case BLUE:
    value = jj_consume_token(BLUE);
    break;
  case DARK_GRAY:
    value = jj_consume_token(DARK_GRAY);
    break;
  case GRAY:
    value = jj_consume_token(GRAY);
    break;
  case GREEN:
    value = jj_consume_token(GREEN);
    break;
  case LIGHT_GRAY:
    value = jj_consume_token(LIGHT_GRAY);
    break;
  case MAGENTA:
    value = jj_consume_token(MAGENTA);
    break;
  case ORANGE:
    value = jj_consume_token(ORANGE);
    break;
  case PINK:
    value = jj_consume_token(PINK);
    break;
  case RED:
    value = jj_consume_token(RED);
    break;
  case WHITE:
    value = jj_consume_token(WHITE);
    break;
  case YELLOW:
    value = jj_consume_token(YELLOW);
    break;
  default:
    jj_la1[62] = jj_gen;
    jj_consume_token(-1);
    throw new ParseException();
  }
      Declaration d = new Declaration(type.image, id.image, value.image);
      script.add(d);
}

final public void DeclareRGB(Token type, Token id) throws ParseException {
  Token r;
  Token g;
  Token b;
  switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
  case INTEGER_LITERAL:
    r = jj_consume_token(INTEGER_LITERAL);
    jj_consume_token(COMMA);
    g = jj_consume_token(INTEGER_LITERAL);
    jj_consume_token(COMMA);
    b = jj_consume_token(INTEGER_LITERAL);
    break;
  case STRING_LITERAL:
    r = jj_consume_token(STRING_LITERAL);
    jj_consume_token(COMMA);
    g = jj_consume_token(STRING_LITERAL);
    jj_consume_token(COMMA);
    b = jj_consume_token(STRING_LITERAL);
    break;
  default:
    jj_la1[63] = jj_gen;
    jj_consume_token(-1);
    throw new ParseException();
  }
      String rgb = r.image + "," + g.image + "," + b.image;
      Declaration d = new Declaration(type.image, id.image, rgb);
      script.add(d);
}

final public void Assignment() throws ParseException {
  Token lhs;
  Token rhs;
  lhs = jj_consume_token(IDENTIFIER);
  jj_consume_token(ASSIGN);
  switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
  case IDENTIFIER:
    rhs = jj_consume_token(IDENTIFIER);
    break;
  case INTEGER_LITERAL:
    rhs = jj_consume_token(INTEGER_LITERAL);
    break;
  case STRING_LITERAL:
    rhs = jj_consume_token(STRING_LITERAL);
    break;
  default:
    jj_la1[64] = jj_gen;
    jj_consume_token(-1);
    throw new ParseException();
  }
      Assignment a = new Assignment(lhs.image, rhs.image);
      script.add(a);
  jj_consume_token(SEMICOLON);
}

public JGSL_ParserTokenManager token_source;
JavaCharStream jj_input_stream;
public Token token, jj_nt;
private int jj_ntk;
private int jj_gen;
final private int[] jj_la1 = new int[65];
static private int[] jj_la1_0;
static private int[] jj_la1_1;
static private int[] jj_la1_2;
static {
    jj_la1_0();
    jj_la1_1();
    jj_la1_2();
 }
 private static void jj_la1_0() {
    jj_la1_0 = new int[] {0x0,0x0,0xff800000,0xff800000,0xff800000,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0xf8000000,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,};
 }
 private static void jj_la1_1() {
      jj_la1_1 = new int[] {0x0,0x800,0x7f,0x7f,0x7f,0x7ff8000,0x0,0x7ff8000,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x78,0x3,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x7ff8000,0x0,0x7ff8000,0x0,0x0,};
   }
   private static void jj_la1_2() {
      jj_la1_2 = new int[] 
{0x4,0x0,0x206,0x206,0x0,0x0,0x108,0x108,0x8000,0x108,0x108,0x10000,0x108,0x8000,0x108,0x108,0x0,0x0,0x8000,0x108,0x108,0x108,0x108,0x8000,0x108,0x108,0x108,0x108,0x8000,0x108,0x108,0x108,0x8000,0x108,0x108,0x180,0x8000,0x108,0x108,0x108,0x108,0x8000,0x108,0x108,0x108,0x108,0x108,0x108,0x8000,0x108,0x108,0x108,0x10000,0x8000,0x108,0x108,0x388,0x20000,0x201,0x20000,0x108,0x20000,0x0,0x108,0x308,};
   }

  public JGSL_Parser(java.io.InputStream stream) {
    jj_input_stream = new JavaCharStream(stream, 1, 1);
    token_source = new JGSL_ParserTokenManager(jj_input_stream);
    token = new Token();
    jj_ntk = -1;
    jj_gen = 0;
    for (int i = 0; i < 65; i++) jj_la1[i] = -1;
  }

  public void ReInit(java.io.InputStream stream) {
    jj_input_stream.ReInit(stream, 1, 1);
    token_source.ReInit(jj_input_stream);
    token = new Token();
    jj_ntk = -1;
    jj_gen = 0;
    for (int i = 0; i < 65; i++) jj_la1[i] = -1;
  }

  public JGSL_Parser(java.io.Reader stream) {
    jj_input_stream = new JavaCharStream(stream, 1, 1);
    token_source = new JGSL_ParserTokenManager(jj_input_stream);
    token = new Token();
    jj_ntk = -1;
    jj_gen = 0;
    for (int i = 0; i < 65; i++) jj_la1[i] = -1;
  }

  public void ReInit(java.io.Reader stream) {
    jj_input_stream.ReInit(stream, 1, 1);
    token_source.ReInit(jj_input_stream);
    token = new Token();
    jj_ntk = -1;
    jj_gen = 0;
    for (int i = 0; i < 65; i++) jj_la1[i] = -1;
  }

  public JGSL_Parser(JGSL_ParserTokenManager tm) {
    token_source = tm;
    token = new Token();
    jj_ntk = -1;
    jj_gen = 0;
    for (int i = 0; i < 65; i++) jj_la1[i] = -1;
  }

  public void ReInit(JGSL_ParserTokenManager tm) {
    token_source = tm;
    token = new Token();
    jj_ntk = -1;
    jj_gen = 0;
    for (int i = 0; i < 65; i++) jj_la1[i] = -1;
  }

  final private Token jj_consume_token(int kind) throws ParseException {
    Token oldToken;
    if ((oldToken = token).next != null) token = token.next;
    else token = token.next = token_source.getNextToken();
    jj_ntk = -1;
    if (token.kind == kind) {
      jj_gen++;
      return token;
    }
    token = oldToken;
    jj_kind = kind;
    throw generateParseException();
  }

  final public Token getNextToken() {
    if (token.next != null) token = token.next;
    else token = token.next = token_source.getNextToken();
    jj_ntk = -1;
    jj_gen++;
    return token;
  }

  final public Token getToken(int index) {
    Token t = token;
    for (int i = 0; i < index; i++) {
      if (t.next != null) t = t.next;
      else t = t.next = token_source.getNextToken();
    }
    return t;
  }

  final private int jj_ntk() {
    if ((jj_nt=token.next) == null)
      return (jj_ntk = (token.next=token_source.getNextToken()).kind);
    else
      return (jj_ntk = jj_nt.kind);
  }

  private java.util.Vector jj_expentries = new java.util.Vector();
  private int[] jj_expentry;
  private int jj_kind = -1;

  public ParseException generateParseException() {
    jj_expentries.removeAllElements();
    boolean[] la1tokens = new boolean[96];
    for (int i = 0; i < 96; i++) {
      la1tokens[i] = false;
    }
    if (jj_kind >= 0) {
      la1tokens[jj_kind] = true;
      jj_kind = -1;
    }
    for (int i = 0; i < 65; i++) {
      if (jj_la1[i] == jj_gen) {
        for (int j = 0; j < 32; j++) {
          if ((jj_la1_0[i] & (1<<j)) != 0) {
            la1tokens[j] = true;
          }
          if ((jj_la1_1[i] & (1<<j)) != 0) {
            la1tokens[32+j] = true;
          }
          if ((jj_la1_2[i] & (1<<j)) != 0) {
            la1tokens[64+j] = true;
          }
        }
      }
    }
    for (int i = 0; i < 96; i++) {
      if (la1tokens[i]) {
        jj_expentry = new int[1];
        jj_expentry[0] = i;
        jj_expentries.addElement(jj_expentry);
      }
    }
    int[][] exptokseq = new int[jj_expentries.size()][];
    for (int i = 0; i < jj_expentries.size(); i++) {
      exptokseq[i] = (int[])jj_expentries.elementAt(i);
    }
    return new ParseException(token, exptokseq, tokenImage);
  }

  final public void enable_tracing() {
  }

  final public void disable_tracing() {
  }

}
```

/Users/jchavez/dev/java.net/jgsl.dev.java.net/jgsl/dev/src/jgsl/parser/ParseException.java

```
    /* Generated By:JavaCC: Do not edit this line. ParseException.java Version 3.0 */
package jgsl.parser;

/**
 * This exception is thrown when parse errors are encountered. You can explicitly create objects of this exception type
 * by calling the method generateParseException in the generated parser.
 * <p/>
 * You can modify this class to customize your error reporting mechanisms so long as you retain the public fields.
 */
public class ParseException extends Exception {
    /**
     * This constructor is used by the method "generateParseException" in the generated parser.  Calling this constructor
     * generates a new object of this type with the fields "currentToken", "expectedTokenSequences", and "tokenImage" set.
     *  The boolean flag "specialConstructor" is also set to true to indicate that this constructor was used to create
     * this object. This constructor calls its super class with the empty string to force the "toString" method of parent
     * class "Throwable" to print the error message in the form: ParseException: <result of getMessage>
     */
    public ParseException(Token currentTokenVal,
                          int[][] expectedTokenSequencesVal,
                          String[] tokenImageVal
    ) {
        super("");
        specialConstructor = true;
        currentToken = currentTokenVal;
        expectedTokenSequences = expectedTokenSequencesVal;
        tokenImage = tokenImageVal;
    }

    /**
     * The following constructors are for use by you for whatever purpose you can think of.  Constructing the exception in
     * this manner makes the exception behave in the normal way - i.e., as documented in the class "Throwable".  The
     * fields "errorToken", "expectedTokenSequences", and "tokenImage" do not contain relevant information.  The
JavaCC
     * generated code does not use these constructors.
     */

    public ParseException() {
        super();
        specialConstructor = false;
    }

    public ParseException(String message) {
        super(message);
        specialConstructor = false;
    }

    /**
     * This variable determines which constructor was used to create this object and thereby affects the semantics of the
```

```
   * "getMessage" method (see below).
   */
  protected boolean specialConstructor;

  /**
   * This is the last token that has been consumed successfully.  If this object has been created due to a parse error,
   * the token followng this token will (therefore) be the first error token.
   */
  public Token currentToken;

  /**
   * Each entry in this array is an array of integers.  Each array of integers represents a sequence of tokens (by their
   * ordinal values) that is expected at this point of the parse.
   */
  public int[][] expectedTokenSequences;

  /**
   * This is a reference to the "tokenImage" array of the generated parser within which the parse error occurred.  This
   * array is defined in the generated ...Constants interface.
   */
  public String[] tokenImage;

  /**
   * This method has the standard behavior when this object has been created using the standard constructors.
   * Otherwise, it uses "currentToken" and "expectedTokenSequences" to generate a parse error message and returns it.
   * If this object has been created due to a parse error, and you do not catch it (it gets thrown from the parser),
   * then this method is called during the printing of the final stack trace, and hence the correct error message gets
   * displayed.
   */
  public String getMessage() {
    if (!specialConstructor) {
      return super.getMessage();
    }
    String expected = "";
    int maxSize = 0;
    for (int i = 0; i < expectedTokenSequences.length; i++) {
      if (maxSize < expectedTokenSequences[i].length) {
        maxSize = expectedTokenSequences[i].length;
      }
      for (int j = 0; j < expectedTokenSequences[i].length; j++) {
        expected += tokenImage[expectedTokenSequences[i][j]] + " ";
      }
      if (expectedTokenSequences[i][expectedTokenSequences[i].length - 1] != 0) {
        expected += "...";
      }
      expected += eol + "    ";
    }
    String retval = "Encountered \"";
    Token tok = currentToken.next;
    for (int i = 0; i < maxSize; i++) {
      if (i != 0) retval += " ";
      if (tok.kind == 0) {
          retval += tokenImage[0];
          break;
        }
        retval += add_escapes(tok.image);
        tok = tok.next;
      }
      retval += "\" at line " + currentToken.next.beginLine + ", column " + currentToken.next.beginColumn;
      retval += "." + eol;
      if (expectedTokenSequences.length == 1) {
        retval += "Was expecting:" + eol + "    ";
      } else {
        retval += "Was expecting one of:" + eol + "    ";
      }
      retval += expected;
      return retval;
    }

    /**
     * The end of line string for this machine.
     */
    protected String eol = System.getProperty("line.separator", "\n");

    /**
     * Used to convert raw characters to their escaped version when these raw version cannot be used as part of an ASCII
     * string literal.
     */
    protected String add_escapes(String str) {
      StringBuffer retval = new StringBuffer();
      char ch;
      for (int i = 0; i < str.length(); i++) {
        switch (str.charAt(i)) {
          case 0 :
            continue;
          case '\b':
            retval.append("\\b");
            continue;
          case '\t':
            retval.append("\\t");
            continue;
          case '\n':
            retval.append("\\n");
            continue;
          case '\f':
            retval.append("\\f");
            continue;
          case '\r':
            retval.append("\\r");
            continue;
          case '\"':
            retval.append("\\\"");
            continue;
          case '\'':
```

```
                    retval.append("\\\'");
                    continue;
                case '\\':
                    retval.append("\\\\");
                    continue;
                default:
                    if ((ch = str.charAt(i)) < 0x20 || ch > 0x7e) {
                        String s = "0000" + Integer.toString(ch, 16);
                        retval.append("\\u" + s.substring(s.length() - 4, s.length()));
                    } else {
                        retval.append(ch);
                    }
                    continue;
            }
        }
        return retval.toString();
    }

}

```

/Users/jchavez/dev/java.net/jgsl.dev.java.net/jgsl/dev/src/jgsl/io/ScriptError.java

```java
     /*
 * Copyright (c) 2005 Perception Software. All Rights Reserved.
 */
package jgsl.io;

/**
 * A single script error with line and column info.
 *
 * @author zenarchitect
 * @version $Id: ScriptError.java,v 1.2 2005/05/16 00:54:16 zenarchitect Exp $
 */
public class ScriptError implements Message {
    private String message;
    private int lineNumber;
    private int colNumber;

    public ScriptError(String message) {
        this.message = message;
    }


    public ScriptError(String message, int lineNumber, int colNumber) {
        this.message = message;
        this.lineNumber = lineNumber;
        this.colNumber = colNumber;
    }

    /**
     * Returns the type of message
     *
     * @return MessageType
     */
    public MessageType getType() {
        return Message.MessageType.ERROR;
    }

    /**
     * Retuns the message
     *
     * @return A string containing a simple message
     */
    public String getMessage() {
        return "\n>>>>> " + getType() + ": at line " + lineNumber + ", column " + colNumber + "\n\t" + message +
"\n>>>>>\n";
    }

    /**
     * Return a detailed message
     *
     * @return A string containing a detailed message
     */
    public String getDetailMessage() {
        return null;  //TODO: To change body of implemented methods use File | Settings | File Templates.
    }
}

```

/Users/jchavez/dev/java.net/jgsl.dev.java.net/jgsl/dev/src/jgsl/io/ScriptMessage.java

```
/*
 * Copyright (c) 2005 Perception Software. All Rights Reserved.
 */
package jgsl.io;

