2.3 Compilers and Operating Systems

< Creeper >

04/03/2022

by YangCheng (474340)

Content:

GitLab link: https://gitlab.com/saxion.nl/2.3-compilers-and-os/2122/73

Language name: Creeper

Introduction

My goal is to create a programming language suitable for building buildings. I named it creeper. It can programmatically operate on floors, rooms, building names, and building status.

Some features of the language, the details are explained in the grammar :

1. This programming language is almost the same as java, it has 4 data types Floor, Name, Room, Finish corresponding to int, String, double and boolean in java.
2. Like Java, this language has a class called hello, so it only has methods. In creeper, all methods are static. The method must return a Finish type value(boolean value). Because in creeper, each method represents the construction of a building, and the Finish value needs to be returned to indicate whether the building is completed.
3. Statements outside the method will be executed by default in the main method. Code needs to be in order, for example: when you want to call a method, you need to make sure that this method has been created before.
4. Scanner only supports Name(String) input.
5. Like Java, Name(String) comparison uses eq (“a” eq “b”).

Grammar

1. Variables

The language supports storing values in variables and using those in expressions. I implemented 4 data types:

Floor - used to define a variable with an integer value.

Name - used to define a variable with a string value.

Room - used to define a variable with a double value.

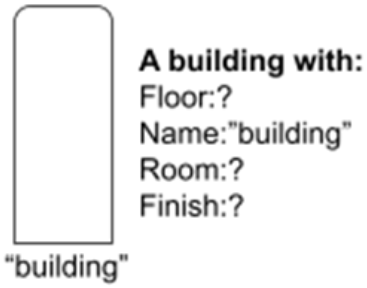
Finish - used to define a variable with a boolean value. It has only 2 possible values: 'yes' or 'no'.

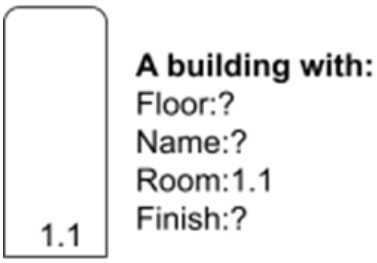
Note: *Variable names only can consist of letters. Its declaration and value definition can be done separately or together. It can be declared outside and inside control flow statements. A 'Floor' value can also be negative.*

Examples:

Initialize	***Assignment value***	***Visualizing***

Floor x	x = 4;
	
	or Floor x = 4;
		

Name xx;	xx = “building”; or Name xx = “building”;	

Room xxx;	xxx = 1.1;
	
	or Room xxx = 1.1;
		

Finish xxxx;	xxxx = Yes;
	
	or Finish xxxx = Yes;
		

2. Operators

It has 3 types of operators:

1. Logical operators:

'=’ - assign value;

‘==’ - compare values(Supported data types: Floor, Room, Finish);

‘eq’ - compare values(Supported data type: Name);

‘!=’ - not equal to num;

‘&&’ - logical operator, which means “and”;

‘||’ -logical operator, which means “or”;

Note: When using && or ||, you need to add () to the values on the left and right sides. like: (xx)&&(xx), (xx)||(xx). for example: (9>=9)&&(“q” eq “q”)

2. Comparison operators:

‘>’ -means greater than

‘<’ -means less than

‘>=’ -means greater than or equal to

‘<=’ -means less than or equal to

3. Mathematical operators:

‘+’ -addition

‘-‘ -subtraction

‘*’ -multiplication

‘/’ -division

Both comparison and logical operators can be used in statements as a part of an argument. Mathematical operators can be a part of variable value definition. These operators have a predefined priority (multiplication precedes addition/subtraction).

3. Negate

Negative numbers can be achieved by adding "-" before the Floor(int) and Room(double) types. for example: -8 , -8.5

4. Scanner

The scanner is implemented similarly to Java. But the most different is It only supports string data type. The keyword is “Enter<<”.

