Contents

1 Basic Test Results

```
1  Logins: roigreenberg
2
3
4
5  compiling with
6      javac -cp .:/cs/course/2013/oop/lib/junit4.jar  *.java oop/ex7/main/*.java
7
8
9  tests output :
10         Perfect!
```

2 README

```
roigreenberg
yosstos

roi greenberg 305571234
Yossi Bachar 205688971
README for ex7
=====================
= File Description =
=====================
README- This file

main:
    Sjavac.java:     the manager, it runs all the program
    Parser.java:    run over the line and check their correctness.
                    create variables of some kind of line for later    checks
    compiler.java:     run right after the parser and check the correctness of the
                    remind code lines
    blocks.java:     class to save the lines of if/while/method block for later use
    SjavaException.java:     the super class for all excaptions
    DuplicateDeclarationExcaption.java: excaption in case of duplicate element
    NoSuchElementExcaption.java:  No such element excaption

arraToCompiler:
    MethodParams.java:         class for method parameters
    MethodReturn.java:         class for method returns
    VariableAsaign.java:     class for variables assignment
    MethodCall.java:         class for method calls


Variables:
    Variables.java:        the super class of the variables
    ArrayVar.java:         decorator class for array variable
    BooleanVar.java:     class for boolean variable
    CharVar.java:         class for char variable
    DoubleVar.java:     class for double variable
    IntVar.java:         class for int variable
    StringVar.java:     class string array variable
    VarFactory.java:     the factory that creates the right variables
    WrongValueTypeExcaption.java: excaption in case the value type is wrong


Methods:
    Methods.java:         the super class of the methods
    ArrayMethod.java:     class for array returning methods
    BooleanMethod.java: class for boolean returning methods
    CharMethod.java:     class for char returning methods
    DoubleMethod.java:     class for double returning methods
    IntMethod.java:     class for int returning methods
    StringMethod.java:     class for string returning methods
    VoidMethod.java:     class for not returning methods
    MethodFactory.java: the factory that creates the right method
    WrongParametersExcaption.java: excaption in case the given parameters are wrong
    WrongReturnStatementExcaption.java: excaption in case the return statement is wrong


=========
= Design=
=========

```

the design we choose to follow is very similar to ex6 design.
We use 2 classes for the main operation. Parser and Compiler.
Idealy, tha Parser would have done only the reading lines and then the Compiler
check the correctness of the action. But since the actions are not exactly linear
with the lines (some line shoould being taking care of before preview lines) we
desided that tha Parser will Also do some of the actions checks.
the Parser recieve an Array of String of code lines then read it one by one.
We create regex of every pattern of correct line the check each option. if none
beign true we return Error.
Since we first need to know every variables and methods on the Parser itself every
new variable or method is taking care of. which mean we create the variable
represent the existing var/method with the factories and add it to the ArrayList
each save it.
Also in case of methods, we run and save their parameters list.
As we mention before some thing need to be take care later so for each method/if
/while block we save their internal code line in a Block class and after finish
the outer block we run over the blocks.
the action of variable assignment and method call are saved in Arrays for later
use also because of the mentioned reason.
After finish the each Parser block it create a compiler with the Arrays we saved.
Then the manager go over the arrays and take care of each part.

for checking the validity of conditions we create a boolean variable and use it
to make sure the condition is of type of boolean.
Same way for the index of an array.(with int variable)

We also save the information about the block type so I check only the relevent
line (for example, return is only inside a method).

We use 2 Arrays for the variables, for local variables and globals, and every
function need to know the variables(such as check if assignment value is correct)
get 2 different arrays.
we did so because there are times that we need to check only one type of variables
like decleration of new variable inside a method that need to check for already
exist variable only in the local variables.
(in such case the argument we give the function is a new empty array).

The classes for the variable and method save mostly general informaition as name
and type and have the abeality to check correctness of value for the type of
variable/method.
those class are the only one who care about the type of the variable/method in
order to keep the oop principles.

for most of the var/method the correctness check is same except the type so we
implement the fanction of the super class Variables/Methods and for those how
act different(as array or void method) we override the functions.

In case of wrong code we throw excaption that being catch by the manager then
print the massage and "1".
there are several type of excaption. each excaption print(system.err) an informative
message
in case no problem found the manager will print "0".
in case of wrong file the manager will print "2".

=========================
= Answers to questions =
=========================
6.1
Error handling
As mentioned above we use excaption for error handling.
In some places we use try-catch and some time if-else for the excaption. according
to the situation and our way to "know" the excaption.
We chose to do it this way because it easy way to finish the program run since
it should stop right after an error.
also it gave as the ability of supposdly not handle an error by knowing that
the function we just call will handle it.
Also as mentioned above each excaption print(system.err) an informative
message according to it’s nature.

6.2.1
adding a new variable type is very simple.
we just need to create it proper class and add it to the factory.
in case of special behavior we might need to Override some of the super class
method. (as if short also can get int type or thing like that)

6.2.2
to support an if-else block, assuming there is no special condition of the behavior
of Sjava if-else mechanism.
In case that we need to consider ALL the if-else's as big block we just add the
method that create the if block's the proper terms to continue run over the second if
In case that we need to consider each one as all new if block we add condition for
end the block when arrive to 'else' line and start a new block.
No new class is needed for that.
Note that WE can't know all the demend for such structure so It possible we might
miss something but as far as we can think of the way we implement the programm can
handle it very easly.

6.3
We didn't really use charAt functions, and barly substring() but mostly regexs.

We use many many regex and it hard to point for the main's one but those that
reapet the most are:

```
Method name = "([a-zA-Z][\\w]*)"
And Variable name = "(_\\w+|[a-zA-Z][\\w]*)"
```

```
=========================
= Implementation issues =
=========================
```

This time, we have some issues.
I will state the main one.

We had problem in case of inner scope which allow "duplicate" global variables.
To solve that we use 2 lists of variables, local and not, so when we need to
compare existance only with the local we can do it easly.
Another small thing of that kind, was in the willing to do method that can
work on many situations, we decided to craete an empty variable list so in case
we need to use a method such createVariable but we don't care about the existing
variable we can give an empty list as parameter.