/**
 * A general script message generated by the parser.
 *
 * @author zenarchitect
 * @version $Id: ScriptMessage.java,v 1.2 2005/05/16 00:54:16 zenarchitect Exp $
 */
public class ScriptMessage implements Message {
    private String message;

    public ScriptMessage(String message) {
        this.message = message;
    }

    /**
     * Returns the type of message
     *
     * @return MessageType
     */
    public MessageType getType() {
        return Message.MessageType.MESSAGE;
    }

    /**
     * Retuns the message
     *
     * @return A string containing a simple message
     */
    public String getMessage() {
        return ">>>>> " + getType() + ":\n" + message + ">>>>>\n\n";
    }

    /**
     * Return a detailed message
     *
     * @return A string containing a detailed message
     */
    public String getDetailMessage() {
        return null;  //TODO: To change body of implemented methods use File | Settings | File Templates.
    }
}

```

/Users/jchavez/dev/java.net/jgsl.dev.java.net/jgsl/dev/src/jgsl/io/Message.java

```
/*
 * Copyright (c) 2005 Perception Software. All Rights Reserved.
 */
package jgsl.io;

/**
 * Interface for message types
 *
 * @author ZenArchitect
 * @version $Id: Message.java,v 1.2 2005/05/16 00:54:16 zenarchitect Exp $
 */
public interface Message {
    /**
     * Message types
     */
    public enum MessageType { MESSAGE, WARNING, ERROR }

    ;

    /**
     * Returns the type of message
     *
     * @return MessageType
     */
    public MessageType getType();

    /**
     * Retuns the message
     *
     * @return A string containing a simple message
     */
    public String getMessage();

    /**
     * Return a detailed message
     *
     * @return A string containing a detailed message
     */
    public String getDetailMessage();

}

```

/Users/jchavez/dev/java.net/jgsl.dev.java.net/jgsl/dev/src/jgsl/io/ParseStatus.java

```
/*
 * Copyright (c) 2005 Perception Software. All Rights Reserved.
 */
package jgsl.io;

/**
 * ParseStatus interface for collecting and reporting errors and error counts.
 *
 * @author zenarchitect
 * @version $Id: ParseStatus.java,v 1.2 2005/05/16 00:54:16 zenarchitect Exp $
 */
public interface ParseStatus {
    /**
     * Add a ScriptError to the parse status
     *
     * @param se
     */
    public void addError(ScriptError se);

    /**
     * Add a ScriptWarning to the parse status
     *
     * @param sw
     */
    public void addWarning(ScriptWarning sw);

    /**
     * Add a ScriptMessage to the parse status
     *
     * @param sm
     */
    public void addMessage(ScriptMessage sm);

    /**
     * Return the error state of the script
     *
     * @return true of the script contains errors or false otherwise
     */
    boolean hasErrors();

    /**
     * Return the warning state of the script
     *
     * @return true of the script contains warnings or false otherwise
     */
    boolean hasWarnings();

    /**
     * Return the message state of the script
     *
     * @return true of the script contains messages or false otherwise
     */
    boolean hasMessages();
}

```

/Users/jchavez/dev/java.net/jgsl.dev.java.net/jgsl/dev/src/jgsl/io/ScriptWarning.java

```
/*
 * Copyright (c) 2005 Perception Software. All Rights Reserved.
 */
package jgsl.io;

/**
 * Record a script warning message.
 *
 * @author zenarchitect
 * @version $Id: ScriptWarning.java,v 1.2 2005/05/16 00:54:17 zenarchitect Exp $
 */
public class ScriptWarning implements Message {
    private String message;
    private int lineNumber = -1;
    private int colNumber = -1;

    public ScriptWarning(String message) {
        this.message = message;
    }

    public ScriptWarning(String message, int lineNumber, int colNumber) {
        this.message = message;
        this.lineNumber = lineNumber;
        this.colNumber = colNumber;
    }

    /**
     * Returns the type of message
     *
     * @return MessageType
     */
    public MessageType getType() {
        return Message.MessageType.WARNING;
    }

    /**
     * Retuns the message
     *
     * @return A string containing a simple message
     */
    public String getMessage() {
```

```java
        return ">>>>> " + getType() + ":\n" + message + ">>>>>\n\n";
    }

    /**
     * Return a detailed message
     *
     * @return A string containing a detailed message
     */
    public String getDetailMessage() {
        return null;  //TODO: To change body of implemented methods use File | Settings | File Templates.
    }
}
```

/Users/jchavez/dev/java.net/jgsl.dev.java.net/jgsl/dev/src/jgsl/model/Script.java

```
/*
 * Copyright (c) 2005 Perception Software. All Rights Reserved.
 */

package jgsl.model;


/**
 * The script interface provide the set of operations for a script
 *
 * @author zenarchitect
 * @version $Id: Script.java,v 1.2 2005/05/16 00:54:19 zenarchitect Exp $
 */
public interface Script {
    /**
     * Get the script name
     *
     * @return String containing the script name
     */
    String getScriptName();

    /**
     * Set the scipt name
     *
     * @param scriptName name of the script file
     */
    void setScriptName(String scriptName);

    /**
     * Return the Java implementation of this script
     *
     * @return the Java language implementation of this script
     */
    String getJava();

    /**
     * Returns the JGSL script documentation as specified in the DOC keyword by the script author.
     *
     * @return The script documentation
     */
    String getDocumentation();

}

```

/Users/jchavez/dev/java.net/jgsl.dev.java.net/jgsl/dev/src/jgsl/model/Documentation.java

```
 /*
 * Copyright (c) 2005 Perception Software. All Rights Reserved.
 */

package jgsl.model;

import java.io.Serializable;
import java.util.ArrayList;

/**
 * A documentation statement is one that contains documentation of the JGSL script as written by the script author.
 *
 * @author zenarchitect
 * @version $Id: Documentation.java,v 1.3 2005/05/16 00:54:18 zenarchitect Exp $
 */
public class Documentation implements Statement, Serializable {
    private ArrayList<String> docs;

    public Documentation() {
        docs = new ArrayList<String>();
    }

    public void addDoc(String doc) {
        docs.add(doc);
    }

    /**
     * This method returns the Java language equivalent of the JGSL statement.
     *
     * @return Java language statement from the JGSL
     */
    public String getJava() {
        StringBuilder docString = new StringBuilder();

        for (String doc : docs) {
            docString.append(doc);
        }

        return docString.toString();
    }

    /**
     * Set the JGSL statement body
     */
```

```java
    public void setJGSL(String jgsl) {
        //TODO: To change body of implemented methods use File | Settings | File Templates.
    }

    /**
     * Return the type of statement. The String form of the class name.
     */
    public String getType() {
        return "Documentation";
    }
}

```

/Users/jchavez/dev/java.net/jgsl.dev.java.net/jgsl/dev/src/jgsl/model/Statement.java

```
    /*
 * Copyright (c) 2005 Perception Software. All Rights Reserved.
 */

package jgsl.model;

/**
 * The Statement interface provides the set of operations common to all JGSL statements.
 *
 * @author zenarchitect
 * @version $Id: Statement.java,v 1.2 2005/05/16 00:54:19 zenarchitect Exp $
 */
public interface Statement {
    /**
     * This method returns the Java language equivalent of the JGSL statement.
     *
     * @return Java language statement from the JGSL
     */
    String getJava();

    /**
     * Set the JGSL statement body
     */
    void setJGSL(String jgsl);

    /**
     * Return the type of statement. The String form of the class name.
     */
    String getType();
}

```

/Users/jchavez/dev/java.net/jgsl.dev.java.net/jgsl/dev/src/jgsl/model/Commands.java

```java
    /*
 * Copyright (c) 2005 Perception Software. All Rights Reserved.
 */
package jgsl.model;

import java.util.ArrayList;
import java.util.ResourceBundle;

/**
 * Enum of all possible commands and their corresponding command template and java code representation.
 *
 * @author zenarchitect
 * @version $Id: Commands.java,v 1.3 2005/05/21 01:42:07 zenarchitect Exp $
 */

public enum Commands {

    DRAW("draw"),
    WAIT("wait"),
    CLEAR("clear"),
    LINE("line"),
    SQUARE("square"),
    RECTANGLE("rectangle"),
    CIRCLE("circle"),
    ELIPSE("elipse"),
    ARC("arc"),
    POLYGON("polygon"),
    CANVAS("canvas"),
    TEXT("text"),
    LOG("log"),
    DEBUG("debug"),
    ERROR("error"),
    WARNING("warning");

    private String name;

    private Commands(String name) {
        this.name = name;
    }

    public String getName() {
        return this.name;
    }

    public String getCommandTemplate() {
        switch (this) {
            case CLEAR:
                return "scriptClearCanvas(g2);\n";
            case WAIT:
                return "scriptSleep(%s);\n";
            case DRAW:
            case LINE:
                return "%s\n" +
                        "scriptDrawLine(g2, %s, %s, %s, %s, %s);\n";
            case SQUARE:
            case RECTANGLE:
                return "%s\n" +
                        "scriptDrawRectangle(g2, %s, %s, %s, %s, %s);\n";
            case CIRCLE:
                return "%s\n" +
                        "scriptDrawCircle(g2, %s, %s, %s, %s);\n";
            case ELIPSE:
                return "%s\n" +
                        "scriptDrawElipse(g2, %s, %s, %s, %s, %s);\n";
            case ARC:
                return "%s\n" +
                        "scriptDrawArc(g2, %s, %s, %s, %s, %s, %s, %s);\n";
            case POLYGON:
                return "%s\n" +
                        "int xp[] = {%s};\n" +
                        "int xp[] = {%s};\n" +
                        "scriptDrawPolygon(g2, xp, yp, %s);\n";
            case CANVAS:
                return  "%s\n" +
                        "%s\n" +
                        "this.setSize(%s, %s);\n" +
                        "this.setBackground(%s);\n" +
                        "this.setForeground(%s);\n" +
                        "this.setTitle(%s);\n";
            case TEXT:
                return "%s\n" +
                        "scriptDrawString(g2, %s, %s, %s, %s);\n";
            case LOG:
                return "jgslLogger.info(%s);\n";
            case WARNING:
                return "jgslLogger.warn(%s);\n";
            case ERROR:
                return "jgslLogger.error(%s);\n";
            case DEBUG:
                return "jgslLogger.debug(%s);\n";
            default:
                return "jgslLogger.debug(\"Unknown command\")\n";
        }
    }

    public String getFormattedCommand(ArrayList<Argument> attributes, ArrayList<Argument> parameters) {
        String cmd = this.getCommandTemplate();
        String colorVarName = "null";
        String colorDecl = "";
         switch (this) {
             case CLEAR:
                 return cmd;
             case WAIT:
                 JGSLInteger duration = (JGSLInteger) parameters.get(0);
                 return String.format(cmd, duration.getJavaValue());
             case DRAW:
             case LINE:
                 JGSLInteger x1 = (JGSLInteger) parameters.get(0);
                 JGSLInteger y1 = (JGSLInteger) parameters.get(1);
                 JGSLInteger x2 = (JGSLInteger) parameters.get(2);
                 JGSLInteger y2 = (JGSLInteger) parameters.get(3);
                 if (attributes.size() == 1) {
                     JGSLColor color = (JGSLColor) attributes.get(0);
                     colorVarName = color.getName();
                     colorDecl = color.getJavaValue();
                 }
                 return String.format(cmd, colorDecl, x1.getJavaValue(), y1.getJavaValue(), x2.getJavaValue(), y2.getJavaValue
     (), colorVarName);
             case SQUARE:
                 JGSLInteger xs = (JGSLInteger) parameters.get(0);
                 JGSLInteger ys = (JGSLInteger) parameters.get(1);
                 JGSLInteger ws = (JGSLInteger) parameters.get(2);
                 if (attributes.size() == 1) {
                     JGSLColor color = (JGSLColor) attributes.get(0);
                     colorVarName = color.getName();
                     colorDecl = color.getJavaValue();
                 }
                 return String.format(cmd, colorDecl, xs.getJavaValue(), ys.getJavaValue(), ws.getJavaValue(), ws.getJavaValue
     (), colorVarName);
             case RECTANGLE:
                 JGSLInteger xr = (JGSLInteger) parameters.get(0);
                 JGSLInteger yr = (JGSLInteger) parameters.get(1);
                 JGSLInteger wr = (JGSLInteger) parameters.get(2);
                 JGSLInteger hr = (JGSLInteger) parameters.get(2);
                 if (attributes.size() == 1) {
                     JGSLColor color = (JGSLColor) attributes.get(0);
                     colorVarName = color.getName();
                     colorDecl = color.getJavaValue();
                 }
                 return String.format(cmd, colorDecl, xr.getJavaValue(), yr.getJavaValue(), wr.getJavaValue(), hr.getJavaValue
     (), colorVarName);
             case CIRCLE:
                 JGSLInteger xc = (JGSLInteger) parameters.get(0);
                 JGSLInteger yc = (JGSLInteger) parameters.get(1);
                 JGSLDouble rc = (JGSLDouble) parameters.get(2);
                 if (attributes.size() == 1) {
                     JGSLColor color = (JGSLColor) attributes.get(0);
                     colorVarName = color.getName();
                     colorDecl = color.getJavaValue();
                 }
                 return String.format(cmd, colorDecl, xc.getJavaValue(), yc.getJavaValue(), rc.getJavaValue(), colorVarName);
             case ELIPSE:
                 JGSLInteger xe = (JGSLInteger) parameters.get(0);
                 JGSLInteger ye = (JGSLInteger) parameters.get(1);
                 JGSLInteger we = (JGSLInteger) parameters.get(2);
                 JGSLInteger he = (JGSLInteger) parameters.get(2);
                 if (attributes.size() == 1) {
                     JGSLColor color = (JGSLColor) attributes.get(0);
                     colorVarName = color.getName();
                     colorDecl = color.getJavaValue();
                 }
                 return String.format(cmd, colorDecl, xe.getJavaValue(), ye.getJavaValue(), we.getJavaValue(), he.getJavaValue
     (), colorVarName);
             case ARC:
                 JGSLInteger xa = (JGSLInteger) parameters.get(0);
                 JGSLInteger ya = (JGSLInteger) parameters.get(1);
                 JGSLInteger wa = (JGSLInteger) parameters.get(2);
                 JGSLInteger ha = (JGSLInteger) parameters.get(2);
                 JGSLInteger sa = (JGSLInteger) parameters.get(2);
                 JGSLInteger aa = (JGSLInteger) parameters.get(2);
                 if (attributes.size() == 1) {
                     JGSLColor color = (JGSLColor) attributes.get(0);
                     colorVarName = color.getName();
                     colorDecl = color.getJavaValue();
                 }
                 return String.format(cmd, colorDecl, xa.getJavaValue(), ya.getJavaValue(), wa.getJavaValue(), ha.getJavaValue
     (),
                         sa.getJavaValue(), aa.getJavaValue(), colorVarName);
             case POLYGON:
                 String xp = "";
                 String yp = "";
                 for (int i = 0; i < parameters.size(); i += 2) {
                     xp += ((JGSLInteger) parameters.get(i)).getJavaValue();
                     yp += ((JGSLInteger) parameters.get(i + 1)).getJavaValue();
                     if (i < parameters.size() - 2) {
                         xp += ",";
                         yp += ",";
                     }
                 }
                 if (attributes.size() == 1) {
                     JGSLColor color = (JGSLColor) attributes.get(0);
                     colorVarName = color.getName();
                     colorDecl = color.getJavaValue();
                 }
                 return String.format(cmd, colorDecl, xp, yp, colorVarName);
             case TEXT:
                 JGSLInteger x = (JGSLInteger) parameters.get(0);
                 JGSLInteger y = (JGSLInteger) parameters.get(1);
                 JGSLString text = (JGSLString) parameters.get(2);
                 if (attributes.size() == 1) {
                     JGSLColor color = (JGSLColor) attributes.get(0);
                     colorVarName = color.getName();
```