Enter<<

When the functional call is executed you have to fill in your response in a terminal to continue code execution. The result of the scan can be used as a value.

Example :

```
Name a = Enter<< ;
```

```
if( “aa” eq Enter<< ){ … };
```

5. Print

The keyword used for printing is "Show>>".

Show>> XXX All data types are supported(Floor, Name, Room , Finish)

Example :

```
Name a = “hello”;
Show>> a;
```

```
Floor b = 8;
Show>> b+2;
```

6. If(boolean){}else{}

This is a way to execute a certain piece of code or not depending on the value of a boolean expression.

if (boolean) {statement*};

if (boolean) {statement*} else {statement*};

“boolean“ is the result of comparing two or more values, supporting all data types. “statement*” are any other possible statements such as variable declarations, control flow and print statements etc.

It is possible to include if..else in if..else.

Example **:**

If (boolean) {statement};

Name a =“hello”;
if(a eq “morning”){
Show>>a;
};
	Floor b = 5;
	if(b==6){
	Show>>b;
	};
		Finish c = yes;
		if(c != no){
		Show>>c;
		};
			Room d = 5.5;
			if(d >= 5.5){
			Show>>d;
			};

If (boolean) {content} else {content};

Name a = Enter<<;
if (a eq “morning”) {
Show>>a;
}else{
Show>>”not morning”;
};
	Floor b = 8;
	if (b != 8) {
	b = 8;
	Show>>b;
	}else{
	Show>>b;
	};
		Finish c = yes;
		if (c != no) {
		c = no;
		Show>>c;
		}else{
		Show>>c;
		};
			Room d = 8.5;
			if (d<=9) {
			d = 9;
			Show>>d;
			}else{
			Show>>d;
			};

7. Repeat(loop)

The language also supports repeating certain pieces of code. The keyword is “Loop{statement*}Times(Int)”.

Loop {statement}Times(n);*

n must be an integer & bigger than 0.

It is possible to include loop in loop.

Example :

```
Floor a = 8;

Loop{a = a+2; Show>> a;}Times(3)
```

8. Method & Method call

language supports the creation of methods. There is no limit to the number of methods created. The keyword to create a method is "#Construct". Then follow the return value type of the method, which only supports finish(boolean) value type.

#Construct xxxx*<Z> {statement*} return(B);*

XXXX is the name of the method.

Z means parameters.

B means Finish value .

‘statement*’ are any other possible statements.

Finish a = xxx(Z);

XXXX is the name of the method.

Z means parameters.

After the method is created, you can call the method by creating a Finish value in the main method. Because the method always returns the Finish(boolean) value, you can directly compare the call method with the Finish(boolean) value.

Example :

```
//create function aa
#Construct aa <Floor m , Name n> {
m = 4;
…
}return(m>2);

//call function aa
Finish call = aa(8 , “hello”);

//compare the call_function with the Finish(boolean) value.
if(aa(8,“hello”) != no){Show>>yes;}else{Show>>no;};
```

9. Scope

There are two kinds of scopes: one is designed for loop and if_statement. When this scope cannot find the required variable name, it will automatically look for the same variable name in the parent scope.However, the variables declared inside the scope cannot be used outside the scope.

The second is designed for method. The method scope is relatively private, because the method cannot look for variables from the main method or other methods. Also, variables created inside the method cannot be used outside the method. if_statement or loop in a method can look for variables from the method scope.