3 oop/ex7/arraysToCompiler/MethodCall.java

```java
/**
 *
 */
package oop.ex7.arraysToCompiler;

import java.util.ArrayList;

import oop.ex7.methods.Methods;

/**
 * this class used for saving code of calling to method for later compling
 * checks
 * @author roigreenberg
 */
public class MethodCall {
    public String params;
    public Methods method;
    /**
     * * the constructor - create an instance of method call
     * @param params - the method parameters
     * @param method - the method that being called
     */
    public MethodCall(String params, Methods method){
        this.method = method;
        this.params = params;

    }
}
```

4 oop/ex7/arraysToCompiler/MethodParams.java

```java
/**
 *
 */
package oop.ex7.arraysToCompiler;

import oop.ex7.methods.Methods;

/**
 * this class used for saving list of parameters of new method for later
 * compling checks
 * @author roigreenberg
 */
public class MethodParams {
    public String params;
    public Methods method;

    /**
     * the constructor - create an instance of method parameters
     * @param params - the method parameters
     * @param method - the method that being created
     */
    public MethodParams(String params, Methods method){
        this.method = method;
        this.params = params;

    }
}
```

5 oop/ex7/arraysToCompiler/MethodReturn.java

```java
/**
 *
 */
package oop.ex7.arraysToCompiler;

import oop.ex7.methods.Methods;

/**
 * this class used for saving code of method 'return' for later
 * compling checks
 * @author roigreenberg
 */
public class MethodReturn {
    public String returnValue;
    public Methods method;
    /**
     * * the constructor - create an instance of method return statement
     * @param returnValue - the value that return
     * @param method - the method
     */
    public MethodReturn(String returnValue, Methods method){
        this.method = method;
        this.returnValue = returnValue;

    }
}
```

6 oop/ex7/arraysToCompiler/VariableAsign.java

```java
/**
 *
 */
package oop.ex7.arraysToCompiler;

import oop.ex7.methods.Methods;
import oop.ex7.variables.Variables;

/**
 * this class used for saving value need to assign in variable for later
 * compling checks
 * @author roigreenberg
 */
public class VariableAsign {
    public String value;
    public Variables var;
    /**
     * * the constructor - create an instance of variable assignment
     * @param value - the value to assign
     * @param var - the variable
     */
    public VariableAsign(String value, Variables var){
        this.value = value;
        this.var = var;

    }
}
```

7 oop/ex7/main/Blocks.java

```java
/**
 *
 */
package oop.ex7.main;

import java.util.ArrayList;

import oop.ex7.methods.Methods;

/**
 * A class for blocks: if,whiles and methods.
 * save the code line inside the block for later use.
 * @author roigreenberg
 *
 */
public class Blocks {
    public Methods method;
    public String condition;
    public ArrayList<String> block;
    /**
     * the constuctor
     * @param method - the block method. null in case of if and while.
     * @throws SjavaException
     */
    public Blocks(Methods method) throws SjavaException{
        this.method = method;
        block = new ArrayList<String>();
    }

    /**
     * add code line to block array for later use.
     * @param line - internal code line
     */
    public void add(String line){
        block.add(line);
    }
}
```

8 oop/ex7/main/Compiler.java

```java
/**
 *
 */
package oop.ex7.main;

import java.util.ArrayList;

import oop.ex7.arraysToCompiler.*;
import oop.ex7.methods.Methods;
import oop.ex7.variables.Variables;

/**
 * this class complete the code check after the parser finished.
 * @author roigreenberg
 *
 */
public class Compiler {
    ArrayList<Variables> localVariables = new  ArrayList<>();
    ArrayList<Variables> variables = new  ArrayList<>();
    ArrayList<MethodCall> methodCall = new ArrayList<>();
    ArrayList<MethodParams> methodsParams = new ArrayList<>();
    ArrayList<VariableAsign> varsAsign = new ArrayList<>();
    ArrayList<MethodReturn> methodsReturn = new ArrayList<>();

    MethodReturn methodReturn;
    public boolean isMethod = false;


    /**
     * the constructor. save all the code line need to be checked later
     * each array save the code need to be check or the existing variables
     * @param variables - the outer block variables
     * @param localVariables - the inner block variables
     * @param methodsParams - the methods parameters
     * @param methodCall - call for methods
     * @param varsAsign - values to assign to variables
     * @param methodsReturn - methods 'return''s
     * @param isMethod - true iff this compile a method
     */
    public Compiler (ArrayList<Variables> variables,
                     ArrayList<Variables> localVariables,
                     ArrayList<MethodParams> methodsParams,
                     ArrayList<MethodCall> methodCall,
                     ArrayList<VariableAsign> varsAsign,
                     ArrayList<MethodReturn> methodsReturn,
                     Boolean isMethod) {

        this.isMethod = isMethod;
        this.localVariables = localVariables;
        this.variables = variables;
        this.methodCall = methodCall;
        this.methodsParams = methodsParams;
        this.varsAsign = varsAsign;
        this.methodsReturn = methodsReturn;
    }

    /**
     * method to check if method call is legal
     * @param methodCall - contain the method and call parameters
```

```
60          * @throws SjavaException
61          */
62         public void compileMethodCall(MethodCall methodCall) throws SjavaException{
63             methodCall.method.isCallLegal(methodCall.params, variables,
64                     localVariables,    Parser.methods);
65         }
66 
67         /**
68          * method to check if you can assign a variable
69          * @param varAsign - contain the variable and the assignment value
70          * @throws SjavaException
71          */
72         public void compileAsignVar(VariableAsign varAsign) throws SjavaException{
73 
74             if ((varAsign.var != null)){
75                 varAsign.var.isValueCorrect(varAsign.value, variables,localVariables,
76                         Parser.methods);
77 
78                 varAsign.var.initVar();
79             } else {
80                 throw new NoSuchElementExcaption("no such variable");
81             }
82         }
83 
84 
85         /**
86          * method to check if 'return' statement is correct
87          * @param methodReturn - contain the method and the return value
88          * @throws SjavaException
89          */
90         public void compileReturn(MethodReturn methodReturn) throws SjavaException{
91 
92                 methodReturn.method.isReturnLegal(methodReturn.returnValue, variables,
93                         localVariables,    Parser.methods);
94 
95         }
96 
97 }
```

9 oop/ex7/main/DuplicateDeclarationExcaption.java

```java
/**
 *
 */
package oop.ex7.main;

import oop.ex7.main.SjavaException;

/**
 * @author roigreenberg
 *
 */
public class DuplicateDeclarationExcaption extends SjavaException {

    /**
     * default constructor
     */
    public DuplicateDeclarationExcaption() {
        super("Element already exist");
        System.err.println("Element already exist");
    }

    /**
     * constructor
     * @param exception - messege for the exception
     */
    public DuplicateDeclarationExcaption(String exception) {
        super(exception);
        System.err.println(exception);
    }

}
```

10 oop/ex7/main/NoSuchElementExcaption.java

```java
/**
 *
 */
package oop.ex7.main;

import oop.ex7.main.SjavaException;

/**
 * @author roigreenberg
 *
 */
public class NoSuchElementExcaption extends SjavaException {

    /**
     * default constructor
     */
    public NoSuchElementExcaption() {
        super("No such element");
        System.err.println("No such element");
    }

    /**
     * constructor
     * @param exception - messege for the exception
     */
    public NoSuchElementExcaption(String exception) {
        super(exception);
        System.err.println(exception);
    }

}
```