```java
                    colorDecl = color.getJavaValue();
                }
                return String.format(cmd, colorDecl, text.getJavaValue(), x.getJavaValue(), y.getJavaValue(), colorVarName);
            case CANVAS:
                JGSLColor bg = (JGSLColor) parameters.get(0);
                JGSLColor fg = (JGSLColor) parameters.get(1);
                JGSLInteger w = (JGSLInteger) parameters.get(2);
                JGSLInteger h = (JGSLInteger) parameters.get(3);
                String title = "\"" + ResourceBundle.getBundle("jgsl.resources.JGSL").getString("app.viewer.default-
name") + "\"";
                if (parameters.size() == 5) {
                    title = ((JGSLString) parameters.get(4)).getJavaValue();
                }
                return String.format(cmd, bg.getJavaValue(), fg.getJavaValue(), w.getJavaValue(), h.getJavaValue(), bg.
getName(), fg.getName(), title);
            case LOG:
                JGSLString logMsg = (JGSLString) parameters.get(0);
                return String.format(cmd, logMsg.getJavaValue());
            case WARNING:
                JGSLString warningMsg = (JGSLString) parameters.get(0);
                return String.format(cmd, warningMsg.getJavaValue());
            case ERROR:
                JGSLString errorMsg = (JGSLString) parameters.get(0);
                return String.format(cmd, errorMsg.getJavaValue());
            case DEBUG:
                JGSLString debugMsg = (JGSLString) parameters.get(0);
                return String.format(cmd, debugMsg.getJavaValue());
            default:
                return "System.out.println(\"Unknown command\")";
        }
    }
}

```

/Users/jchavez/dev/java.net/jgsl.dev.java.net/jgsl/dev/src/jgsl/io/ImageFileFilter.java

```
    /*
 * Copyright (c) 2005 Perception Software. All Rights Reserved.
 */
package jgsl.io;


import javax.swing.filechooser.FileFilter;
import java.io.File;

/**
 * FileFilter for Image files
 *
 * @author zenarchitect
 * @version $Id: ImageFileFilter.java,v 1.1 2005/05/21 01:42:07 zenarchitect Exp $
 */
public class ImageFileFilter extends FileFilter {
    //Accept all directories and all JAR files.
            public boolean accept(File f) {
        if (f.isDirectory()) {
            return true;
        }

        String extension = ImageUtils.getExtension(f);
        if (extension != null) {
            if (extension.equalsIgnoreCase(ImageUtils.jpg) ||
                    extension.equalsIgnoreCase(ImageUtils.png) ||
                    extension.equalsIgnoreCase(ImageUtils.gif) ||
                    extension.equalsIgnoreCase(ImageUtils.bmp)) {
                return true;
            } else {
                return false;
            }
        }

        return false;
    }

    public String getDescription() {
        return "Image Files (*.jpg, *.png, *.gif, *.bmp)";
    }
}

class ImageUtils {
    public final static String jpg = "jpg";
    public final static String png = "png";
    public final static String gif = "gif";
    public final static String bmp = "bmp";

    /*
     * Get the extension of a file.
     */
            public static String getExtension(File f) {
        String ext = null;
        String s = f.getName();
        int i = s.lastIndexOf('.');

        if (i > 0 && i < s.length() - 1) {
            ext = s.substring(i + 1).toLowerCase();
        }
        return ext;
    }
}
```

/Users/jchavez/dev/java.net/jgsl.dev.java.net/jgsl/dev/src/jgsl/io/JARFileFilter.java

```
    /*
 * Copyright (c) 2005 Perception Software. All Rights Reserved.
 */
package jgsl.io;


import javax.swing.filechooser.FileFilter;
import java.io.File;

/**
 * FileFilter for .JAR files
 *
 * @author zenarchitect
 * @version $Id: JARFileFilter.java,v 1.2 2005/05/16 00:54:16 zenarchitect Exp $
 */
public class JARFileFilter extends FileFilter {
    //Accept all directories and all JAR files.
            public boolean accept(File f) {
        if (f.isDirectory()) {
            return true;
        }

        String extension = JGSLUtils.getExtension(f);
        if (extension != null) {
            if (extension.equals(JARUtils.jar)) {
                return true;
            } else {
                return false;
            }
        }

        return false;
    }

    //The description of this filter
            public String getDescription() {
        return "JAR Files (*.jar)";
    }
}
```

```

class JARUtils {
    public final static String jar = "jar";

    /*
     * Get the extension of a file.
     */
            public static String getExtension(File f) {
        String ext = null;
        String s = f.getName();
        int i = s.lastIndexOf('.');

        if (i > 0 && i < s.length() - 1) {
            ext = s.substring(i + 1).toLowerCase();
        }
        return ext;
    }
}
```

/Users/jchavez/dev/java.net/jgsl.dev.java.net/jgsl/dev/src/jgsl/io/JGSLFileFilter.java

```
    /*
 * Copyright (c) 2005 Perception Software. All Rights Reserved.
 */
package jgsl.io;


import javax.swing.filechooser.FileFilter;
import java.io.File;

/**
 * FileFilter for .jgsl files
 *
 * @author zenarchitect
 * @version $Id: JGSLFileFilter.java,v 1.5 2005/05/16 00:54:16 zenarchitect Exp $
 */
public class JGSLFileFilter extends FileFilter {
    //Accept all directories and all jgsl files.
        public boolean accept(File f) {
        if (f.isDirectory()) {
            return true;
        }

        String extension = JGSLUtils.getExtension(f);
        if (extension != null) {
            if (extension.equals(JGSLUtils.jgsl)) {
                return true;
            } else {
                return false;
            }
        }

        return false;
    }

    //The description of this filter
        public String getDescription() {
        return "JGSL Files (*.jgsl)";
    }
}
```

```java

class JGSLUtils {
    public final static String jgsl = "jgsl";

    /*
     * Get the extension of a file.
     */
            public static String getExtension(File f) {
        String ext = null;
        String s = f.getName();
        int i = s.lastIndexOf('.');

        if (i > 0 && i < s.length() - 1) {
            ext = s.substring(i + 1).toLowerCase();
        }
        return ext;
    }
}
```

/Users/jchavez/dev/java.net/jgsl.dev.java.net/jgsl/dev/src/jgsl/view/ScriptViewer.java

```java
/*
 * Copyright (c) 2005 Perception Software. All Rights Reserved.
 */
package jgsl.view;

/**
 * Interface for JGSL script viewer windows.
 *
 * @author zenarchitect
 * @version $Id: ScriptViewer.java,v 1.3 2005/05/21 01:42:12 zenarchitect Exp $
 */

public interface ScriptViewer {
    /**
     * Render the script code in fullClassName to a GUI window.
     *
     * @param fullClassName
     */
    void renderScript(String fullClassName, String saveToFileType);

}

```

/Users/jchavez/dev/java.net/jgsl.dev.java.net/jgsl/dev/src/jgsl/io/ScriptParserUtil.java

```java
/*
 * Copyright (c) 2005 Perception Software. All Rights Reserved.
 */
package jgsl.io;

/**
 * Parse a Sting into and int
 *
 * @author zenarchitect
 * @version $Id: ScriptParserUtil.java,v 1.2 2005/05/16 00:54:17 zenarchitect Exp $
 */
public class ScriptParserUtil {
    public static int parseInt(String val) throws ScriptParserException {
        int retVal = 0;
        try {
            Integer rInt = Integer.parseInt(val);
            retVal = rInt.intValue();
        }
        catch (NumberFormatException e) {
            throw new ScriptParserException("Unable to convert " + val + "to a number.");
        }
        return retVal;
    }
}
```

/Users/jchavez/dev/java.net/jgsl.dev.java.net/jgsl/dev/src/jgsl/model/Assignment.java

```java
/*
 * Copyright (c) 2005 Perception Software. All Rights Reserved.
 */

package jgsl.model;

import java.io.Serializable;

/**
 * An Assignment is a statement in which the value of one attribute is assigned to another via the "=" operator.
 *
 * @author zenarchitect
 * @version $Id: Assignment.java,v 1.2 2005/05/16 00:54:17 zenarchitect Exp $
 */
public class Assignment implements Statement, Serializable {
    /**
     * Left-hand side of assignment
     */
    private String lhs;

    /**
     * Right-hand side of the assignment
     */
    private String rhs;


    /**
     * Constructs an instance with the left-hand side and right-hand side arguments of the assignment statement
     *
     * @param lhs Left-hand side of the assignment
     * @param rhs Right-hand side of the assignment
     */
    public Assignment(String lhs, String rhs) {
        this.lhs = lhs;
        this.rhs = rhs;
    }

    /**
     * This method returns the Java language equivalent of the JGSL statement.
     *
     * @return Java language statement from the JGSL
     */
    public String getJava() {
        return null;  //To change body of implemented methods use File | Settings | File Templates.
    }

    /**
     * Set the JGSL statement body
     */
    public void setJGSL(String jgsl) {
        //To change body of implemented methods use File | Settings | File Templates.
    }

    /**
     * Return the type of statement. The String form of the class name.
     */
    public String getType() {
        return null;  //To change body of implemented methods use File | Settings | File Templates.
    }

    public String getLhs() {
        return lhs;
    }

    public void setLhs(String lhs) {
        this.lhs = lhs;
    }

    public String getRhs() {
        return rhs;
    }

    public void setRhs(String rhs) {
        this.rhs = rhs;
    }
}

```

/Users/jchavez/dev/java.net/jgsl.dev.java.net/jgsl/dev/src/jgsl/model/Command.java

```
/*
 * Copyright (c) 2005 Perception Software. All Rights Reserved.
 */

package jgsl.model;

import java.io.Serializable;
import java.util.ArrayList;

/**
 * A command statement is a JGSL command that performs a graphics operation.
 *
 * @author zenarchitect
 * @version $Id: Command.java,v 1.3 2005/05/16 00:54:17 zenarchitect Exp $
 */
public class Command implements Statement, Serializable {
    private Commands name;
    private ArrayList<Argument> attributes;
    private ArrayList<Argument> parameters;

    public Command(String name) {
        setName(name);
    }

    public Command(String name, ArrayList<Argument> parameters) {
        setName(name);
        this.parameters = parameters;
    }

    public Command(String name, ArrayList<Argument> attributes, ArrayList<Argument> parameters)
{
        setName(name);
        this.attributes = attributes;
        this.parameters = parameters;
    }

    public void setName(String name) {
        if (name.equals(Commands.LINE.getName())) {
            this.name = Commands.LINE;
        }
        if (name.equals(Commands.RECTANGLE.getName())) {
            this.name = Commands.RECTANGLE;
        }
        if (name.equals(Commands.SQUARE.getName())) {
            this.name = Commands.SQUARE;
        }
        if (name.equals(Commands.CIRCLE.getName())) {
            this.name = Commands.CIRCLE;
        }
        if (name.equals(Commands.ELIPSE.getName())) {
            this.name = Commands.ELIPSE;
        }
        if (name.equals(Commands.ARC.getName())) {
            this.name = Commands.ARC;
        }
        if (name.equals(Commands.POLYGON.getName())) {
            this.name = Commands.POLYGON;
        }
        if (name.equals(Commands.DRAW.getName())) {
            this.name = Commands.DRAW;
        } else if (name.equals(Commands.CLEAR.getName())) {
            this.name = Commands.CLEAR;
        } else if (name.equals(Commands.WAIT.getName())) {
            this.name = Commands.WAIT;
        } else if (name.equals(Commands.CANVAS.getName())) {
            this.name = Commands.CANVAS;
        } else if (name.equals(Commands.TEXT.getName())) {
            this.name = Commands.TEXT;
        } else if (name.equals(Commands.LOG.getName())) {
            this.name = Commands.LOG;
        } else if (name.equals(Commands.WARNING.getName())) {
            this.name = Commands.WARNING;
        } else if (name.equals(Commands.ERROR.getName())) {
            this.name = Commands.ERROR;
        } else if (name.equals(Commands.DEBUG.getName())) {
            this.name = Commands.DEBUG;
        }
    }

    /**
     * This method returns the Java language equivalent of the JGSL statement.
     *
     * @return Java language statement from the JGSL
     */
    public String getJava() {
        assert(name != null);
        return name.getFormattedCommand(attributes, parameters);
    }

    /**
     * Set the JGSL statement body
     */
    public void setJGSL(String jgsl) {
        //TODO: To change body of implemented methods use File | Settings | File Templates.
    }

    /**
     * Return the type of statement. The String form of the class name.
     */
    public String getType() {
        assert(name != null);
        return name.getName();
    }
}
```

/Users/jchavez/dev/java.net/jgsl.dev.java.net/jgsl/dev/src/jgsl/model/Declaration.java

```java
/*
 * Copyright (c) 2005 Perception Software. All Rights Reserved.
 */

package jgsl.model;

import java.io.Serializable;


/**
 * A Declaration statement is one that contains the declaration of a script variable.
 *
 * @author zenarchitect
 * @version $Id: Declaration.java,v 1.2 2005/05/16 00:54:18 zenarchitect Exp $
 */
public class Declaration implements Statement, Serializable {
    private String type;
    private String identifier;
    private String value;

    /**
     * Create a declaration with a given type, identifier and initial value
     *
     * @param type       type of the declaration
     * @param identifier script identifier
     * @param value      initial value
     */
    public Declaration(String type, String identifier, String value) {
        this.type = type;
        this.identifier = identifier;
        this.value = value;
    }

    /**
     * This method returns the Java language equivalent of the JGSL statement.
     *
     * @return Java language statement from the JGSL
     */
    public String getJava() {
        return null;  //To change body of implemented methods use File | Settings | File Templates.
    }

    /**
     * Set the JGSL statement body
     */
    public void setJGSL(String jgsl) {
        //To change body of implemented methods use File | Settings | File Templates.
    }

    /**
     * Return the type of statement. The String form of the class name.
     */
    public String getType() {
        return null;  //To change body of implemented methods use File | Settings | File Templates.
    }
}

```

/Users/jchavez/dev/java.net/jgsl.dev.java.net/jgsl/dev/src/jgsl/model/JGSLColor.java

```java
/*
 * Copyright (c) 2005 Perception Software. All Rights Reserved.
 */
package jgsl.model;

import java.awt.Color;

/**
 * Declare an instance of a color type.
 *
 * @author zenarchitect
 * @version $Id: JGSLColor.java,v 1.3 2005/05/21 01:42:07 zenarchitect Exp $
 */
public class JGSLColor implements Type, Value, Argument {
    private String name;
    private Color color = Color.BLACK;

    public JGSLColor(String name, Color color) {
        this.name = name + System.currentTimeMillis();
        this.color = color;
    }

    public Color getColor() {
        return color;
    }

    /**
     * Get the java Class meta-data for this type
     *
     * @return The Class mete-data for this type
     */
    public Class getJavaClass() {
        return color.getClass();
    }

    /**
     * Get the Java type as a String
     *
     * @return a String containing the type
     */
    public String getJavaType() {
        return color.getClass().getName();
    }

    /**
     * Get the Java representation of this value
     *
     * @return A String containing the Java representation of this value
     */
    public String getJavaValue() {
        String javaValue = String.format("java.awt.Color %s = new java.awt.Color(%d, %d, %d, %d);\n", name, color.getRed(), color.getGreen(), color.getBlue(), color.getAlpha());
        return javaValue;
    }

    /**
     * Get the name of the argument
     *
     * @return String containing the name
     */
    public String getName() {
        return name;
    }

}

```

/Users/jchavez/dev/java.net/jgsl.dev.java.net/jgsl/dev/src/jgsl/model/JGSLDouble.java

```
/*
 * Copyright (c) 2005 Perception Software. All Rights Reserved.
 */
package jgsl.model;

// TODO - write javadocs
/**
 * @author zenarchitect
 * @version $Id: JGSLDouble.java,v 1.2 2005/05/16 00:54:18 zenarchitect Exp $
 */
public class JGSLDouble implements Type, Value, Argument {
    private String name;
    private Double value;

    public JGSLDouble(String name, Double value) {
        this.name = name;
        this.value = value;
    }

    public JGSLDouble(String name, String value) {
        this.name = name;
        this.value = Double.valueOf(value);
    }

    public Double getValue() {
        return value;
    }

    /**
     * Get the java Class meta-data for this type
     *
     * @return The Class mete-data for this type
     */
    public Class getJavaClass() {
        return value.getClass();
    }