In addition to loop and if_statement, you can specify the scope of certain pieces of code by

{ }

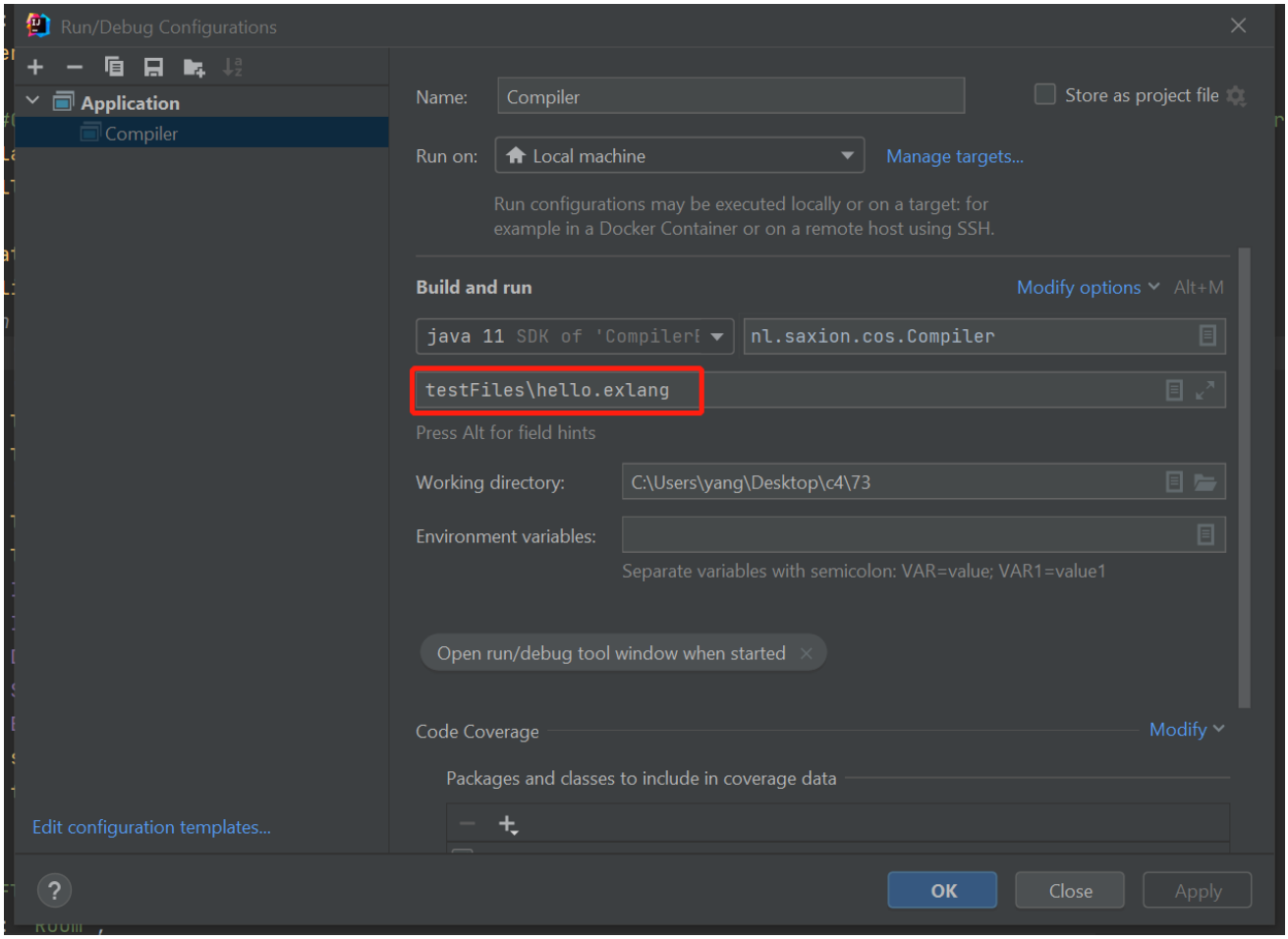
Example :

```
Floor a =8;

{
 Show>>a;
 Floor b = 9;
}

//error! Undefined variable: b
Show>>b;
```

Test

1. Test configuration

Add testFiles\hello.exlang to build and run configuration.