11 oop/ex7/main/Parser.java

```java
/**
 *
 */
package oop.ex7.main;

import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.NoSuchElementException;
import java.util.Scanner;
import java.util.regex.Matcher;
import oop.ex7.arraysToCompiler.*;
import oop.ex7.methods.MethodFactory;
import oop.ex7.methods.Methods;
import oop.ex7.variables.*;


/**
 * The parser of the program.
 * Firstly, get the commands from the SJavac, which is the main program,
 * converts them to lines and start reading them with the Reader function
 * Some of the code checks is happend here and other save to the 'Compiler'
 * for later(described also in the README)
 * @author roigreenberg
 *
 */

public class Parser {
    private FileReader file = null;
    private Scanner commands;
    public ArrayList<String> lines = new ArrayList<>();
    public ArrayList<Blocks> blocks;
    private ArrayList<Variables> localVariables, variables = new  ArrayList<>();
    public static ArrayList<Methods> methods = new ArrayList<>();
    private ArrayList<MethodCall> methodsCall = new ArrayList<>();
    private ArrayList<MethodParams> methodsParams = new ArrayList<>();
    private ArrayList<VariableAsign> varsAsign = new ArrayList<>();
    private ArrayList<MethodReturn> methodsReturn = new ArrayList<>();
    private String type;
    private boolean isMethod = false;
    private Methods blockMethod;
    public static ArrayList<Variables> emptyVar = new ArrayList<>();
    /**
     * the "main" constructor - call onse at the begining
     * Read the file and convert it to array of strings
     * Also initialized all the Array will use in the class.
     * @param filePath - the path of the Sjava file
     * @throws IOException - in case of wrong file
     */
    public Parser(String filePath) throws IOException {
        methods = new ArrayList<>();
        blocks = new ArrayList<Blocks>();

        localVariables = new ArrayList<>();
        Sjavac.compilers = new ArrayList<>();
        try {
            file = new FileReader(filePath);
```

```
60          } catch (FileNotFoundException e) {
61              throw new IOException();
62          }
63          Scanner scanner = new Scanner (file);
64          commands = scanner.useDelimiter("\\s*\n\\s*");
65          while (commands.hasNext()) {
66
67              lines.add(commands.next());
68          }
69          commands.close();
70          scanner.close();
71      }
72      /**
73       * the "blocks" constructor - call for any inner block
74       * in case of method block, add the method parameters to localVariables
75       * Also initialezed new Array for inner blocks
76       * @param variables - the outer block variables
77       * @param localVariables - the inner block variables
78       * @param method - the block method. null in case of if/while block
79       */
80      public Parser(ArrayList<Variables> localVariables,
81              ArrayList<Variables> variables, Methods method) {
82
83          this.variables = variables;
84          this.blockMethod = method;
85          isMethod = (blockMethod != null) ;
86          this.localVariables = localVariables;
87          if (isMethod){
88              this.localVariables.addAll(blockMethod.paramList);
89          }
90          blocks = new ArrayList<Blocks>();
91      }
92
93      /**
94       * Knows how to read correctly the given codelines.
95       * It knows to distinguish between every variable, every array
96       * and every method.
97       * Uses the programs inside the Variables and Methods packages
98       * to determine where every line belong and if it is correct.
99       * Also distinguishes between every block and blocks inside blocks
100      * THe parser is working mostly like "switch-case" but since it use
101      * boolean(.matches()) we did it with "if-else" structure.
102      * if none of the if get true, that mean there a wrong line.
103      * After the check create an instanse of 'compiler' for later use then run
104      * on internal block recursivly
105      *
106      * Note: we know the parser maybe too long and we tried to seperet it
107      * (that also way we did the 'Compiler' and 'Regex' classes and move some
108      * methods to 'Variables' and 'Methods' (like .isCorrect()) but since there
109      * a lot of variables any time more sepereting were become vary ugly (as
110      * can be seen at the 'compiler' creation which need 7(!) parameters)
111      *
112      * @param codelines - the given code lines
113      * @throws SjavaException
114      *
115      */
116     public void Reader(ArrayList<String> codelines)
117             throws SjavaException{
118         Iterator<String> linesIter = codelines.iterator();
119         String line;
120
121         while (linesIter.hasNext()){
122             line = linesIter.next();
123
124             if (line.matches(Regex.METHOD) && !(isMethod)){
125                 Matcher match = Regex.pattMethod.matcher(line);
126                 match.find();
127                 if (match.group(2) != null)
```

```
128                     type = match.group(2);
129                 else
130                     type = match.group(1);
131
132                 Methods method = MethodFactory.createMethod(type,
133                         match.group(4), !(match.group(3) == null));
134
135                 this.block(linesIter, line, method);
136
137                 methods.add(method);
138
139                 Matcher matchVar = Regex.pattMultiParam.matcher(match.group(5));
140                 Variables var;
141                 while (matchVar.find()){
142                     try {
143                         var = VarFactory.createVariable(matchVar.group(2),
144                                 matchVar.group(4), !(matchVar.group(3) == null),
145                                 method.paramList, emptyVar);
146                         var.initVar();
147                         method.paramList.add(var);
148                     } catch (DuplicateDeclarationExcaption e) {
149                         throw new DuplicateDeclarationExcaption("can't duplicate"
150                                 + " parameters name");
151                     }
152                 }
153
154             } else if (line.matches(Regex.METHOD_CALL)){
155                 Matcher match = Regex.pattMethodCall.matcher(line);
156                 match.find();
157                 Methods method = Methods.isMethodExists(methods, match.group(1));
158                 if (method == null)
159                     throw new NoSuchElementExcaption("no such method");
160                 MethodCall call = new MethodCall(match.group(3), method);
161                 methodsCall.add(call);
162
163             } else if (line.matches(Regex.IF+"|"+Regex.WHILE)){
164
165                 Matcher match = Regex.pattBoolean.matcher(line);
166                 match.find();
167
168                 this.block(linesIter, line, blockMethod);
169                 Matcher matchCondition = Regex.pattBoolean.matcher(line);
170                 matchCondition.find();
171                 Variables condition = new BooleanVar("condition", "boolean");
172
173                 VariableAsign conditionValue =
174                         new VariableAsign(matchCondition.group(1), condition);
175                 varsAsign.add(conditionValue);
176
177             } else if (line.matches(Regex.INIT_VAR)){
178
179                 Matcher match = Regex.pattInitVar.matcher(line);
180                 match.find();
181                 Variables var;
182                 try {
183                     var = VarFactory.createVariable(match.group(1),
184                             match.group(2),false, localVariables, emptyVar);
185                     if (line.contains("=")){
186                         VariableAsign varAsign =
187                                 new VariableAsign(match.group(4), var);
188
189                         varsAsign .add(varAsign);
190
191                     }
192                     localVariables.add(var);
193                 } catch (DuplicateDeclarationExcaption e) {
194                     throw new DuplicateDeclarationExcaption("Variable already exist");
195                 }
```