    /**
     * Get the Java type as a String
```

```java
     *
     * @return a String containing the type
     */
    public String getJavaType() {
        return value.getClass().getName();
    }

    /**
     * Get the Java representation of this value
     *
     * @return A String containing the Java representation of this value
     */
    public String getJavaValue() {
        return value.toString();
    }

    /**
     * Get the name of the argument
     *
     * @return String containing the name
     */
    public String getName() {
        return name;
    }
}

```

/Users/jchavez/dev/java.net/jgsl.dev.java.net/jgsl/dev/src/jgsl/model/JGSLInteger.java

```
/*
 * Copyright (c) 2005 Perception Software. All Rights Reserved.
 */
package jgsl.model;

// TODO - write javadocs
/**
 * @author zenarchitect
 * @version $Id: JGSLInteger.java,v 1.2 2005/05/16 00:54:18 zenarchitect Exp $
 */
public class JGSLInteger implements Type, Value, Argument {
    private String name;
    private Integer value;

    public JGSLInteger(String name, Integer value) {
        this.name = name;
        this.value = value;
    }

    public JGSLInteger(String name, String value) {
        this.name = name;
        this.value = Integer.valueOf(value);
    }

    public Integer getValue() {
        return value;
    }

    /**
     * Get the java Class meta-data for this type
     *
     * @return The Class mete-data for this type
     */
    public Class getJavaClass() {
        return value.getClass();
    }

    /**
     * Get the Java type as a String
     *
     * @return a String containing the type
     */
    public String getJavaType() {
        return value.getClass().getName();
    }

    /**
     * Get the Java representation of this value
     *
     * @return A String containing the Java representation of this value
     */
    public String getJavaValue() {
        return value.toString();
    }

    /**
     * Get the name of the argument
     *
     * @return String containing the name
     */
    public String getName() {
        return name;
    }
}

```

/Users/jchavez/dev/java.net/jgsl.dev.java.net/jgsl/dev/src/jgsl/model/JGSLString.java

```java
    /*
 * Copyright (c) 2005 Perception Software. All Rights Reserved.
 */
package jgsl.model;

// TODO - write javadocs
/**
 * @author zenarchitect
 * @version $Id: JGSLString.java,v 1.2 2005/05/16 00:54:18 zenarchitect Exp $
 */
public class JGSLString implements Type, Value, Argument {
    private String name;
    private String value;

    public JGSLString(String name, String value) {
        this.name = name;
        this.value = value;
    }

    public String getValue() {
        return value;
    }

    /**
     * Get the java Class meta-data for this type
     *
     * @return The Class mete-data for this type
     */
    public Class getJavaClass() {
        return value.getClass();
    }

    /**
     * Get the Java type as a String
     *
     * @return a String containing the type
     */
    public String getJavaType() {
        return value.getClass().getName();
    }

    /**
     * Get the Java representation of this value
     *
     * @return A String containing the Java representation of this value
     */
    public String getJavaValue() {
        return value;
    }

    /**
     * Get the name of the argument
     *
     * @return String containing the name
     */
    public String getName() {
        return name;
    }
}

```

/Users/jchavez/dev/java.net/jgsl.dev.java.net/jgsl/dev/src/jgsl/parser/JGSL_ParserConstants.java

```java
/* Generated By:JavaCC: Do not edit this line. JGSL_ParserConstants.java */
package jgsl.parser;

public interface JGSL_ParserConstants {
    int EOF = 0;
    int SINGLE_LINE_COMMENT = 10;
    int FORMAL_COMMENT = 11;
    int MULTI_LINE_COMMENT = 12;
    int AND = 14;
    int OR = 15;
    int NOT = 16;
    int IF = 17;
    int THEN = 18;
    int ELSE = 19;
    int ELSEIF = 20;
    int TRUE = 21;
    int FALSE = 22;
    int CLEAR = 23;
    int CANVAS = 24;
    int DRAW = 25;
    int TEXT = 26;
    int RECTANGLE = 27;
    int SQUARE = 28;
    int CIRCLE = 29;
    int ELIPSE = 30;
    int ARC = 31;
    int POLYGON = 32;
    int LINE = 33;
    int WAIT = 34;
    int LOG = 35;
    int DEBUG = 36;
    int ERROR = 37;
    int WARNING = 38;
    int WRITE = 39;
    int READ = 40;
    int JGSL = 41;
    int VERSION = 42;
    int BEGIN = 43;
    int END = 44;
    int REPEAT = 45;
    int LOOP = 46;
    int BLACK = 47;
    int BLUE = 48;
    int DARK_GRAY = 49;
    int GRAY = 50;
    int GREEN = 51;
    int LIGHT_GRAY = 52;
    int MAGENTA = 53;
    int ORANGE = 54;
    int PINK = 55;
    int RED = 56;
    int WHITE = 57;
    int YELLOW = 58;
    int GRADIENT = 59;
    int FILL = 60;
    int BORDER = 61;
    int FOREGROUND = 62;
    int BACKGROUND = 63;
    int COLOR = 64;
    int DECLARE = 65;
    int DOC = 66;
    int INTEGER_LITERAL = 67;
    int DECIMAL_LITERAL = 68;
    int HEX_LITERAL = 69;
    int OCTAL_LITERAL = 70;
    int FLOATING_POINT_LITERAL = 71;
    int STRING_LITERAL = 72;
    int IDENTIFIER = 73;
    int LETTER = 74;
    int DIGIT = 75;
    int LPAREN = 76;
    int RPAREN = 77;
    int SEMICOLON = 78;
    int COLON = 79;
    int COMMA = 80;
    int ASSIGN = 81;
    int GT = 82;
    int LT = 83;
    int BANG = 84;
    int EQ = 85;
    int LE = 86;
```

```java
    int GE = 87;
    int NE = 88;
    int SC_OR = 89;
    int SC_AND = 90;
    int PLUS = 91;
    int MINUS = 92;
    int STAR = 93;
    int SLASH = 94;
    int MOD = 95;

    int DEFAULT = 0;
    int IN_SINGLE_LINE_COMMENT = 1;
    int IN_FORMAL_COMMENT = 2;
    int IN_MULTI_LINE_COMMENT = 3;

    String[] tokenImage = {
        "<EOF>",
        "\" \"",
        "\"\\t\"",
        "\"\\n\"",
        "\"\\r\"",
        "\"\\f\"",
        "\"//\"",
        "\"#\"",
        "<token of kind 8>",
        "\"/*\"",
        "<SINGLE_LINE_COMMENT>",
        "\"*/\"",
        "\"*/\"",
        "<token of kind 13>",
        "\"and\"",
        "\"or\"",
        "\"not\"",
        "\"if\"",
        "\"then\"",
        "\"else\"",
        "\"elseif\"",
        "\"true\"",
        "\"false\"",
        "\"clear\"",
        "\"canvas\"",
        "\"draw\"",
        "\"text\"",
        "\"rectangle\"",
        "\"square\"",
        "\"circle\"",
        "\"elipse\"",
        "\"arc\"",
        "\"polygon\"",
        "\"line\"",
        "\"wait\"",
        "\"log\"",
        "\"debug\"",
        "\"error\"",
        "\"warning\"",
        "\"write\"",
        "\"read\"",
        "\"jgsl\"",
        "\"version\"",
        "\"begin\"",
        "\"end\"",
        "\"repeat\"",
        "\"loop\"",
        "\"black\"",
        "\"blue\"",
        "\"dark_gray\"",
        "\"gray\"",
        "\"green\"",
        "\"light_gray\"",
        "\"magenta\"",
        "\"orange\"",
        "\"pink\"",
        "\"red\"",
        "\"white\"",
        "\"yellow\"",
        "\"gradient\"",
        "\"fill\"",
        "\"border\"",
        "\"foreground\"",
        "\"background\"",
        "\"color\"",
        "\"declare\"",
        "\"doc\"",
        "<INTEGER_LITERAL>",
        "<DECIMAL_LITERAL>",
        "<HEX_LITERAL>",
            "<OCTAL_LITERAL>",
            "<FLOATING_POINT_LITERAL>",
            "<STRING_LITERAL>",
            "<IDENTIFIER>",
            "<LETTER>",
            "<DIGIT>",
            "\"(\"",
            "\")\"",
            "\";\"",
            "\":\"",
            "\",\"",
            "\"=\"",
            "\">\"",
            "\"<\"",
            "\"!\"",
            "\"==\"",
            "\"<=\"",
            "\">=\"",
            "\"!=\"",
            "\"||\"",
            "\"&&\"",
            "\"+\"",
            "\"-\"",
            "\"*\"",
            "\"/\"",
            "\"%\"",
        };

}

```

/Users/jchavez/dev/java.net/jgsl.dev.java.net/jgsl/dev/src/jgsl/parser/Token.java

```
    /* Generated By:JavaCC: Do not edit this line. Token.java Version 3.0 */
package jgsl.parser;

/**
 * Describes the input token stream.
 */

public class Token {
    /**
     * An integer that describes the kind of this token.  This numbering system is determined by JavaCCParser, and a
table
     * of these numbers is stored in the file ...Constants.java.
     */
    public int kind;

    /**
     * beginLine and beginColumn describe the position of the first character of this token; endLine and endColumn
     * describe the position of the last character of this token.
     */
    public int beginLine
    ,
    beginColumn
    ,
    endLine
    ,
    endColumn;

    /**
     * The string image of the token.
     */
    public String image;

    /**
     * A reference to the next regular (non-special) token from the input stream.  If this is the last token from the
     * input stream, or if the token manager has not read tokens beyond this one, this field is set to null.  This is true
     * only if this token is also a regular token.  Otherwise, see below for a description of the contents of this field.
     */
    public Token next;

    /**
     * This field is used to access special tokens that occur prior to this token, but after the immediately preceding
     * regular (non-special) token. If there are no such special tokens, this field is set to null. When there are more
     * than one such special token, this field refers to the last of these special tokens, which in turn refers to the
     * next previous special token through its specialToken field, and so on until the first special token (whose
     * specialToken field is null). The next fields of special tokens refer to other special tokens that immediately
     * follow it (without an intervening regular token).  If there is no such token, this field is null.
     */
    public Token specialToken;
```

```java

    /**
     * Returns the image.
     */
    public String toString() {
        return image;
    }

    /**
     * Returns a new Token object, by default. However, if you want, you can create and return subclass objects based on
     * the value of ofKind. Simply add the cases to the switch for all those special cases. For example, if you have a
     * subclass of Token called IDToken that you want to create if ofKind is ID, simlpy add something like :
     * <p/>
     * case MyParserConstants.ID : return new IDToken();
     * <p/>
     * to the following switch statement. Then you can cast matchedToken variable to the appropriate type and use it in
     * your lexical actions.
     */
    public static final Token newToken(int ofKind) {
        switch (ofKind) {
            default :
                return new Token();
        }
    }

}

```

/Users/jchavez/dev/java.net/jgsl.dev.java.net/jgsl/dev/src/jgsl/parser/JGSL_ParserTokenManager.java

```java
    /* Generated By:JavaCC: Do not edit this line. JGSL_ParserTokenManager.java */
package jgsl.parser;
public class JGSL_ParserTokenManager implements JGSL_ParserConstants
{
  public  java.io.PrintStream debugStream = System.out;
  public  void setDebugStream(java.io.PrintStream ds) { debugStream = ds; }
private final int jjStopStringLiteralDfa_0(int pos, long active0, long active1)
{
   switch (pos)
   {
      case 0:
         if ((active0 & 0x240L) != 0L || (active1 & 0x40000000L) != 0L)
            return 2;
         if ((active0 & 0xffffffffffffc000L) != 0L || (active1 & 0x7L) != 0L)
         {
            jjmatchedKind = 73;
            return 19;
         }
         return -1;
      case 1:
         if ((active0 & 0x200L) != 0L)
            return 0;
         if ((active0 & 0xffbfffffffffd4000L) != 0L || (active1 & 0x7L) != 0L)
         {
            if (jjmatchedPos != 1)
            {
               jjmatchedKind = 73;
               jjmatchedPos = 1;
            }
            return 19;
         }
         if ((active0 & 0x40000000028000L) != 0L)
            return 19;
         return -1;
      case 2:
         if ((active0 & 0xfeffeff77ffc0000L) != 0L || (active1 & 0x3L) != 0L)
         {
            jjmatchedKind = 73;
            jjmatchedPos = 2;
            return 19;
         }
         if ((active0 & 0x1001008800014000L) != 0L || (active1 & 0x4L) != 0L)
            return 19;
         return -1;
      case 3:
         if ((active0 & 0xee7aacf179c00000L) != 0L || (active1 & 0x3L) != 0L)
         {
            if (jjmatchedPos != 3)
            {
               jjmatchedKind = 73;
               jjmatchedPos = 3;
            }
            return 19;
         }
         if ((active0 & 0x10854306063c0000L) != 0L)
            return 19;
         return -1;
      case 4:
         if ((active0 & 0xec72244179100000L) != 0L || (active1 & 0x2L) != 0L)
         {
            jjmatchedKind = 73;
            jjmatchedPos = 4;
            return 19;
         }
         if ((active0 & 0x20888b000c00000L) != 0L || (active1 & 0x1L) != 0L)
            return 19;
         return -1;
      case 5:
         if ((active0 & 0xc832044108000000L) != 0L || (active1 & 0x2L) != 0L)
         {
            jjmatchedKind = 73;
            jjmatchedPos = 5;
            return 19;
         }
         if ((active0 & 0x2440200071100000L) != 0L)
            return 19;
         return -1;
      case 6:
         if ((active0 & 0xc812000008000000L) != 0L)
         {
            jjmatchedKind = 73;
            jjmatchedPos = 6;
            return 19;
         }
         if ((active0 & 0x20044100000000L) != 0L || (active1 & 0x2L) != 0L)
            return 19;
         return -1;
      case 7:
         if ((active0 & 0xc012000008000000L) != 0L)
         {
            jjmatchedKind = 73;
            jjmatchedPos = 7;
            return 19;
         }
         if ((active0 & 0x800000000000000L) != 0L)
            return 19;
         return -1;
      case 8:
         if ((active0 & 0xc010000000000000L) != 0L)
         {
            jjmatchedKind = 73;
            jjmatchedPos = 8;
            return 19;
         }
         if ((active0 & 0x2000008000000L) != 0L)
            return 19;
         return -1;
      default :
         return -1;
   }
}
private final int jjStartNfa_0(int pos, long active0, long active1)
{
   return jjMoveNfa_0(jjStopStringLiteralDfa_0(pos, active0, active1), pos + 1);
}
private final int jjStopAtPos(int pos, int kind)
{
   jjmatchedKind = kind;
   jjmatchedPos = pos;
   return pos + 1;
}
private final int jjStartNfaWithStates_0(int pos, int kind, int state)
{
   jjmatchedKind = kind;
   jjmatchedPos = pos;
   try { curChar = input_stream.readChar(); }
   catch(java.io.IOException e) { return pos + 1; }
   return jjMoveNfa_0(state, pos + 1);
}
private final int jjMoveStringLiteralDfa0_0()
{
   switch(curChar)
   {
      case 9:
         return jjStopAtPos(0, 2);
      case 10:
         return jjStopAtPos(0, 3);
      case 12:
         return jjStopAtPos(0, 5);
      case 13:
         return jjStopAtPos(0, 4);
      case 32:
         return jjStopAtPos(0, 1);
      case 33:
         jjmatchedKind = 84;
         return jjMoveStringLiteralDfa1_0(0x0L, 0x1000000L);
      case 35:
         return jjStopAtPos(0, 7);
      case 37:
```