-5/-5 Your code is very hard to read and understand. (code='readability_problem') when a method arrive to such size that mean you need to split to helper method!! very difficult to understand your code

```
196
197             } else if (line.matches(Regex.ASIGN_VAR)){
198                 Matcher match = Regex.pattAsignVar.matcher(line);
199                 match.find();
200                 Variables var = Variables.isVarExists(localVariables, variables,
201                         match.group(1));
202                 if (var == null)
203                     throw new NoSuchElementExcaption("no such variable");
204                 if (var.isArray)
205                     throw new SjavaException("can't assign an array");
206                 VariableAsign varAsign = new VariableAsign(match.group(2), var);
207
208                 varsAsign.add(varAsign);
209
210             } else if (line.matches(Regex.INIT_ARR)){
211                 Matcher match = Regex.pattInitArr.matcher(line);
212                 match.find();
213                 Variables var;
214                 try {
215                     var = VarFactory.createVariable(match.group(1),
216                             match.group(2),true, localVariables, variables);
217                     if (line.contains("=")){
218                         String values = match.group(4);
219                         if (values != null) {
220                             Matcher matchVar = Regex.pattMultiVar.matcher(values);
221                             while (matchVar.find()){
222                                 VariableAsign varAsign =
223                                     new VariableAsign(matchVar.group(1), var);
224                                 varsAsign.add(varAsign);
225                             }
226                         }
227                     }
228                     localVariables.add(var);
229                 } catch (SjavaException e) {
230                     throw new SjavaException();
231                 }
232
233             } else if (line.matches(Regex.ASIGN_ARR)){
234                 Matcher match = Regex.pattAsignArr.matcher(line);
235                 match.find();
236                 Variables var = Variables.isVarExists(localVariables, variables,
237                         match.group(1));
238                 if (match.group(2).matches("-\\s*\\d+\\s*"))
239                     throw new SjavaException("negetive index");
240                 VariableAsign indexAsign = new VariableAsign(match.group(2),
241                         new IntVar("index", "int"));
242                 VariableAsign varAsign = new VariableAsign(match.group(3), var);
243                 varsAsign.add(indexAsign);
244                 varsAsign.add(varAsign);
245
246             } else if (isMethod    && line.matches(Regex.RETURN)){
247                     Matcher match = Regex.pattReturn.matcher(line);
248                     match.find();
249                     MethodReturn methodReturn = new MethodReturn(match.group(2),
250                             blockMethod);
251                     methodsReturn.add(methodReturn);
252
253             } else if (!line.matches(Regex.IGNORE)){
254
255                 throw new SjavaException();
256             }
257
258         }
259
260         Compiler compiler = new Compiler(variables,
261                 localVariables,
262                 methodsParams,
263                 methodsCall,
```

```
                varsAsign,
                methodsReturn,
                isMethod);

        Sjavac.compilers.add(compiler);

        localVariables.addAll(variables);

        for (Blocks block: blocks){

            Parser b = new Parser(new ArrayList<Variables>(),localVariables,
                    block.method);
            b.Reader(block.block);
        }

    }
     /**
     * Method for creating instance of blocks.
     * save the code line of inner block
     * checks if it fits the correct rules for "{" and "}"
     * also, if each block is opened and closes properly,
     * not too much "{" and "}"s and if they are placed
     * correctly.
     * @param linesIter - the lines iterator
     * @param line - the current line
     * @param method - the block method. null in case of if and while.
     * @throws SjavaException
     */
    public void block(Iterator<String> linesIter, String line, Methods method)
            throws SjavaException{

        Blocks block = new Blocks(method);

        int counter = 1;
        if (line.matches(Regex.IF+"|"+Regex.WHILE)){
            block.condition = line;
        }
        while (counter != 0){
            try {
            line = linesIter.next();
            }catch (NoSuchElementException e) {
                throw new SjavaException();
            }
            block.block.add(line);
            if (line.matches(Regex.IF+"|"+Regex.WHILE)){

                counter += 1 ;
            }
            if (line.matches("\\s*\\}\\s*"))
                counter -= 1;
        }

        blocks.add(block);
    }

}
```

12 oop/ex7/main/Regex.java

```java
/**
 *
 */
package oop.ex7.main;

import java.util.regex.Pattern;

/**
 * this class holds All the regular exprations(regex) and pattern use in the program
 * @author roigreenberg
 *
 */
public class Regex {
    public static final String IGNORE = "//.*|\\s*|\\}.*";
    public static final String IF = "\\s*if\\s*\\(\\s*(.+)\\s*\\)\\s*\\{\\s*";
    public static final String WHILE = "\\s*while\\s*\\(\\s*(.+)"
            + "\\s*\\)\\s*\\{\\s*";
    public static final String METHOD_NAME = "([a-zA-Z][\\w]*)";
    public static final String TYPE = "(int|double|String|char|boolean)";
    public static final String VAR_NAME = "(_\\w+|[a-zA-Z][\\w]*)";
    public static final String METHOD_PARAM = "(\\s*"+TYPE+"(\\s*\\[\\])?\\s*"
                                        +VAR_NAME+")";
    public static final String METHOD = "\\s*(void|"+TYPE + "(\\[\\])?)\\s+" +
            METHOD_NAME + "\\s*(\\(("+METHOD_PARAM+"(,"+METHOD_PARAM+
            ")*)?\\))\\s*\\{\\s*";
    public static final String METHOD_CALL = "\\s*"+METHOD_NAME+"\\s*\\(\\s*"
            + "(([^,]+(,[^,]+)*?)?\\s*\\))\\s*;\\s*";
    public static final String INIT_VAR = "\\s*" + TYPE + "\\s+" +
            VAR_NAME + "(\\s*=\\s*" + "(.+)" + "\\s*)?;\\s*";
    public static final String ASIGN_VAR = "\\s*" + VAR_NAME +
            "\\s*=\\s*" + "(.+)" + "\\s*;\\s*";
    public static final String ARR_VALUE = "\\s*\\{\\s*([^,]+(,[^,]+)*?)?\\s*\\}\\s*";
    public static final String INIT_ARR = "\\s*" + TYPE + "\\s*\\[\\s*\\]\\s*" +
            VAR_NAME + "\\s*(="+ARR_VALUE+")?;\\s*";
    public static final String ASIGN_ARR = "\\s*" + VAR_NAME +
            "\\s*\\[(.+)\\]\\s*=\\s*" + "(.+)" + "\\s*;";
    public static final String BLOCK_END = "\\}";
    public static final String RETURN = "\\s*return\\s*(\\s+(.+)\\s*)?;\\s*";
    public static final String ILEGAL = "\\s*(true|false|if|while)|"+Regex.TYPE+"\\s*";
    public static final String INT = "-?[\\d]+";
    public static final String DOUBLE = "-?([\\d]+(\\.[\\d]+)?)";
    public static final String STRING = "\\\".*\\\"";
    public static final String CHAR = "\\'.\\'";
    public static final String BOOLEAN = "(true|false)";
    public static final String VALUE = "("+DOUBLE+"|"+INT+"|"+STRING+"|"
                +CHAR+"|"+BOOLEAN+")(\\[\\])?";

    public static final String OPERATOR = "\\s*-?([^\\+\\-\\*/]+)\\s*"
            + "([\\+\\-\\*/]\\s*(.+))?\\s*";
    public static Pattern pattOperator = Pattern.compile(OPERATOR);
    public static Pattern pattInitVar = Pattern.compile(INIT_VAR);
    public static Pattern pattAsignVar = Pattern.compile(ASIGN_VAR);
    public static Pattern pattInitArr = Pattern.compile(INIT_ARR);
    public static Pattern pattMethod = Pattern.compile(METHOD);
    public static Pattern pattMethodCall = Pattern.compile(METHOD_CALL);
    public static Pattern pattAsignArr = Pattern.compile(ASIGN_ARR);
    public static Pattern pattMultiParam = Pattern.compile(METHOD_PARAM);
    public static Pattern pattReturn = Pattern.compile(RETURN);
    public static Pattern pattBoolean = Pattern.compile("\\(\\s*(true|false|.+)\\s*\\)");
```

```
60      public static Pattern pattMultiVar = Pattern.compile("([^,]*[^,])");
61      public static Pattern pattValue = Pattern.compile("=.+;");
62      public static Pattern pattArrValue = Pattern.compile(ARR_VALUE);
63  }
```

13 oop/ex7/main/SjavaException.java

```java
package oop.ex7.main;

/**
 * Execption
 * @author roigreenberg
 *
 */
public class SjavaException extends Exception {
    /**
     * default constructor
     */
    public SjavaException() {
        super("wrong s-java code");
        System.err.println("wrong s-java code");
    }
    /**
     * constructor
     * @param exception  - messege for the exception
     */
    public SjavaException(String exception) {
        super(exception);
        System.err.println(exception);
    }
}
```

14 oop/ex7/main/Sjavac.java

```java
/**
 *
 */
package oop.ex7.main;

import java.io.IOException;
import java.util.ArrayList;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import javax.management.monitor.CounterMonitorMBean;

import oop.ex7.arraysToCompiler.MethodCall;
import oop.ex7.arraysToCompiler.MethodReturn;
import oop.ex7.arraysToCompiler.VariableAsign;

/**
 * The main program. The manager
 * call the parser firstly, for reading the file and initilized it.
 * then call the compiler to finish the compiling check.
 * @author roigreenbeg
 *
 */
public class Sjavac {

    public static ArrayList<Compiler> compilers = new ArrayList<>();
    /**
     * The manager.
     * call the parser firstly, for reading the file and initilized it.
     * then call the compiler to finish the compiling check.
     * Print "2" in case of wrong file, "1" in case of comiliation error
     * or "0" if the code is correct.
     * @param args - the path to the Sjava file
     * @throws IOException
     */
    public static void main(String[] args) throws IOException {
        try {
            Parser p = new Parser(args[0]);
            try {
                p.Reader(p.lines);
                for (Compiler compiler: compilers){

                    for (VariableAsign var: compiler.varsAsign)
                        compiler.compileAsignVar(var);

                    for (MethodCall method: compiler.methodCall)
                        compiler.compileMethodCall(method);

                    for (MethodReturn method: compiler.methodsReturn){
                        compiler.compileReturn(method);
                    }

                }
                System.out.println("0");
            } catch (SjavaException e) {
                System.out.println("1");
            }
        } catch (IOException e) {
            System.out.println("2");
```

```
60              }
61          }
62      }
```

15 oop/ex7/methods/ArrayMethod.java

```java
/**
 *
 */
package oop.ex7.methods;

import java.lang.reflect.Method;
import java.util.ArrayList;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import oop.ex7.main.DuplicateDeclarationExcaption;
import oop.ex7.main.Parser;
import oop.ex7.main.Regex;
import oop.ex7.main.SjavaException;
import oop.ex7.variables.VarFactory;
import oop.ex7.variables.Variables;

/**
 * The class for the method that return arrays
 * Override some of the method to act for the array type (like int for int[])
 *
 * @author roigreenberg
 *
 */
public class ArrayMethod extends Methods {

    /**
     * the constructor
     * change returnType to the array type as mention above
     * @param type - method returning type
     * @param name - method name
     * @throws DuplicateDeclarationExcaption - in case variable already exist
     */
    public ArrayMethod(String type, String name)
            throws DuplicateDeclarationExcaption {

        super(name, "array");
        returnType = VarFactory.createVariable(type, "returnType", true,
                Parser.emptyVar, Parser.emptyVar);
    }

    /**
     * Override the method so it will check return of array and of instance
     *  of the array
     * @see oop.ex7.methods.Methods#isReturnLegal(java.lang.String,
     *     java.util.ArrayList, java.util.ArrayList, java.util.ArrayList)
     */
    @Override
    public void isReturnLegal(String returnValue, ArrayList<Variables> variables,
            ArrayList<Variables> localVariables ,ArrayList<Methods> methods)
                    throws SjavaException {
        if (returnValue != null && returnValue.matches(Regex.ARR_VALUE)){

            Matcher matchReturn = Regex.pattArrValue.matcher(returnValue);
            matchReturn.find();
            if (matchReturn.group(1) != null){
                Matcher match = Regex.pattMultiVar.matcher(matchReturn.group(1));
                while (match.find()) {
                    if (!this.returnType.isValueCorrect(match.group(1),
```

```
60                              variables, localVariables, methods)){
61                          throw new SjavaException();
62                      }
63                  }
64              }
65          } else {
66              super.isReturnLegal(returnValue, variables, localVariables, methods);
67
68          }
69
70      }
71
72  }
```

16 oop/ex7/methods/BooleanMethod.java

```java
package oop.ex7.methods;

import oop.ex7.main.DuplicateDeclarationExcaption;


/**
 * class of instance represent method with boolean type retrun
 * @author roigreenberg
 *
 */
public class BooleanMethod extends Methods {


    /**
     * the constructor
     * @param type - method returning type
     * @param name - method name
     * @throws DuplicateDeclarationExcaption - in case variable already exist
     */
    public BooleanMethod(String type, String name)
            throws DuplicateDeclarationExcaption {
        super(name,type);
    }



}
```

17 oop/ex7/methods/CharMethod.java

```java
/**
 *
 */
package oop.ex7.methods;

import oop.ex7.main.DuplicateDeclarationExcaption;
import oop.ex7.variables.VarFactory;

/**
 * class of instance represent method with chat=r type retrun
 * @author roigreenberg
 *
 */
public class CharMethod extends Methods{
    /**
     * the constructor
     * @param type - method returning type
     * @param name - method name
     * @throws DuplicateDeclarationExcaption - in case variable already exist
     */
    public CharMethod(String type, String name)
            throws DuplicateDeclarationExcaption {
        super(name,type);
    }

}
```

18 oop/ex7/methods/DoubleMethod.java

```java
/**
 *
 */
package oop.ex7.methods;

import oop.ex7.main.DuplicateDeclarationExcaption;
import oop.ex7.variables.VarFactory;

/**
 * class of instance represent method with double type retrun
 * @author roigreenberg
 *
 */
public class DoubleMethod extends Methods{
    /**
     * the constructor
     * @param type - method returning type
     * @param name - method name
     * @throws DuplicateDeclarationExcaption - in case variable already exist
     */
    public DoubleMethod(String type, String name)
            throws DuplicateDeclarationExcaption {
        super(name,type);
    }

}
```

19 oop/ex7/methods/IntMethod.java

```java
/**
 *
 */
package oop.ex7.methods;

import oop.ex7.main.DuplicateDeclarationExcaption;
import oop.ex7.variables.VarFactory;

/**
 * class of instance represent method with int type retrun
 * @author roigreenberg
 *
 */
public class IntMethod extends Methods{
    /**
     * the constructor
     * @param type - method returning type
     * @param name - method name
     * @throws DuplicateDeclarationExcaption- in case variable already exist
     */
    public IntMethod(String type, String name)
            throws DuplicateDeclarationExcaption {
        super(name,type);
    }
}
```