```
      return jjStopAtPos(0, 95);
    case 38:
      return jjMoveStringLiteralDfa1_0(0x0L, 0x4000000L);
    case 40:
      return jjStopAtPos(0, 76);
    case 41:
      return jjStopAtPos(0, 77);
    case 42:
      return jjStopAtPos(0, 93);
    case 43:
      return jjStopAtPos(0, 91);
    case 44:
      return jjStopAtPos(0, 80);
    case 45:
      return jjStopAtPos(0, 92);
    case 47:
      jjmatchedKind = 94;
      return jjMoveStringLiteralDfa1_0(0x240L, 0x0L);
    case 58:
      return jjStopAtPos(0, 79);
    case 59:
      return jjStopAtPos(0, 78);
    case 60:
      jjmatchedKind = 83;
      return jjMoveStringLiteralDfa1_0(0x0L, 0x400000L);
    case 61:
      jjmatchedKind = 81;
      return jjMoveStringLiteralDfa1_0(0x0L, 0x200000L);
    case 62:
      jjmatchedKind = 82;
      return jjMoveStringLiteralDfa1_0(0x0L, 0x800000L);
    case 65:
    case 97:
      return jjMoveStringLiteralDfa1_0(0x80004000L, 0x0L);
    case 66:
    case 98:
      return jjMoveStringLiteralDfa1_0(0xa001880000000000L, 0x0L);
    case 67:
    case 99:
      return jjMoveStringLiteralDfa1_0(0x21800000L, 0x1L);
    case 68:
    case 100:
      return jjMoveStringLiteralDfa1_0(0x2001002000000L, 0x6L);
    case 69:
    case 101:
      return jjMoveStringLiteralDfa1_0(0x102040180000L, 0x0L);
    case 70:
    case 102:
      return jjMoveStringLiteralDfa1_0(0x5000000000400000L, 0x0L);
    case 71:
    case 103:
            return jjMoveStringLiteralDfa1_0(0x80c000000000000L, 0x0L);
        case 73:
        case 105:
            return jjMoveStringLiteralDfa1_0(0x20000L, 0x0L);
        case 74:
        case 106:
            return jjMoveStringLiteralDfa1_0(0x20000000000L, 0x0L);
        case 76:
        case 108:
            return jjMoveStringLiteralDfa1_0(0x10400a00000000L, 0x0L);
        case 77:
        case 109:
            return jjMoveStringLiteralDfa1_0(0x20000000000000L, 0x0L);
        case 78:
        case 110:
            return jjMoveStringLiteralDfa1_0(0x10000L, 0x0L);
        case 79:
        case 111:
            return jjMoveStringLiteralDfa1_0(0x40000000008000L, 0x0L);
        case 80:
        case 112:
            return jjMoveStringLiteralDfa1_0(0x80000100000000L, 0x0L);
        case 82:
        case 114:
            return jjMoveStringLiteralDfa1_0(0x100210008000000L, 0x0L);
        case 83:
        case 115:
            return jjMoveStringLiteralDfa1_0(0x10000000L, 0x0L);
        case 84:
        case 116:
            return jjMoveStringLiteralDfa1_0(0x4240000L, 0x0L);
        case 86:
        case 118:
            return jjMoveStringLiteralDfa1_0(0x40000000000L, 0x0L);
        case 87:
        case 119:
            return jjMoveStringLiteralDfa1_0(0x20000c400000000L, 0x0L);
        case 89:
        case 121:
            return jjMoveStringLiteralDfa1_0(0x400000000000000L, 0x0L);
        case 124:
            return jjMoveStringLiteralDfa1_0(0x0L, 0x2000000L);
        default :
            return jjMoveNfa_0(3, 0);
    }
}
private final int jjMoveStringLiteralDfa1_0(long active0, long active1)
{
    try { curChar = input_stream.readChar(); }
    catch(java.io.IOException e) {
        jjStopStringLiteralDfa_0(0, active0, active1);
```

```
      return 1;
   }
   switch(curChar)
   {
      case 38:
         if ((active1 & 0x4000000L) != 0L)
            return jjStopAtPos(1, 90);
         break;
      case 42:
         if ((active0 & 0x200L) != 0L)
            return jjStartNfaWithStates_0(1, 9, 0);
         break;
      case 47:
         if ((active0 & 0x40L) != 0L)
            return jjStopAtPos(1, 6);
         break;
      case 61:
         if ((active1 & 0x200000L) != 0L)
            return jjStopAtPos(1, 85);
         else if ((active1 & 0x400000L) != 0L)
            return jjStopAtPos(1, 86);
         else if ((active1 & 0x800000L) != 0L)
            return jjStopAtPos(1, 87);
         else if ((active1 & 0x1000000L) != 0L)
            return jjStopAtPos(1, 88);
         break;
      case 65:
      case 97:
         return jjMoveStringLiteralDfa2_0(active0, 0x8022004401400000L, active1, 0L);
      case 69:
      case 101:
         return jjMoveStringLiteralDfa2_0(active0, 0x5002d100c000000L, active1, 0x2L);
      case 70:
      case 102:
         if ((active0 & 0x20000L) != 0L)
            return jjStartNfaWithStates_0(1, 17, 19);
         break;
      case 71:
      case 103:
         return jjMoveStringLiteralDfa2_0(active0, 0x20000000000L, active1, 0L);
      case 72:
      case 104:
         return jjMoveStringLiteralDfa2_0(active0, 0x200000000040000L, active1, 0L);
      case 73:
      case 105:
         return jjMoveStringLiteralDfa2_0(active0, 0x1090000220000000L, active1, 0L);
      case 76:
      case 108:
         return jjMoveStringLiteralDfa2_0(active0, 0x1800040980000L, active1, 0L);
      case 78:
      case 110:
         return jjMoveStringLiteralDfa2_0(active0, 0x100000004000L, active1, 0L);
      case 79:
      case 111:
         return jjMoveStringLiteralDfa2_0(active0, 0x6000400900010000L, active1, 0x5L);
      case 81:
      case 113:
         return jjMoveStringLiteralDfa2_0(active0, 0x10000000L, active1, 0L);
      case 82:
      case 114:
         if ((active0 & 0x8000L) != 0L)
         {
            jjmatchedKind = 15;
            jjmatchedPos = 1;
         }
         return jjMoveStringLiteralDfa2_0(active0, 0x84c00a082200000L, active1, 0L);
      case 124:
         if ((active1 & 0x2000000L) != 0L)
            return jjStopAtPos(1, 89);
         break;
      default :
         break;
   }
   return jjStartNfa_0(0, active0, active1);
}
private final int jjMoveStringLiteralDfa2_0(long old0, long active0, long old1, long active1)
{
   if (((active0 &= old0) | (active1 &= old1)) == 0L)
      return jjStartNfa_0(0, old0, old1);
   try { curChar = input_stream.readChar(); }
   catch(java.io.IOException e) {
      jjStopStringLiteralDfa_0(1, active0, active1);
      return 2;
   }
   switch(curChar)
   {
      case 65:
      case 97:
         return jjMoveStringLiteralDfa3_0(active0, 0x844810002000000L, active1, 0L);
      case 66:
      case 98:
         return jjMoveStringLiteralDfa3_0(active0, 0x1000000000L, active1, 0L);
      case 67:
      case 99:
         if ((active0 & 0x80000000L) != 0L)
            return jjStartNfaWithStates_0(2, 31, 19);
         else if ((active1 & 0x4L) != 0L)
            return jjStartNfaWithStates_0(2, 66, 19);
         return jjMoveStringLiteralDfa3_0(active0, 0x8000000008000000L, active1, 0x2L);
      case 68:
      case 100:
         if ((active0 & 0x4000L) != 0L)
```

```
            return jjStartNfaWithStates_0(2, 14, 19);
         else if ((active0 & 0x100000000000L) != 0L)
            return jjStartNfaWithStates_0(2, 44, 19);
         else if ((active0 & 0x100000000000000L) != 0L)
            return jjStartNfaWithStates_0(2, 56, 19);
         break;
      case 69:
      case 101:
         return jjMoveStringLiteralDfa3_0(active0, 0x8000000840000L, active1, 0L);
      case 71:
      case 103:
         if ((active0 & 0x800000000L) != 0L)
            return jjStartNfaWithStates_0(2, 35, 19);
         return jjMoveStringLiteralDfa3_0(active0, 0x30080000000000L, active1, 0L);
      case 73:
      case 105:
         return jjMoveStringLiteralDfa3_0(active0, 0x200008440000000L, active1, 0L);
      case 76:
      case 108:
         return jjMoveStringLiteralDfa3_0(active0, 0x1400000100400000L, active1, 0x1L);
      case 78:
      case 110:
         return jjMoveStringLiteralDfa3_0(active0, 0x80000201000000L, active1, 0L);
      case 79:
      case 111:
         return jjMoveStringLiteralDfa3_0(active0, 0x400000000000L, active1, 0L);
      case 80:
      case 112:
         return jjMoveStringLiteralDfa3_0(active0, 0x200000000000L, active1, 0L);
      case 82:
      case 114:
         return jjMoveStringLiteralDfa3_0(active0, 0x6002046020000000L, active1, 0L);
      case 83:
      case 115:
         return jjMoveStringLiteralDfa3_0(active0, 0x20000180000L, active1, 0L);
      case 84:
      case 116:
         if ((active0 & 0x10000L) != 0L)
            return jjStartNfaWithStates_0(2, 16, 19);
         break;
      case 85:
      case 117:
         return jjMoveStringLiteralDfa3_0(active0, 0x1000010200000L, active1, 0L);
      case 88:
      case 120:
         return jjMoveStringLiteralDfa3_0(active0, 0x4000000L, active1, 0L);
      default :
         break;
   }
   return jjStartNfa_0(1, active0, active1);
}
```

```
private final int jjMoveStringLiteralDfa3_0(long old0, long active0, long old1, long active1)
{
   if (((active0 &= old0) | (active1 &= old1)) == 0L)
      return jjStartNfa_0(1, old0, old1);
   try { curChar = input_stream.readChar(); }
   catch(java.io.IOException e) {
      jjStopStringLiteralDfa_0(2, active0, active1);
      return 3;
   }
   switch(curChar)
   {
      case 65:
      case 97:
         return jjMoveStringLiteralDfa4_0(active0, 0x10800000L, active1, 0L);
      case 67:
      case 99:
         return jjMoveStringLiteralDfa4_0(active0, 0x800020000000L, active1, 0L);
      case 68:
      case 100:
         if ((active0 & 0x10000000000L) != 0L)
            return jjStartNfaWithStates_0(3, 40, 19);
         return jjMoveStringLiteralDfa4_0(active0, 0x2800000000000000L, active1, 0L);
      case 69:
      case 101:
         if ((active0 & 0x80000L) != 0L)
         {
            jjmatchedKind = 19;
            jjmatchedPos = 3;
         }
         else if ((active0 & 0x200000L) != 0L)
            return jjStartNfaWithStates_0(3, 21, 19);
         else if ((active0 & 0x200000000L) != 0L)
            return jjStartNfaWithStates_0(3, 33, 19);
         else if ((active0 & 0x1000000000000L) != 0L)
            return jjStartNfaWithStates_0(3, 48, 19);
         return jjMoveStringLiteralDfa4_0(active0, 0x4028200000100000L, active1, 0L);
      case 72:
      case 104:
         return jjMoveStringLiteralDfa4_0(active0, 0x10000000000000L, active1, 0L);
      case 73:
      case 105:
         return jjMoveStringLiteralDfa4_0(active0, 0x80000000000L, active1, 0L);
      case 75:
      case 107:
         if ((active0 & 0x80000000000000L) != 0L)
            return jjStartNfaWithStates_0(3, 55, 19);
         return jjMoveStringLiteralDfa4_0(active0, 0x8002000000000000L, active1, 0L);
      case 76:
      case 108:
         if ((active0 & 0x20000000000L) != 0L)
            return jjStartNfaWithStates_0(3, 41, 19);
         else if ((active0 & 0x1000000000000000L) != 0L)
            return jjStartNfaWithStates_0(3, 60, 19);
         return jjMoveStringLiteralDfa4_0(active0, 0x400000000000000L, active1, 0x2L);
      case 78:
      case 110:
         if ((active0 & 0x40000L) != 0L)
            return jjStartNfaWithStates_0(3, 18, 19);
         return jjMoveStringLiteralDfa4_0(active0, 0x40004000000000L, active1, 0L);
      case 79:
      case 111:
         return jjMoveStringLiteralDfa4_0(active0, 0x2000000000L, active1, 0x1L);
      case 80:
      case 112:
         if ((active0 & 0x400000000000L) != 0L)
            return jjStartNfaWithStates_0(3, 46, 19);
         return jjMoveStringLiteralDfa4_0(active0, 0x40000000L, active1, 0L);
      case 83:
      case 115:
         return jjMoveStringLiteralDfa4_0(active0, 0x40000400000L, active1, 0L);
      case 84:
      case 116:
         if ((active0 & 0x4000000L) != 0L)
            return jjStartNfaWithStates_0(3, 26, 19);
         else if ((active0 & 0x400000000L) != 0L)
            return jjStartNfaWithStates_0(3, 34, 19);
         return jjMoveStringLiteralDfa4_0(active0, 0x200008008000000L, active1, 0L);
      case 85:
      case 117:
         return jjMoveStringLiteralDfa4_0(active0, 0x1000000000L, active1, 0L);
      case 86:
      case 118:
         return jjMoveStringLiteralDfa4_0(active0, 0x1000000L, active1, 0L);
      case 87:
      case 119:
         if ((active0 & 0x2000000L) != 0L)
            return jjStartNfaWithStates_0(3, 25, 19);
         break;
      case 89:
      case 121:
         if ((active0 & 0x4000000000000L) != 0L)
            return jjStartNfaWithStates_0(3, 50, 19);
         return jjMoveStringLiteralDfa4_0(active0, 0x100000000L, active1, 0L);
      default :
         break;
   }
   return jjStartNfa_0(2, active0, active1);
 }
 private final int jjMoveStringLiteralDfa4_0(long old0, long active0, long old1, long active1)
 {
   if (((active0 &= old0) | (active1 &= old1)) == 0L)
      return jjStartNfa_0(2, old0, old1);
```