20 oop/ex7/methods/MethodFactory.java

```java
/**
 *
 */
package oop.ex7.methods;

import oop.ex7.main.DuplicateDeclarationExcaption;
import oop.ex7.main.Parser;
import oop.ex7.variables.Variables;


/**
 * This class is the factory that creates all types of methods.
 * called in the parser to make sure we're adding the right type of method.
 * @author roigreenberg
 */
public class MethodFactory {
    private static Methods method;


    /**
     * this method create method instance according to the given parameters
     * @param methodType - the method returning type
     * @param methodName - the method name
     * @param isArray - true iff the method returning an array
     * @return method - the created method.
     * @throws DuplicateDeclarationExcaption  - in case method already exist
     */
    public static Methods createMethod(String methodType, String methodName,
            boolean isArray) throws DuplicateDeclarationExcaption{

        if (Methods.isMethodExists(Parser.methods, methodName)!=null)
            throw new DuplicateDeclarationExcaption("Method already"
                    + " exist");

        switch (methodType){
        case ("int"): {
            method = new IntMethod(methodType ,methodName);
            break;
        }
        case ("double"): {
            method = new DoubleMethod(methodType ,methodName);
            break;
        }
        case ("String"): {
            method = new StringMethod(methodType ,methodName);
            break;
        }
        case ("boolean"): {
            method = new BooleanMethod(methodType ,methodName);
            break;
        }
        case ("char"): {
            method = new CharMethod(methodType ,methodName);
            break;
        }
        case ("void"): {
        method = new VoidMethod(methodType, methodName);
        break;
    }
```

```
60          }
61          if (isArray){
62              return new ArrayMethod(methodType, methodName);
63          } else {
64              return method;
65          }
66      }
67  }
```

21 oop/ex7/methods/Methods.java

```java
/**
 *
 */
package oop.ex7.methods;

import java.util.ArrayList;
import java.util.regex.Matcher;

import oop.ex7.main.DuplicateDeclarationExcaption;
import oop.ex7.main.Parser;
import oop.ex7.main.Regex;
import oop.ex7.main.SjavaException;
import oop.ex7.variables.VarFactory;
import oop.ex7.variables.Variables;

/**
 * Abstract class for all Methods
 * Contain some method to operate on the method instance
 * @author roigreenberg
 *
 */
public abstract class Methods {
    public ArrayList<Variables> paramList = new ArrayList<Variables>();
    public Variables returnType;
    protected String name;
    protected String type;

    /**
     * the constructor - create instance of Method
     * @param name - the method name
     * @param type - the method returning type
     * @throws DuplicateDeclarationExcaption - in case variable already exist
     */
    public Methods(String name, String type) throws DuplicateDeclarationExcaption{
        this.name = name;
        this.type = type;
        returnType = VarFactory.createVariable(type, "returnType", false,
                Parser.emptyVar, Parser.emptyVar);
    }

    /**
     * returns true iff the given name equal to the method name
     * @param name - name of method
     * @return true iff the given name equal to the method name
     */
    public boolean isEqual(String name) {

        return this.name.equals(name);
    }


    /**
     * checks if the call is legal according to sjava rules
     * @param values - parameters values
     * @param variables - outer scope variables list
     * @param localVariables - local variables list
     * @param methods - the existing method list
     * @throws SjavaException
     */
```

```
60      public void isCallLegal(String values, ArrayList<Variables> variables,
61              ArrayList<Variables> localVariables ,ArrayList<Methods> methods)
62                      throws SjavaException {
63          if (values != null) {
64              Matcher matchVar = Regex.pattMultiVar.matcher(values);
65              for (Variables var: paramList){
66                  try {
67                      matchVar.find();
68                      var.isValueCorrect(values.substring(matchVar.start(),
69                          matchVar.end()), variables, localVariables, methods);
70                  } catch (SjavaException e) {
71                      throw new WrongParametersExcaption();
72                  }
73              }
74              if (matchVar.find()){
75                  throw new WrongParametersExcaption();
76              }
77          } else if (!paramList.isEmpty())
78              throw new WrongParametersExcaption();
79      }
80
81      /**
82       * checks if the return statement is legal
83       * @param returnValue - the return statement
84       * @param variables - outer scope variables list
85       * @param localVariables - local variables list
86       * @param methods - the existing method list
87       * @throws SjavaException
88       */
89      public void isReturnLegal(String returnValue, ArrayList<Variables> variables,
90              ArrayList<Variables> localVariables ,ArrayList<Methods> methods)
91                      throws SjavaException {
92          if (returnValue != null){
93
94              try {
95                  this.returnType.isValueCorrect(returnValue, variables,
96                          localVariables, methods);
97              } catch (SjavaException e){
98                  throw new WrongReturnStatementExcaption();
99              }
100         } else {
101             throw new WrongReturnStatementExcaption();
102         }
103
104     }
105
106     /**
107      * check if there a method with the given name
108      * @param methods - the existing method list
109      * @param methodName - method name
110      * @return the method with the given name or null if not exist
111      */
112     public static Methods isMethodExists (ArrayList<Methods> methods, String methodName){
113         for (Methods method:methods){
114             if (method.isEqual(methodName)){
115                 return method;
116             }
117         }
118         return null;
119     }
120
121 }
```

22 oop/ex7/methods/StringMethod.java

```java
package oop.ex7.methods;

import oop.ex7.main.DuplicateDeclarationExcaption;
import oop.ex7.variables.VarFactory;

/**
 * class of instance represent method with string type retrun
 * @author roigreenberg
 *
 */
public class StringMethod extends Methods{
    /**
     * the constructor
     * @param type - method returning type
     * @param name - method name
     * @throws DuplicateDeclarationExcaption - in case variable already exist
     */
    public StringMethod(String type, String name)
            throws DuplicateDeclarationExcaption {
        super(name,type);
    }

}
```

23 oop/ex7/methods/VoidMethod.java

```java
/**
 *
 */
package oop.ex7.methods;

import java.util.ArrayList;

import oop.ex7.main.DuplicateDeclarationExcaption;
import oop.ex7.main.SjavaException;
import oop.ex7.variables.Variables;

/**
 * @author roigreenberg
 *
 */
public class VoidMethod extends Methods {

    /**
     * the constructor
     * change the returnType to null.
     * @param type - method returning type
     * @param name - method name
     * @throws DuplicateDeclarationExcaption - in case variable already exist
     */
    public VoidMethod(String type, String name)
            throws DuplicateDeclarationExcaption {
        super(name,type);
        returnType = null;
    }
    /**
     * checks if the return statement is legal
     * oeride the method to check if the return is null
     * @see oop.ex7.methods.Methods#isReturnLegal(java.lang.String,
     *  java.util.ArrayList, java.util.ArrayList, java.util.ArrayList)
     */
    @Override
    public void isReturnLegal(String returnValue, ArrayList<Variables> variables,
            ArrayList<Variables> localVariables, ArrayList<Methods> methods)
                    throws SjavaException{
        if (returnValue != null)
            throw new WrongReturnStatementExcaption();

    }

}
```

24 oop/ex7/methods/WrongParametersExcaption.java

```java
/**
 *
 */
package oop.ex7.methods;

import oop.ex7.main.SjavaException;

/**
 * @author roigreenberg
 *
 */
public class WrongParametersExcaption extends SjavaException {

    /**
     *
     */
    public WrongParametersExcaption() {
        super("wrong parameters for method");
        System.err.println("wrong parameters for method");
    }

    /**
     * constructor
     * @param exception  - messege for the exception
     */
    public WrongParametersExcaption(String exception) {
        super(exception);
        // TODO Auto-generated constructor stub
    }

}
```