```
try { curChar = input_stream.readChar(); }
catch(java.io.IOException e) {
  jjStopStringLiteralDfa_0(3, active0, active1);
  return 4;
}
switch(curChar)
{
  case 95:
    return jjMoveStringLiteralDfa5_0(active0, 0x2000000000000L, active1, 0L);
  case 65:
  case 97:
    return jjMoveStringLiteralDfa5_0(active0, 0x200009000000L, active1, 0x2L);
  case 69:
  case 101:
    if ((active0 & 0x400000L) != 0L)
      return jjStartNfaWithStates_0(4, 22, 19);
    else if ((active0 & 0x8000000000L) != 0L)
      return jjStartNfaWithStates_0(4, 39, 19);
    else if ((active0 & 0x200000000000000L) != 0L)
      return jjStartNfaWithStates_0(4, 57, 19);
    return jjMoveStringLiteralDfa5_0(active0, 0x2000000000000000L, active1, 0L);
  case 71:
  case 103:
    if ((active0 & 0x1000000000L) != 0L)
      return jjStartNfaWithStates_0(4, 36, 19);
    return jjMoveStringLiteralDfa5_0(active0, 0xc040000100000000L, active1, 0L);
  case 73:
  case 105:
    return jjMoveStringLiteralDfa5_0(active0, 0x800044000100000L, active1, 0L);
  case 75:
  case 107:
    if ((active0 & 0x800000000000L) != 0L)
      return jjStartNfaWithStates_0(4, 47, 19);
    break;
  case 76:
  case 108:
    return jjMoveStringLiteralDfa5_0(active0, 0x20000000L, active1, 0L);
  case 78:
  case 110:
    if ((active0 & 0x80000000000L) != 0L)
      return jjStartNfaWithStates_0(4, 43, 19);
    else if ((active0 & 0x8000000000000L) != 0L)
      return jjStartNfaWithStates_0(4, 51, 19);
    return jjMoveStringLiteralDfa5_0(active0, 0x20000000000000L, active1, 0L);
  case 79:
  case 111:
    return jjMoveStringLiteralDfa5_0(active0, 0x400000000000000L, active1, 0L);
  case 82:
  case 114:
    if ((active0 & 0x800000L) != 0L)
      return jjStartNfaWithStates_0(4, 23, 19);
      else if ((active0 & 0x2000000000L) != 0L)
         return jjStartNfaWithStates_0(4, 37, 19);
      else if ((active1 & 0x1L) != 0L)
         return jjStartNfaWithStates_0(4, 64, 19);
      return jjMoveStringLiteralDfa5_0(active0, 0x10000000L, active1, 0L);
   case 83:
   case 115:
      return jjMoveStringLiteralDfa5_0(active0, 0x40000000L, active1, 0L);
   case 84:
   case 116:
      return jjMoveStringLiteralDfa5_0(active0, 0x10000000000000L, active1, 0L);
   default :
      break;
 }
 return jjStartNfa_0(3, active0, active1);
}
private final int jjMoveStringLiteralDfa5_0(long old0, long active0, long old1, long active1)
{
 if (((active0 &= old0) | (active1 &= old1)) == 0L)
   return jjStartNfa_0(3, old0, old1);
 try { curChar = input_stream.readChar(); }
 catch(java.io.IOException e) {
   jjStopStringLiteralDfa_0(4, active0, active1);
   return 5;
 }
 switch(curChar)
 {
   case 95:
      return jjMoveStringLiteralDfa6_0(active0, 0x10000000000000L, active1, 0L);
   case 69:
   case 101:
      if ((active0 & 0x10000000L) != 0L)
         return jjStartNfaWithStates_0(5, 28, 19);
      else if ((active0 & 0x20000000L) != 0L)
         return jjStartNfaWithStates_0(5, 29, 19);
      else if ((active0 & 0x40000000L) != 0L)
         return jjStartNfaWithStates_0(5, 30, 19);
      else if ((active0 & 0x40000000000000L) != 0L)
         return jjStartNfaWithStates_0(5, 54, 19);
      return jjMoveStringLiteralDfa6_0(active0, 0x800000000000000L, active1, 0L);
   case 70:
   case 102:
      if ((active0 & 0x100000L) != 0L)
         return jjStartNfaWithStates_0(5, 20, 19);
      break;
   case 71:
   case 103:
      return jjMoveStringLiteralDfa6_0(active0, 0x2000000000000L, active1, 0L);
   case 78:
   case 110:
      return jjMoveStringLiteralDfa6_0(active0, 0x4008000000L, active1, 0L);
   case 79:
   case 111:
      return jjMoveStringLiteralDfa6_0(active0, 0x40100000000L, active1, 0L);
   case 82:
   case 114:
      if ((active0 & 0x2000000000000000L) != 0L)
         return jjStartNfaWithStates_0(5, 61, 19);
      return jjMoveStringLiteralDfa6_0(active0, 0xc000000000000000L, active1, 0x2L);
   case 83:
   case 115:
      if ((active0 & 0x1000000L) != 0L)
         return jjStartNfaWithStates_0(5, 24, 19);
      break;
   case 84:
   case 116:
      if ((active0 & 0x200000000000L) != 0L)
         return jjStartNfaWithStates_0(5, 45, 19);
      return jjMoveStringLiteralDfa6_0(active0, 0x20000000000000L, active1, 0L);
   case 87:
   case 119:
      if ((active0 & 0x400000000000000L) != 0L)
         return jjStartNfaWithStates_0(5, 58, 19);
      break;
   default :
      break;
 }
 return jjStartNfa_0(4, active0, active1);
}
private final int jjMoveStringLiteralDfa6_0(long old0, long active0, long old1, long active1)
{
 if (((active0 &= old0) | (active1 &= old1)) == 0L)
   return jjStartNfa_0(4, old0, old1);
 try { curChar = input_stream.readChar(); }
 catch(java.io.IOException e) {
   jjStopStringLiteralDfa_0(5, active0, active1);
   return 6;
 }
 switch(curChar)
 {
   case 65:
   case 97:
      if ((active0 & 0x20000000000000L) != 0L)
         return jjStartNfaWithStates_0(6, 53, 19);
      break;
   case 69:
   case 101:
      if ((active1 & 0x2L) != 0L)
         return jjStartNfaWithStates_0(6, 65, 19);
      break;
   case 71:
   case 103:
      if ((active0 & 0x4000000000L) != 0L)
         return jjStartNfaWithStates_0(6, 38, 19);
      return jjMoveStringLiteralDfa7_0(active0, 0x10000008000000L, active1, 0L);
   case 78:
   case 110:
      if ((active0 & 0x100000000L) != 0L)
         return jjStartNfaWithStates_0(6, 32, 19);
      else if ((active0 & 0x40000000000L) != 0L)
         return jjStartNfaWithStates_0(6, 42, 19);
      return jjMoveStringLiteralDfa7_0(active0, 0x800000000000000L, active1, 0L);
   case 79:
   case 111:
      return jjMoveStringLiteralDfa7_0(active0, 0xc000000000000000L, active1, 0L);
   case 82:
   case 114:
      return jjMoveStringLiteralDfa7_0(active0, 0x2000000000000L, active1, 0L);
   default :
      break;
 }
 return jjStartNfa_0(5, active0, active1);
}
private final int jjMoveStringLiteralDfa7_0(long old0, long active0, long old1, long active1)
{
 if (((active0 &= old0) | (active1 &= old1)) == 0L)
   return jjStartNfa_0(5, old0, old1);
 try { curChar = input_stream.readChar(); }
 catch(java.io.IOException e) {
   jjStopStringLiteralDfa_0(6, active0, 0L);
   return 7;
 }
 switch(curChar)
 {
   case 65:
   case 97:
      return jjMoveStringLiteralDfa8_0(active0, 0x2000000000000L);
   case 76:
   case 108:
      return jjMoveStringLiteralDfa8_0(active0, 0x8000000L);
   case 82:
   case 114:
      return jjMoveStringLiteralDfa8_0(active0, 0x10000000000000L);
   case 84:
   case 116:
      if ((active0 & 0x800000000000000L) != 0L)
         return jjStartNfaWithStates_0(7, 59, 19);
      break;
   case 85:
   case 117:
      return jjMoveStringLiteralDfa8_0(active0, 0xc000000000000000L);
   default :
      break;
```

```java
   }
   return jjStartNfa_0(6, active0, 0L);
}
private final int jjMoveStringLiteralDfa8_0(long old0, long active0)
{
   if (((active0 &= old0)) == 0L)
      return jjStartNfa_0(6, old0, 0L);
   try { curChar = input_stream.readChar(); }
   catch(java.io.IOException e) {
      jjStopStringLiteralDfa_0(7, active0, 0L);
      return 8;
   }
   switch(curChar)
   {
      case 65:
      case 97:
         return jjMoveStringLiteralDfa9_0(active0, 0x10000000000000L);
      case 69:
      case 101:
         if ((active0 & 0x8000000L) != 0L)
            return jjStartNfaWithStates_0(8, 27, 19);
         break;
      case 78:
      case 110:
         return jjMoveStringLiteralDfa9_0(active0, 0xc000000000000000L);
      case 89:
      case 121:
         if ((active0 & 0x2000000000000L) != 0L)
            return jjStartNfaWithStates_0(8, 49, 19);
         break;
      default :
         break;
   }
   return jjStartNfa_0(7, active0, 0L);
}
private final int jjMoveStringLiteralDfa9_0(long old0, long active0)
{
   if (((active0 &= old0)) == 0L)
      return jjStartNfa_0(7, old0, 0L);
   try { curChar = input_stream.readChar(); }
   catch(java.io.IOException e) {
      jjStopStringLiteralDfa_0(8, active0, 0L);
      return 9;
   }
   switch(curChar)
   {
      case 68:
      case 100:
         if ((active0 & 0x4000000000000000L) != 0L)
            return jjStartNfaWithStates_0(9, 62, 19);
         else if ((active0 & 0x8000000000000000L) != 0L)
```

```
            return jjStartNfaWithStates_0(9, 63, 19);
         break;
      case 89:
      case 121:
         if ((active0 & 0x10000000000000L) != 0L)
            return jjStartNfaWithStates_0(9, 52, 19);
         break;
      default :
         break;
   }
   return jjStartNfa_0(8, active0, 0L);
}
private final void jjCheckNAdd(int state)
{
   if (jjrounds[state] != jjround)
   {
      jjstateSet[jjnewStateCnt++] = state;
      jjrounds[state] = jjround;
   }
}
private final void jjAddStates(int start, int end)
{
   do {
      jjstateSet[jjnewStateCnt++] = jjnextStates[start];
   } while (start++ != end);
}
private final void jjCheckNAddTwoStates(int state1, int state2)
{
   jjCheckNAdd(state1);
   jjCheckNAdd(state2);
}
private final void jjCheckNAddStates(int start, int end)
{
   do {
      jjCheckNAdd(jjnextStates[start]);
   } while (start++ != end);
}
private final void jjCheckNAddStates(int start)
{
   jjCheckNAdd(jjnextStates[start]);
   jjCheckNAdd(jjnextStates[start + 1]);
}
static final long[] jjbitVec0 = {
   0xfffffffffffffffeL, 0xffffffffffffffffL, 0xffffffffffffffffL, 0xffffffffffffffffL
};
static final long[] jjbitVec2 = {
   0x0L, 0x0L, 0xffffffffffffffffL, 0xffffffffffffffffL
};
static final long[] jjbitVec3 = {
   0x1ff00000fffffffeL, 0xffffffffffffc000L, 0xffffffffL, 0x600000000000000L
};
```

```java
static final long[] jjbitVec4 = {
   0x0L, 0x0L, 0x0L, 0xff7fffffff7fffffL
};
static final long[] jjbitVec5 = {
   0x0L, 0xffffffffffffffffL, 0xffffffffffffffffL, 0xffffffffffffffffL
};
static final long[] jjbitVec6 = {
   0xffffffffffffffffL, 0xffffffffffffffffL, 0xffffL, 0x0L
};
static final long[] jjbitVec7 = {
   0xffffffffffffffffL, 0xffffffffffffffffL, 0x0L, 0x0L
};
static final long[] jjbitVec8 = {
   0x3fffffffffffL, 0x0L, 0x0L, 0x0L
};
private final int jjMoveNfa_0(int startState, int curPos)
{
   int[] nextStates;
   int startsAt = 0;
   jjnewStateCnt = 24;
   int i = 1;
   jjstateSet[0] = startState;
   int j, kind = 0x7fffffff;
   for (;;)
   {
      if (++jjround == 0x7fffffff)
         ReInitRounds();
      if (curChar < 64)
      {
         long l = 1L << curChar;
         MatchLoop: do
         {
            switch(jjstateSet[--i])
            {
               case 3:
                  if ((0x3ff000000000000L & l) != 0L)
                     jjCheckNAddTwoStates(6, 7);
                  else if (curChar == 36)
                  {
                     if (kind > 73)
                        kind = 73;
                     jjCheckNAdd(19);
                  }
                  else if (curChar == 34)
                     jjCheckNAddStates(0, 2);
                  else if (curChar == 47)
                     jjstateSet[jjnewStateCnt++] = 2;
                  if ((0x3fe000000000000L & l) != 0L)
                  {
                     if (kind > 67)
                        kind = 67;
                     jjCheckNAdd(5);
                  }
                  else if (curChar == 48)
                  {
                     if (kind > 67)
                        kind = 67;
                     jjCheckNAddTwoStates(21, 23);
                  }
                  break;
               case 0:
                  if (curChar == 42)
                     jjstateSet[jjnewStateCnt++] = 1;
                  break;
               case 1:
                  if ((0xffff7fffffffffffL & l) != 0L && kind > 8)
                     kind = 8;
                  break;
               case 2:
                  if (curChar == 42)
                     jjstateSet[jjnewStateCnt++] = 0;
                  break;
               case 4:
                  if ((0x3fe000000000000L & l) == 0L)
                     break;
                  if (kind > 67)
                     kind = 67;
                  jjCheckNAdd(5);
                  break;
               case 5:
                  if ((0x3ff000000000000L & l) == 0L)
                     break;
                  if (kind > 67)
                     kind = 67;
                  jjCheckNAdd(5);
                  break;
               case 6:
                  if ((0x3ff000000000000L & l) != 0L)
                     jjCheckNAddTwoStates(6, 7);
                  break;
               case 7:
                  if (curChar != 46)
                     break;
                  if (kind > 71)
                     kind = 71;
                  jjCheckNAdd(8);
                  break;
               case 8:
                  if ((0x3ff000000000000L & l) == 0L)
                     break;
                  if (kind > 71)
                     kind = 71;
                  jjCheckNAdd(8);
                  break;
               case 9:
                  if (curChar == 34)
                     jjCheckNAddStates(0, 2);
                  break;
               case 10:
                  if ((0xfffffffbffffdbffL & l) != 0L)
                     jjCheckNAddStates(0, 2);
                  break;
               case 12:
                  if ((0x8400000000L & l) != 0L)
                     jjCheckNAddStates(0, 2);
                  break;
               case 13:
                  if (curChar == 34 && kind > 72)
                     kind = 72;
                  break;
               case 14:
                  if ((0xff000000000000L & l) != 0L)
                     jjCheckNAddStates(3, 6);
                  break;
               case 15:
                  if ((0xff000000000000L & l) != 0L)
                     jjCheckNAddStates(0, 2);
                  break;
               case 16:
                  if ((0xf000000000000L & l) != 0L)
                     jjstateSet[jjnewStateCnt++] = 17;
                  break;
               case 17:
                  if ((0xff000000000000L & l) != 0L)
                     jjCheckNAdd(15);
                  break;
               case 18:
                  if (curChar != 36)
                     break;
                  if (kind > 73)
                     kind = 73;
                  jjCheckNAdd(19);
                  break;
               case 19:
                  if ((0x3ff001000000000L & l) == 0L)
                     break;
                  if (kind > 73)
                     kind = 73;
                  jjCheckNAdd(19);
                  break;
               case 20:
                  if (curChar != 48)
                     break;
                  if (kind > 67)
                     kind = 67;
                  jjCheckNAddTwoStates(21, 23);
                  break;
               case 22:
                  if ((0x3ff000000000000L & l) == 0L)
                     break;
                  if (kind > 67)
                     kind = 67;
                  jjstateSet[jjnewStateCnt++] = 22;
                  break;
               case 23:
                  if ((0xff000000000000L & l) == 0L)
                     break;
                  if (kind > 67)
                     kind = 67;
                  jjCheckNAdd(23);
                  break;
               default : break;
            }
         } while(i != startsAt);
      }
      else if (curChar < 128)
      {
         long l = 1L << (curChar & 077);
         MatchLoop: do
         {
            switch(jjstateSet[--i])
            {
               case 3:
               case 19:
                  if ((0x7fffffe87fffffeL & l) == 0L)
                     break;
                  if (kind > 73)
                     kind = 73;
                  jjCheckNAdd(19);
                  break;
               case 1:
                  if (kind > 8)
                     kind = 8;
                  break;
               case 10:
                  if ((0xffffffffefffffffL & l) != 0L)
                     jjCheckNAddStates(0, 2);
                  break;
               case 11:
                  if (curChar == 92)
                     jjAddStates(7, 9);
                  break;
               case 12:
                  if ((0x14404410144044L & l) != 0L)
                     jjCheckNAddStates(0, 2);
                  break;
               case 21:
                  if ((0x100000001000000L & l) != 0L)
                     jjCheckNAdd(22);
                  break;
               case 22:
                  if ((0x7e0000007eL & l) == 0L)
                     break;
                  if (kind > 67)
                     kind = 67;
                  jjCheckNAdd(22);
                  break;
               default : break;
            }
         } while(i != startsAt);
      }
      else
      {
         int hiByte = (int)(curChar >> 8);
         int i1 = hiByte >> 6;
         long l1 = 1L << (hiByte & 077);
         int i2 = (curChar & 0xff) >> 6;
         long l2 = 1L << (curChar & 077);
         MatchLoop: do
         {
            switch(jjstateSet[--i])
            {
               case 3:
               case 19:
                  if (!jjCanMove_1(hiByte, i1, i2, l1, l2))
                     break;
                  if (kind > 73)
                     kind = 73;
                  jjCheckNAdd(19);
                  break;
               case 1:
                  if (jjCanMove_0(hiByte, i1, i2, l1, l2) && kind > 8)
                     kind = 8;
                  break;
               case 10:
                  if (jjCanMove_0(hiByte, i1, i2, l1, l2))
                     jjAddStates(0, 2);
                  break;
               default : break;
            }
         } while(i != startsAt);
      }
      if (kind != 0x7fffffff)
      {
         jjmatchedKind = kind;
```