25 oop/ex7/methods/WrongReturnStatementExcaptio

```java
/**
 *
 */
package oop.ex7.methods;

import oop.ex7.main.SjavaException;

/**
 * @author roigreenberg
 *
 */
public class WrongReturnStatementExcaption extends SjavaException {

    /**
     * default constructor
     */
    public WrongReturnStatementExcaption() {
        super("wrong return statment for method");
        System.err.println("wrong return statment for method");
    }

    /**
     * constructor
     * @param exception  - messege for the exception
     */
    public WrongReturnStatementExcaption(String exception) {
        super(exception);
        System.err.println(exception);
    }

}
```

26 oop/ex7/variables/ArrayVar.java

```java
/**
 *
 */
package oop.ex7.variables;

import java.util.ArrayList;

import oop.ex7.main.SjavaException;
import oop.ex7.methods.Methods;

/**
 * class of instance represent array.
 * this class use as decorator of the array instance
 * @author roigreenberg
 *
 */
public class ArrayVar extends Variables {
    private Variables valueVar;
    /**
     * the constructor
     * set the constructor with the values of the array type
     * change 'isArray' to be true and initialized the variable
     * @param valueVar - variable of the type of the array
     */
    public ArrayVar(Variables valueVar){
        super (valueVar.name, valueVar.type, valueVar.value);
        this.isArray = true;
        this.valueVar = valueVar;
        this.initVar();
    }


    /**
     * override the method to so it will check the correctness agains variable type
     * @see oop.ex7.variables.Variables#isValueCorrect(java.lang.String,
     * java.util.ArrayList, java.util.ArrayList, java.util.ArrayList)
     */
    public boolean isValueCorrect(String data, ArrayList<Variables> variables
            ,ArrayList<Variables> localVariables, ArrayList<Methods> methods)
                    throws SjavaException {
        return valueVar.isValueCorrect(data, variables,
                localVariables, methods);
    }


}
```

27 oop/ex7/variables/BooleanVar.java

```java
/**
 *
 */
package oop.ex7.variables;

import java.util.ArrayList;

import oop.ex7.main.Regex;
import oop.ex7.methods.Methods;

/**
 * class of instance represent variable of boolean type
 * @author roigreenberg
 *
 */
public class BooleanVar extends Variables {
    private static String value = "\\s*"+Regex.BOOLEAN+"s*";
    /**
     * the constructor
     * @param type - variable type
     * @param name - variable name
     */
    public BooleanVar(String name, String type){
        super(name, type, value);
    }
}
```

28 oop/ex7/variables/CharVar.java

```java
/**
 *
 */
package oop.ex7.variables;

import oop.ex7.main.Regex;

/**
 * class of instance represent variable of char type
 * @author roigreenberg
 *
 */
public class CharVar extends Variables {
    private static String value = "\\s*"+Regex.CHAR+"s*";
    /**
     * the constructor
     * @param type - variable type
     * @param name - variable name
     */
    public CharVar(String name, String type){
        super(name, type, value);
    }

}
```

29 oop/ex7/variables/DoubleVar.java

```java
/**
 *
 */
package oop.ex7.variables;

import java.util.ArrayList;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import oop.ex7.main.Regex;
import oop.ex7.main.SjavaException;
import oop.ex7.methods.Methods;

/**
 * class of instance represent variable of double type
 * @author roigreenberg
 *
 */
public class DoubleVar extends Variables {

    private static String value = "\\s*"+Regex.DOUBLE+"s*";
    /**
     * the constructor
     * @param type - variable type
     * @param name - variable name
     */
    public DoubleVar(String name, String type){
        super(name, type, value);
    }

    /**
     * override the method to extend it to support operations
     * @see oop.ex7.variables.Variables#isValueCorrect(java.lang.String,
     *  java.util.ArrayList, java.util.ArrayList, java.util.ArrayList)
     */
    @Override
    public boolean isValueCorrect(String data, ArrayList<Variables> variables
            ,ArrayList<Variables> localVariables, ArrayList<Methods> methods)
                    throws SjavaException {
        Matcher match = Regex.pattOperator.matcher(data);
        match.find();
        super.isValueCorrect(match.group(1), variables,
                localVariables, methods);
        if (match.group(3) != null)
            super.isValueCorrect(match.group(3), variables,
                    localVariables, methods);

        return true;
    }


    /**
     * override the method to extend it to support also int type
     * @see oop.ex7.variables.Variables#isCorrect(oop.ex7.variables.Variables)
     */
    public boolean isCorrect(Variables var){
        return var != null && var.initialized &&
                (this.type.equals(var.type) || "int".equals(var.type));
    }
```

```
60      /**
61       * override the method to extend it to support also int type
62       * @see oop.ex7.variables.Variables#isCorrect(oop.ex7.methods.Methods)
63       */
64      public boolean isCorrect(Methods method) throws SjavaException{
65          if (method.returnType == null)
66              throw new WrongValueTypeExcaption();
67          return (this.type.equals(method.returnType.type) ||
68                  "int".equals(method.returnType.type));
69      }
70
71
72  }
```

30 oop/ex7/variables/IntVar.java

```java
/**
 *
 */
package oop.ex7.variables;

import java.util.ArrayList;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import oop.ex7.main.Regex;
import oop.ex7.main.SjavaException;
import oop.ex7.methods.Methods;

/**
 * class of instance represent variable of int type
 * @author roigreenberg
 *
 */
public class IntVar extends Variables {

    private static String value = "\\s*-?"+Regex.INT+"\\s*";
    /**
     * the constructor
     * @param type - variable type
     * @param name - variable name
     */
    public IntVar(String name, String type){
        super(name, type, value);
    }


    /**
     * override the method to extend it to support operations
     * @see oop.ex7.variables.Variables#isValueCorrect(java.lang.String,
     *  java.util.ArrayList, java.util.ArrayList, java.util.ArrayList)
     */
    @Override
    public boolean isValueCorrect(String data, ArrayList<Variables> variables
            ,ArrayList<Variables> localVariables, ArrayList<Methods> methods)
                    throws SjavaException {
        if (data.matches(Regex.OPERATOR)){
            Matcher match = Regex.pattOperator.matcher(data);
            match.find();
            super.isValueCorrect(match.group(1), variables,
                    localVariables, methods);
            if (match.group(3) != null)
                super.isValueCorrect(match.group(3), variables,
                        localVariables, methods);
        } else {
            super.isValueCorrect(data, variables,
                    localVariables, methods);
        }
        return true;
    }

}
```

31 oop/ex7/variables/StringVar.java

```java
/**
 *
 */
package oop.ex7.variables;

import oop.ex7.main.Regex;

/**
 * class of instance represent variable of string type
 * @author roigreenberg
 *
 */
public class StringVar extends Variables {

    private static String value = "\\s*"+Regex.STRING+"s*";
    /**
     * the constructor
     * @param type - variable type
     * @param name - variable name
     */
    public StringVar(String name, String type){
        super(name, type, value);
    }

}
```