```
         jjmatchedPos = curPos;
         kind = 0x7fffffff;
      }
      ++curPos;
      if ((i = jjnewStateCnt) == (startsAt = 24 - (jjnewStateCnt = startsAt)))
         return curPos;
      try { curChar = input_stream.readChar(); }
      catch(java.io.IOException e) { return curPos; }
   }
}
private final int jjMoveStringLiteralDfa0_3()
{
   switch(curChar)
   {
      case 42:
         return jjMoveStringLiteralDfa1_3(0x1000L);
      default :
         return 1;
   }
}
private final int jjMoveStringLiteralDfa1_3(long active0)
{
   try { curChar = input_stream.readChar(); }
   catch(java.io.IOException e) {
      return 1;
   }
   switch(curChar)
   {
      case 47:
         if ((active0 & 0x1000L) != 0L)
            return jjStopAtPos(1, 12);
         break;
      default :
         return 2;
   }
   return 2;
}
private final int jjMoveStringLiteralDfa0_1()
{
   return jjMoveNfa_1(0, 0);
}
private final int jjMoveNfa_1(int startState, int curPos)
{
   int[] nextStates;
   int startsAt = 0;
   jjnewStateCnt = 3;
   int i = 1;
   jjstateSet[0] = startState;
   int j, kind = 0x7fffffff;
   for (;;)
   {
```

```java
      if (++jjround == 0x7fffffff)
         ReInitRounds();
      if (curChar < 64)
      {
         long l = 1L << curChar;
         MatchLoop: do
         {
            switch(jjstateSet[--i])
            {
               case 0:
                  if ((0x2400L & l) != 0L)
                  {
                     if (kind > 10)
                        kind = 10;
                  }
                  if (curChar == 13)
                     jjstateSet[jjnewStateCnt++] = 1;
                  break;
               case 1:
                  if (curChar == 10 && kind > 10)
                     kind = 10;
                  break;
               case 2:
                  if (curChar == 13)
                     jjstateSet[jjnewStateCnt++] = 1;
                  break;
               default : break;
            }
         } while(i != startsAt);
      }
      else if (curChar < 128)
      {
         long l = 1L << (curChar & 077);
         MatchLoop: do
         {
            switch(jjstateSet[--i])
            {
               default : break;
            }
         } while(i != startsAt);
      }
      else
      {
         int hiByte = (int)(curChar >> 8);
         int i1 = hiByte >> 6;
         long l1 = 1L << (hiByte & 077);
         int i2 = (curChar & 0xff) >> 6;
         long l2 = 1L << (curChar & 077);
         MatchLoop: do
         {
            switch(jjstateSet[--i])
            {
               default : break;
            }
         } while(i != startsAt);
      }
      if (kind != 0x7fffffff)
      {
         jjmatchedKind = kind;
         jjmatchedPos = curPos;
         kind = 0x7fffffff;
      }
      ++curPos;
      if ((i = jjnewStateCnt) == (startsAt = 3 - (jjnewStateCnt = startsAt)))
         return curPos;
      try { curChar = input_stream.readChar(); }
      catch(java.io.IOException e) { return curPos; }
   }
}
private final int jjMoveStringLiteralDfa0_2()
{
   switch(curChar)
   {
      case 42:
         return jjMoveStringLiteralDfa1_2(0x800L);
      default :
         return 1;
   }
}
private final int jjMoveStringLiteralDfa1_2(long active0)
{
   try { curChar = input_stream.readChar(); }
   catch(java.io.IOException e) {
      return 1;
   }
   switch(curChar)
   {
      case 47:
         if ((active0 & 0x800L) != 0L)
            return jjStopAtPos(1, 11);
         break;
      default :
         return 2;
   }
   return 2;
}
static final int[] jjnextStates = {
   10, 11, 13, 10, 11, 15, 13, 12, 14, 16,
};
private static final boolean jjCanMove_0(int hiByte, int i1, int i2, long l1, long l2)
{
   switch(hiByte)
   {
      case 0:
         return ((jjbitVec2[i2] & l2) != 0L);
      default :
         if ((jjbitVec0[i1] & l1) != 0L)
            return true;
         return false;
   }
}
private static final boolean jjCanMove_1(int hiByte, int i1, int i2, long l1, long l2)
{
   switch(hiByte)
   {
      case 0:
         return ((jjbitVec4[i2] & l2) != 0L);
      case 48:
         return ((jjbitVec5[i2] & l2) != 0L);
      case 49:
         return ((jjbitVec6[i2] & l2) != 0L);
      case 51:
         return ((jjbitVec7[i2] & l2) != 0L);
      case 61:
         return ((jjbitVec8[i2] & l2) != 0L);
      default :
         if ((jjbitVec3[i1] & l1) != 0L)
            return true;
         return false;
   }
}
public static final String[] jjstrLiteralImages = {
"", null, null, null, null, null, null, null, null, null, null, null, null,
null, null, null, null, null, null, null, null, null, null, null, null, null, null,
null, null, null, null, null, null, null, null, null, null, null, null, null, null,
null, null, null, null, null, null, null, null, null, null, null, null, null, null,
null, null, null, null, null, null, null, null, null, null, null, null, null, null,
null, null, null, null, null, null, null, "\50", "\51", "\73", "\72", "\54", "\75",
"\76", "\74", "\41", "\75\75", "\74\75", "\76\75", "\41\75", "\174\174", "\46\46",
"\53", "\55", "\52", "\57", "\45", };
public static final String[] lexStateNames = {
   "DEFAULT",
   "IN_SINGLE_LINE_COMMENT",
   "IN_FORMAL_COMMENT",
   "IN_MULTI_LINE_COMMENT",
};
public static final int[] jjnewLexState = {
   -1, -1, -1, -1, -1, -1, 1, 1, 2, 3, 0, 0, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
   -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
   -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
   -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
};
static final long[] jjtoToken = {
   0xfffffffffffc001L, 0xfffff38fL,
};
static final long[] jjtoSkip = {
   0x1c3eL, 0x0L,
};
static final long[] jjtoSpecial = {
   0x1c3eL, 0x0L,
};
static final long[] jjtoMore = {
   0x23c0L, 0x0L,
};
protected JavaCharStream input_stream;
private final int[] jjrounds = new int[24];
private final int[] jjstateSet = new int[48];
StringBuffer image;
int jjimageLen;
int lengthOfMatch;
protected char curChar;
public JGSL_ParserTokenManager(JavaCharStream stream)
{
   if (JavaCharStream.staticFlag)
      throw new Error("ERROR: Cannot use a static CharStream class with a non-static lexical analyzer.");
   input_stream = stream;
}
public JGSL_ParserTokenManager(JavaCharStream stream, int lexState)
{
   this(stream);
   SwitchTo(lexState);
}
public void ReInit(JavaCharStream stream)
{
   jjmatchedPos = jjnewStateCnt = 0;
   curLexState = defaultLexState;
   input_stream = stream;
   ReInitRounds();
}
private final void ReInitRounds()
{
   int i;
   jjround = 0x80000001;
   for (i = 24; i-- > 0;)
      jjrounds[i] = 0x80000000;
}
public void ReInit(JavaCharStream stream, int lexState)
{
   ReInit(stream);
   SwitchTo(lexState);
}
public void SwitchTo(int lexState)
{
   if (lexState >= 4 || lexState < 0)
```

```
      throw new TokenMgrError("Error: Ignoring invalid lexical state : " + lexState + ". State unchanged.", TokenMgrError.INVALID_LEXICAL_STATE);
   else
      curLexState = lexState;
}

protected Token jjFillToken()
{
   Token t = Token.newToken(jjmatchedKind);
   t.kind = jjmatchedKind;
   String im = jjstrLiteralImages[jjmatchedKind];
   t.image = (im == null) ? input_stream.GetImage() : im;
   t.beginLine = input_stream.getBeginLine();
   t.beginColumn = input_stream.getBeginColumn();
   t.endLine = input_stream.getEndLine();
   t.endColumn = input_stream.getEndColumn();
   return t;
}

int curLexState = 0;
int defaultLexState = 0;
int jjnewStateCnt;
int jjround;
int jjmatchedPos;
int jjmatchedKind;

public Token getNextToken()
{
  int kind;
  Token specialToken = null;
  Token matchedToken;
  int curPos = 0;

  EOFLoop :
  for (;;)
  {
   try
   {
      curChar = input_stream.BeginToken();
   }
   catch(java.io.IOException e)
   {
      jjmatchedKind = 0;
      matchedToken = jjFillToken();
      matchedToken.specialToken = specialToken;
      return matchedToken;
   }
   image = null;
   jjimageLen = 0;

```

```
for (;;)
{
  switch(curLexState)
  {
    case 0:
      jjmatchedKind = 0x7fffffff;
      jjmatchedPos = 0;
      curPos = jjMoveStringLiteralDfa0_0();
      break;
    case 1:
      jjmatchedKind = 0x7fffffff;
      jjmatchedPos = 0;
      curPos = jjMoveStringLiteralDfa0_1();
      if (jjmatchedPos == 0 && jjmatchedKind > 13)
      {
         jjmatchedKind = 13;
      }
      break;
    case 2:
      jjmatchedKind = 0x7fffffff;
      jjmatchedPos = 0;
      curPos = jjMoveStringLiteralDfa0_2();
      if (jjmatchedPos == 0 && jjmatchedKind > 13)
      {
         jjmatchedKind = 13;
      }
      break;
    case 3:
      jjmatchedKind = 0x7fffffff;
      jjmatchedPos = 0;
      curPos = jjMoveStringLiteralDfa0_3();
      if (jjmatchedPos == 0 && jjmatchedKind > 13)
      {
         jjmatchedKind = 13;
      }
      break;
  }
  if (jjmatchedKind != 0x7fffffff)
  {
     if (jjmatchedPos + 1 < curPos)
        input_stream.backup(curPos - jjmatchedPos - 1);
     if ((jjtoToken[jjmatchedKind >> 6] & (1L << (jjmatchedKind & 077))) != 0L)
     {
        matchedToken = jjFillToken();
        matchedToken.specialToken = specialToken;
    if (jjnewLexState[jjmatchedKind] != -1)
      curLexState = jjnewLexState[jjmatchedKind];
        return matchedToken;
     }
     else if ((jjtoSkip[jjmatchedKind >> 6] & (1L << (jjmatchedKind & 077))) != 0L)
     {
          if ((jjtoSpecial[jjmatchedKind >> 6] & (1L << (jjmatchedKind & 077))) != 0L)
          {
             matchedToken = jjFillToken();
             if (specialToken == null)
                specialToken = matchedToken;
             else
             {
                matchedToken.specialToken = specialToken;
                specialToken = (specialToken.next = matchedToken);
             }
             SkipLexicalActions(matchedToken);
          }
          else
             SkipLexicalActions(null);
        if (jjnewLexState[jjmatchedKind] != -1)
          curLexState = jjnewLexState[jjmatchedKind];
          continue EOFLoop;
       }
       MoreLexicalActions();
     if (jjnewLexState[jjmatchedKind] != -1)
       curLexState = jjnewLexState[jjmatchedKind];
       curPos = 0;
       jjmatchedKind = 0x7fffffff;
       try {
          curChar = input_stream.readChar();
          continue;
       }
       catch (java.io.IOException e1) { }
    }
    int error_line = input_stream.getEndLine();
    int error_column = input_stream.getEndColumn();
    String error_after = null;
    boolean EOFSeen = false;
    try { input_stream.readChar(); input_stream.backup(1); }
    catch (java.io.IOException e1) {
       EOFSeen = true;
       error_after = curPos <= 1 ? "" : input_stream.GetImage();
       if (curChar == '\n' || curChar == '\r') {
          error_line++;
          error_column = 0;
       }
       else
          error_column++;
    }
    if (!EOFSeen) {
       input_stream.backup(1);
       error_after = curPos <= 1 ? "" : input_stream.GetImage();
    }
    throw new TokenMgrError(EOFSeen, curLexState, error_line, error_column, error_after, curChar, TokenMgrError.LEXICAL_ERROR);
  }
  }
}

void SkipLexicalActions(Token matchedToken)
{
   switch(jjmatchedKind)
   {
      default :
         break;
   }
}
void MoreLexicalActions()
{
   jjimageLen += (lengthOfMatch = jjmatchedPos + 1);
   switch(jjmatchedKind)
   {
      case 8 :
         if (image == null)
              image = new StringBuffer(new String(input_stream.GetSuffix(jjimageLen)));
         else
            image.append(input_stream.GetSuffix(jjimageLen));
         jjimageLen = 0;
                     input_stream.backup(1);
         break;
      default :
         break;
   }
}
}

```

/Users/jchavez/dev/java.net/jgsl.dev.java.net/jgsl/dev/src/jgsl/parser/JavaCharStream.java

```java
    /* Generated By:JavaCC: Do not edit this line. JavaCharStream.java Version 3.0 */
package jgsl.parser;

/**
 * An implementation of interface CharStream, where the stream is assumed to contain only ASCII characters 
(with
 * java-like unicode escape processing).
 */

public class JavaCharStream {
    public static final boolean staticFlag = false;

    static final int hexval(char c) throws java.io.IOException {
        switch (c) {
            case '0' :
                return 0;
            case '1' :
                return 1;
            case '2' :
                return 2;
            case '3' :
                return 3;
            case '4' :
                return 4;
            case '5' :
                return 5;
            case '6' :
                return 6;
            case '7' :
                return 7;
            case '8' :
                return 8;
            case '9' :
                return 9;

            case 'a' :
            case 'A' :
                return 10;
            case 'b' :
            case 'B' :
                return 11;
            case 'c' :
            case 'C' :
                return 12;
            case 'd' :
```

```
            case 'D' :
               return 13;
            case 'e' :
            case 'E' :
               return 14;
            case 'f' :
            case 'F' :
               return 15;
         }

         throw new java.io.IOException(); // Should never come here
      }

      public int bufpos = -1;
      int bufsize;
      int available;
      int tokenBegin;
      protected int bufline[];
      protected int bufcolumn[];

      protected int column = 0;
      protected int line = 1;

      protected boolean prevCharIsCR = false;
      protected boolean prevCharIsLF = false;

      protected java.io.Reader inputStream;

      protected char[] nextCharBuf;
      protected char[] buffer;
      protected int maxNextCharInd = 0;
      protected int nextCharInd = -1;
      protected int inBuf = 0;

      protected void ExpandBuff(boolean wrapAround) {
         char[] newbuffer = new char[bufsize + 2048];
         int newbufline[] = new int[bufsize + 2048];
         int newbufcolumn[] = new int[bufsize + 2048];

         try {
            if (wrapAround) {
               System.arraycopy(buffer, tokenBegin, newbuffer, 0, bufsize - tokenBegin);
               System.arraycopy(buffer, 0, newbuffer,
                     bufsize - tokenBegin, bufpos);
               buffer = newbuffer;

               System.arraycopy(bufline, tokenBegin, newbufline, 0, bufsize - tokenBegin);
               System.arraycopy(bufline, 0, newbufline, bufsize - tokenBegin, bufpos);
               bufline = newbufline;
```

```

              System.arraycopy(bufcolumn, tokenBegin, newbufcolumn, 0, bufsize - tokenBegin);
              System.arraycopy(bufcolumn, 0, newbufcolumn, bufsize - tokenBegin, bufpos);
              bufcolumn = newbufcolumn;

              bufpos += (bufsize - tokenBegin);
           } else {
               System.arraycopy(buffer, tokenBegin, newbuffer, 0, bufsize - tokenBegin);
               buffer = newbuffer;

               System.arraycopy(bufline, tokenBegin, newbufline, 0, bufsize - tokenBegin);
               bufline = newbufline;

               System.arraycopy(bufcolumn, tokenBegin, newbufcolumn, 0, bufsize - tokenBegin);
               bufcolumn = newbufcolumn;

               bufpos -= tokenBegin;
           }
        }
        catch (Throwable t) {
           throw new Error(t.getMessage());
        }

        available = (bufsize += 2048);
        tokenBegin = 0;
     }

     protected void FillBuff() throws java.io.IOException {
        int i;
        if (maxNextCharInd == 4096)
           maxNextCharInd = nextCharInd = 0;

        try {
           if ((i = inputStream.read(nextCharBuf, maxNextCharInd,
                 4096 - maxNextCharInd)) == -1) {
              inputStream.close();
              throw new java.io.IOException();
           } else
              maxNextCharInd += i;
           return;
        }
        catch (java.io.IOException e) {
           if (bufpos != 0) {
              --bufpos;
              backup(0);
           } else {
              bufline[bufpos] = line;
              bufcolumn[bufpos] = column;
           }
           throw e;
        }
     }

     protected char ReadByte() throws java.io.IOException {
        if (++nextCharInd >= maxNextCharInd)
           FillBuff();

        return nextCharBuf[nextCharInd];
     }

     public char BeginToken() throws java.io.IOException {
        if (inBuf > 0) {
           --inBuf;

           if (++bufpos == bufsize)
             bufpos = 0;

           tokenBegin = bufpos;
           return buffer[bufpos];
        }

        tokenBegin = 0;
        bufpos = -1;

        return readChar();
     }

     protected void AdjustBuffSize() {
        if (available == bufsize) {
           if (tokenBegin > 2048) {
             bufpos = 0;
             available = tokenBegin;
           } else
             ExpandBuff(false);
        } else if (available > tokenBegin)
           available = bufsize;
        else if ((tokenBegin - available) < 2048)
           ExpandBuff(true);
        else
           available = tokenBegin;
     }

     protected void UpdateLineColumn(char c) {
        column++;

        if (prevCharIsLF) {
           prevCharIsLF = false;
           line += (column = 1);
        } else if (prevCharIsCR) {
           prevCharIsCR = false;
           if (c == '\n') {
              prevCharIsLF = true;
           } else
              line += (column = 1);
        }

        switch (c) {
           case '\r' :
              prevCharIsCR = true;
              break;
           case '\n' :
              prevCharIsLF = true;
              break;
           case '\t' :
              column--;
              column += (8 - (column & 07));
              break;
           default :
              break;
        }

        bufline[bufpos] = line;
        bufcolumn[bufpos] = column;
     }

     public char readChar() throws java.io.IOException {
        if (inBuf > 0) {
           --inBuf;

           if (++bufpos == bufsize)
              bufpos = 0;

           return buffer[bufpos];
        }

        char c;

        if (++bufpos == available)
           AdjustBuffSize();

        if ((buffer[bufpos] = c = ReadByte()) == '\\') {
           UpdateLineColumn(c);

           int backSlashCnt = 1;

           for (; ;) // Read all the backslashes
           {
                if (++bufpos == available)
                    AdjustBuffSize();

                try {
                    if ((buffer[bufpos] = c = ReadByte()) != '\\') {
                        UpdateLineColumn(c);
                        // found a non-backslash char.
                        if ((c == 'u') && ((backSlashCnt & 1) == 1)) {
                            if (--bufpos < 0)
                                bufpos = bufsize - 1;

                            break;
                        }

                        backup(backSlashCnt);
                        return '\\';
                    }
                }
                catch (java.io.IOException e) {
                    if (backSlashCnt > 1)
                        backup(backSlashCnt);

                    return '\\';
                }

                UpdateLineColumn(c);
                backSlashCnt++;
            }

            // Here, we have seen an odd number of backslash's followed by a 'u'
            try {
                while ((c = ReadByte()) == 'u') ++column;

                buffer[bufpos] = c = (char) (hexval(c) << 12 |
                        hexval(ReadByte()) << 8 |
                        hexval(ReadByte()) << 4 |
                        hexval(ReadByte()));

                column += 4;
            }
            catch (java.io.IOException e) {
                throw new Error("Invalid escape character at line " + line +
                        " column " + column + ".");
            }

            if (backSlashCnt == 1)
                return c;
            else {
                backup(backSlashCnt - 1);
            return '\\';
          }
      } else {
          UpdateLineColumn(c);
          return (c);
      }
  }

  /**
   * @see #getEndColumn
   * @deprecated
   */

  public int getColumn() {
      return bufcolumn[bufpos];
  }

  /**
   * @see #getEndLine
   * @deprecated
   */

  public int getLine() {
      return bufline[bufpos];
  }

  public int getEndColumn() {
      return bufcolumn[bufpos];
  }

  public int getEndLine() {
      return bufline[bufpos];
  }

  public int getBeginColumn() {
      return bufcolumn[tokenBegin];
  }

  public int getBeginLine() {
      return bufline[tokenBegin];
  }

  public void backup(int amount) {

      inBuf += amount;
      if ((bufpos -= amount) < 0)
          bufpos += bufsize;
  }

  public JavaCharStream(java.io.Reader dstream,
                        int startline, int startcolumn, int buffersize) {
      inputStream = dstream;
      line = startline;
      column = startcolumn - 1;

      available = bufsize = buffersize;
      buffer = new char[buffersize];
      bufline = new int[buffersize];
      bufcolumn = new int[buffersize];
      nextCharBuf = new char[4096];
  }

  public JavaCharStream(java.io.Reader dstream,
                        int startline, int startcolumn) {
      this(dstream, startline, startcolumn, 4096);
  }

  public JavaCharStream(java.io.Reader dstream) {
      this(dstream, 1, 1, 4096);
  }

  public void ReInit(java.io.Reader dstream,
                     int startline, int startcolumn, int buffersize) {
      inputStream = dstream;
      line = startline;
      column = startcolumn - 1;

      if (buffer == null || buffersize != buffer.length) {
          available = bufsize = buffersize;
          buffer = new char[buffersize];
          bufline = new int[buffersize];
          bufcolumn = new int[buffersize];
          nextCharBuf = new char[4096];
      }
      prevCharIsLF = prevCharIsCR = false;
      tokenBegin = inBuf = maxNextCharInd = 0;
      nextCharInd = bufpos = -1;
  }

  public void ReInit(java.io.Reader dstream,
                     int startline, int startcolumn) {
      ReInit(dstream, startline, startcolumn, 4096);
  }

  public void ReInit(java.io.Reader dstream) {
      ReInit(dstream, 1, 1, 4096);
  }

```

```java
  public JavaCharStream(java.io.InputStream dstream, int startline,
                        int startcolumn, int buffersize) {
     this(new java.io.InputStreamReader(dstream), startline, startcolumn, 4096);
  }

  public JavaCharStream(java.io.InputStream dstream, int startline,
                        int startcolumn) {
     this(dstream, startline, startcolumn, 4096);
  }

  public JavaCharStream(java.io.InputStream dstream) {
     this(dstream, 1, 1, 4096);
  }

  public void ReInit(java.io.InputStream dstream, int startline,
                     int startcolumn, int buffersize) {
     ReInit(new java.io.InputStreamReader(dstream), startline, startcolumn, 4096);
  }

  public void ReInit(java.io.InputStream dstream, int startline,
                     int startcolumn) {
     ReInit(dstream, startline, startcolumn, 4096);
  }

  public void ReInit(java.io.InputStream dstream) {
     ReInit(dstream, 1, 1, 4096);
  }

  public String GetImage() {
     if (bufpos >= tokenBegin)
        return new String(buffer, tokenBegin, bufpos - tokenBegin + 1);
     else
        return new String(buffer, tokenBegin, bufsize - tokenBegin) +
              new String(buffer, 0, bufpos + 1);
  }

  public char[] GetSuffix(int len) {
     char[] ret = new char[len];

     if ((bufpos + 1) >= len)
        System.arraycopy(buffer, bufpos - len + 1, ret, 0, len);
     else {
        System.arraycopy(buffer, bufsize - (len - bufpos - 1), ret, 0,
              len - bufpos - 1);
        System.arraycopy(buffer, 0, ret, len - bufpos - 1, bufpos + 1);
     }

     return ret;
  }
```

```java

    public void Done() {
        nextCharBuf = null;
        buffer = null;
        bufline = null;
        bufcolumn = null;
    }

    /**
     * Method to adjust line and column numbers for the start of a token.
     */
    public void adjustBeginLineColumn(int newLine, int newCol) {
        int start = tokenBegin;
        int len;

        if (bufpos >= tokenBegin) {
            len = bufpos - tokenBegin + inBuf + 1;
        } else {
            len = bufsize - tokenBegin + bufpos + 1 + inBuf;
        }

        int i = 0, j = 0, k = 0;
        int nextColDiff = 0, columnDiff = 0;

        while (i < len &&
                bufline[j = start % bufsize] == bufline[k = ++start % bufsize]) {
            bufline[j] = newLine;
            nextColDiff = columnDiff + bufcolumn[k] - bufcolumn[j];
            bufcolumn[j] = newCol + columnDiff;
            columnDiff = nextColDiff;
            i++;
        }

        if (i < len) {
            bufline[j] = newLine++;
            bufcolumn[j] = newCol + columnDiff;

            while (i++ < len) {
                if (bufline[j = start % bufsize] != bufline[++start % bufsize])
                    bufline[j] = newLine++;
                else
                    bufline[j] = newLine;
            }
        }

        line = bufline[j];
        column = bufcolumn[j];
    }

```

```
}

```

/Users/jchavez/dev/java.net/jgsl.dev.java.net/jgsl/dev/src/jgsl/model/Type.java

```java
     /*
 * Copyright (c) 2005 Perception Software. All Rights Reserved.
 */
package jgsl.model;

// TODO - write javadocs
/**
 * @author zenarchitect
 * @version $Id: Type.java,v 1.2 2005/05/16 00:54:19 zenarchitect Exp $
  */
 public interface Type {
    /**
     * Get the java Class meta-data for this type
     *
     * @return The Class mete-data for this type
     */
    public Class getJavaClass();

    /**
     * Get the Java type as a String
     *
     * @return a String containing the type
     */
    public String getJavaType();

 }
```

/Users/jchavez/dev/java.net/jgsl.dev.java.net/jgsl/dev/src/jgsl/model/Value.java

```
/*
 * Copyright (c) 2005 Perception Software. All Rights Reserved.
 */
package jgsl.model;

// TODO - write javadocs
/**
 * @author zenarchitect
 * @version $Id: Value.java,v 1.2 2005/05/16 00:54:19 zenarchitect Exp $
 */
public interface Value {
    /**
     * Get the Java representation of this value
     *
     * @return A String containing the Java representation of this value
     */
    public String getJavaValue();
}

```

/Users/jchavez/dev/java.net/jgsl.dev.java.net/jgsl/dev/src/jgsl/model/Argument.java

```
/*
 * Copyright (c) 2005 Perception Software. All Rights Reserved.
 */
package jgsl.model;


/**
 * The name of a script argument.
 *
 * @author zenarchitect
 * @version $Id: Argument.java,v 1.2 2005/05/16 00:54:17 zenarchitect Exp $
 */
public interface Argument {
    /**
     * Get the name of the argument
     *
     * @return String containing the name
     */
    public String getName();

}

```

/Users/jchavez/dev/java.net/jgsl.dev.java.net/jgsl/dev/src/jgsl/parser/TokenMgrError.java

```
 /* Generated By:JavaCC: Do not edit this line. TokenMgrError.java Version 3.0 */
package jgsl.parser;

public class TokenMgrError extends Error {
   /*
    * Ordinals for various reasons why an Error of this type can be thrown.
    */

   /**
    * Lexical error occured.
    */
   static final int LEXICAL_ERROR = 0;

   /**
    * An attempt wass made to create a second instance of a static token manager.
    */
   static final int STATIC_LEXER_ERROR = 1;

   /**
    * Tried to change to an invalid lexical state.
    */
   static final int INVALID_LEXICAL_STATE = 2;

   /**
    * Detected (and bailed out of) an infinite loop in the token manager.
    */
   static final int LOOP_DETECTED = 3;

   /**
    * Indicates the reason why the exception is thrown. It will have one of the above 4 values.
    */
   int errorCode;

   /**
    * Replaces unprintable characters by their espaced (or unicode escaped) equivalents in the given string
    */
   protected static final String addEscapes(String str) {
      StringBuffer retval = new StringBuffer();
      char ch;
      for (int i = 0; i < str.length(); i++) {
         switch (str.charAt(i)) {
            case 0 :
               continue;
            case '\b':
               retval.append("\\b");
               continue;
            case '\t':
               retval.append("\\t");
```

```java
                 continue;
              case '\n':
                 retval.append("\\n");
                 continue;
              case '\f':
                 retval.append("\\f");
                 continue;
              case '\r':
                 retval.append("\\r");
                 continue;
              case '\"':
                 retval.append("\\\"");
                 continue;
              case '\'':
                 retval.append("\\\'");
                 continue;
              case '\\':
                 retval.append("\\\\");
                 continue;
              default:
                 if ((ch = str.charAt(i)) < 0x20 || ch > 0x7e) {
                    String s = "0000" + Integer.toString(ch, 16);
                    retval.append("\\u" + s.substring(s.length() - 4, s.length()));
                 } else {
                    retval.append(ch);
                 }
                 continue;
           }
        }
        return retval.toString();
     }

     /**
      * Returns a detailed message for the Error when it is thrown by the token manager to indicate a lexical error.
      * Parameters : EOFSeen     : indicates if EOF caused the lexicl error curLexState : lexical state in which this
      * error occured errorLine   : line number when the error occured errorColumn : column number when the error occured
      * errorAfter  : prefix that was seen before this error occured curchar     : the offending character Note: You can
      * customize the lexical error message by modifying this method.
      */
     protected static String LexicalError(boolean EOFSeen, int lexState, int errorLine, int errorColumn, String errorAfter, char curChar) {
        return("Lexical error at line " +
              errorLine + ", column " +
              errorColumn + ".  Encountered: " +
              (EOFSeen ? "<EOF> " : ("\"" + addEscapes(String.valueOf(curChar)) + "\"") + " (" + (int) curChar + "), ") +
              "after : \"" + addEscapes(errorAfter) + "\"");
     }

     /**
      * You can also modify the body of this method to customize your error messages. For example, cases like
```

```
    * LOOP_DETECTED and INVALID_LEXICAL_STATE are not of end-users concern, so you can return something like :
    * <p/>
     * "Internal Error : Please file a bug report .... "
     * <p/>
     * from this method for such cases in the release version of your parser.
     */
    public String getMessage() {
       return super.getMessage();
    }

    /*
    * Constructors of various flavors follow.
    */

    public TokenMgrError() {
    }

    public TokenMgrError(String message, int reason) {
       super(message);
       errorCode = reason;
    }

    public TokenMgrError(boolean EOFSeen, int lexState, int errorLine, int errorColumn, String errorAfter, char curChar, int reason) {
       this(LexicalError(EOFSeen, lexState, errorLine, errorColumn, errorAfter, curChar), reason);
    }
}

```