32 oop/ex7/variables/VarFactory.java

```java
/**
 *
 */
package oop.ex7.variables;

import java.util.ArrayList;

import oop.ex7.main.DuplicateDeclarationExcaption;
import oop.ex7.main.Parser;
import oop.ex7.methods.Methods;

/**
 * This class is the factory that creates all types of variables.
 * called in the parser to make sure we're adding the right type of variables.
 * @author roigreenberg
 */
public class VarFactory {
    private static Variables var;
    /**
     * this method create variable instance according to the given parameters
     * @param varType - the variable returning type
     * @param varName - the variable name
     * @param isArray - true iff the variable is an array
     * @param localVariables
     * @param variables
     * @return variable - the created variable.
     * @throws DuplicateDeclarationExcaption - in case variable already exist
     */
    public static Variables createVariable(String varType, String varName,
            boolean isArray, ArrayList<Variables> localVariables,
            ArrayList<Variables> variables)
                    throws DuplicateDeclarationExcaption{

        if (Variables.isVarExists(localVariables, variables, varName)!=null)
            throw new DuplicateDeclarationExcaption("variable already"
                    + " exist");

        switch (varType){
        case ("int"): {
            var = new IntVar(varName, varType);
            break;
        }
        case ("double"): {
            var = new DoubleVar(varName, varType);
            break;
        }
        case ("String"): {
            var = new StringVar(varName, varType);
            break;
        }
        case ("boolean"): {
            var = new BooleanVar(varName, varType);
            break;
        }
        case ("char"): {
            var = new CharVar(varName, varType);
            break;
        }
        }
```

```
60            if (isArray)
61                return new ArrayVar(var);
62            else {
63                return var;
64            }
65        }
66    }
```

33 oop/ex7/variables/Variables.java

```java
/**
 *
 */
package oop.ex7.variables;

import java.util.ArrayList;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import javax.security.sasl.SaslException;

import org.hamcrest.core.IsSame;

import oop.ex7.main.Regex;
import oop.ex7.main.SjavaException;
import oop.ex7.methods.Methods;

/**
 * Abstract class for all Variable
 * Contain some method to operate on the variable instance
 * @author roigreenberg
 *
 */
public abstract class Variables implements Cloneable{

    public String name;
    public String type;
    protected String value;
    public boolean isArray = false;
    public boolean initialized = false;
    private static final String METHOD_NAME = Regex.METHOD_NAME+"\\s*\\(\\s*(.*)\\s*\\)\\s*";
    private static final String VAR_NAME = Regex.VAR_NAME+"\\s*(\\[\\s*(.+)\\s*\\])?";
    Pattern pattVarName = Pattern.compile(VAR_NAME);
    Pattern pattMethodName = Pattern.compile(METHOD_NAME);

    /**
     * the constructor - create instance of Variable
     * @param name - the variable name
     * @param type - the variable type
     * @param value - the regex pattern of the variable value
     */
    public Variables(String name, String type, String value){
        this.name = name;
        this.type = type;
        this.value = value;
    }
    /**
     * changes the initialized feature of the variable to true
     */
    public void initVar(){
        initialized = true;
    }


    /**
     * check if the given data that ask to be assign to the variable is of
     * correct type.
     * It check if the data is of type of method call, ather variable or value
     * and call the right method to check the correctness of the data.
```

```
60          * @param data - the value to check
61          * @param variables - outer scope variables list
62          * @param localVariables - local variables list
63          * @param methods - the existing method list
64          * @return true iff the data is correct
65          * @throws SjavaException - in case of Sjava compilation error
66          */
67         public boolean isValueCorrect(String data, ArrayList<Variables> variables,
68                 ArrayList<Variables> localVariables ,ArrayList<Methods> methods)
69                         throws SjavaException {
70
71
72             if (data.matches(VAR_NAME) && !data.matches(Regex.ILEGAL)){
73                 Matcher match = pattVarName.matcher(data);
74                 match.find();
75                 Variables existVar = this.isVarExists(localVariables, variables, match.group(1));
76
77                 if (this.isCorrect(existVar) ) {
78
79                     return true;
80                 } else {
81                     throw new WrongValueTypeExcaption();
82                 }
83
84             } else if (data.matches(METHOD_NAME) && !data.matches(Regex.ILEGAL)){
85                 Matcher match = pattMethodName.matcher(data);
86                 match.find();
87
88                 Methods existMethod = Methods.isMethodExists(methods, match.group(1));
89                 if (existMethod != null && this.isCorrect(existMethod)) {
90                     existMethod.isCallLegal(match.group(2), variables,
91                             localVariables, methods);
92
93                     return true;
94                 } else {
95                     throw new WrongValueTypeExcaption();
96                 }
97             } else {
98                 if (this.isCorrect(data)) {
99                     return true;
100                } else {
101                    throw new WrongValueTypeExcaption();
102                }
103            }
104
105        }
106
107        /**
108         * check if the data is similar to the variable type
109         * @param data - string of a value to assign
110         * @return true iff the data type is correct
111         * @throws SjavaException
112         */
113        public boolean isCorrect(String data) throws SjavaException{
114            return (data.matches(value));
115        }
116
117        /**
118         * check if the given variable is initialized and from the same type of this
119         * variable
120         * @param var - the variable to assign to this one
121         * @return true iff the assignment is legal
122         */
123        public boolean isCorrect(Variables var){
124            return var != null && var.initialized &&
125                    this.type.equals(var.type) && !this.isArray;
126        }
127        /**
```

```
128         * check if the given method return the same type of this one
129         * @param method the method to assign to this one
130         * @return true iff the assignment is legal
131         * @throws WrongValueTypeExcaption  - in case method is void
132         */
133        public boolean isCorrect(Methods method) throws SjavaException{
134            if (method.returnType == null)
135                throw new WrongValueTypeExcaption();
136            return this.type.equals(method.returnType.type);
137        }
138
139        /**
140         * returns true iff the given string is equal to the variable name
141         * @param name - the name to check
142         * @return true iff the given string is equal to the variable name
143         */
144        public boolean isEqual(String name){
145            return this.name.equals(name);
146        }
147
148        /**
149         * check if there a variable with the given name
150         * @param variables - outer scope variables list
151         * @param localVariables - local variables list
152         * @param varName - variable name
153         * @return the variable with the given name or null if not exist
154         */
155        public static Variables isVarExists (ArrayList<Variables> localVariables,
156                ArrayList<Variables> variables, String varName){
157            for (Variables var:localVariables){
158                if (var.isEqual(varName))
159                    return var;
160            }
161            for (Variables var:variables){
162                if (var.isEqual(varName))
163                    return var;
164            }
165            return null;
166        }
167    }
```

34 oop/ex7/variables/WrongValueTypeExcaption.java

```java
/**
 *
 */
package oop.ex7.variables;

import oop.ex7.main.SjavaException;

/**
 * @author roigreenberg
 *
 */
public class WrongValueTypeExcaption extends SjavaException {

    /**
     * default constructor
     */
    public WrongValueTypeExcaption() {
        super("wrong value for variable");
        System.err.println("wrong value for variable");
    }

    /**
     * constructor
     * @param exception - messege for the exception
     */
    public WrongValueTypeExcaption(String exception) {
        super(exception);
        System.err.println(exception);
    }

}
```