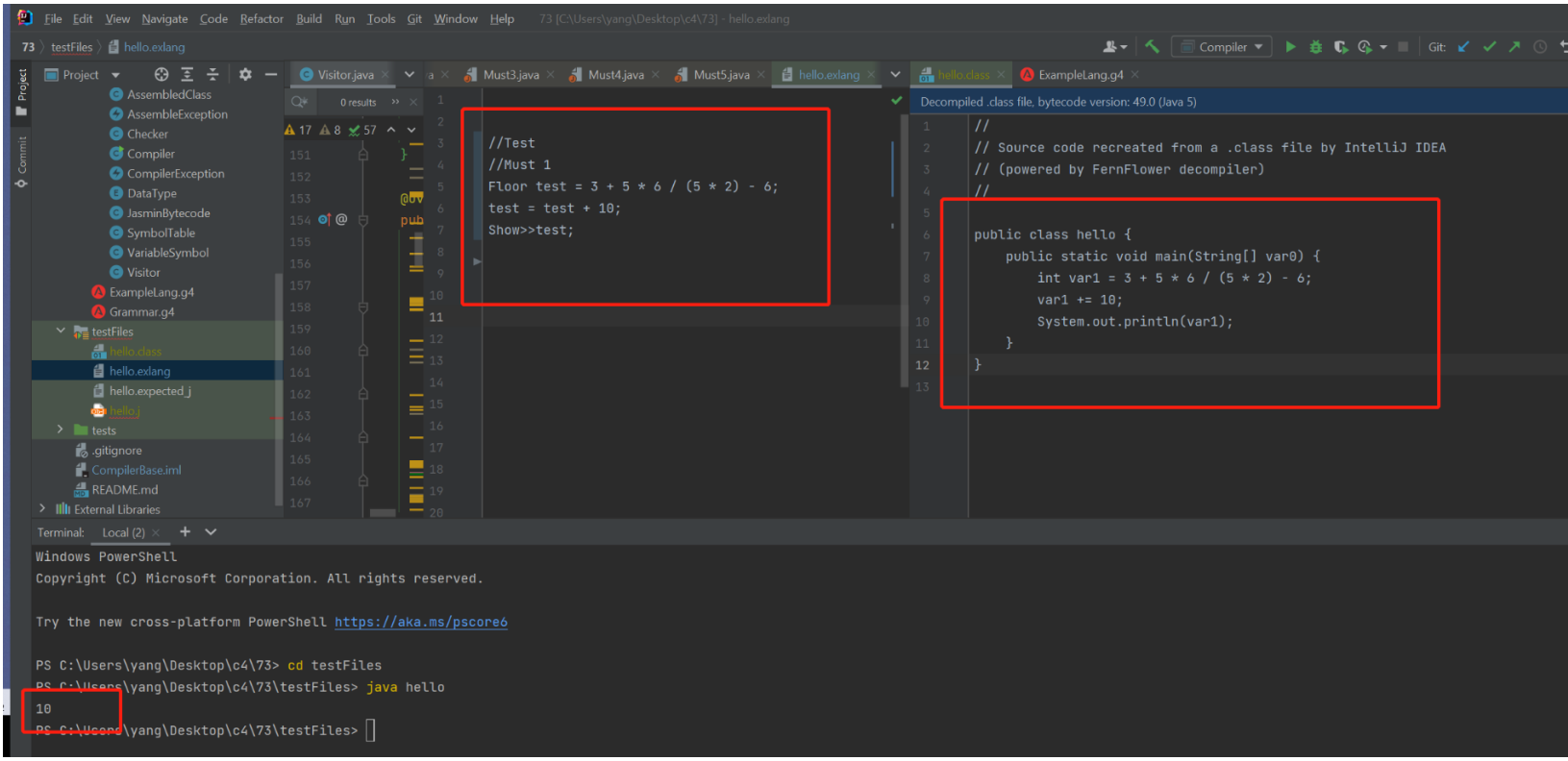
2. calculation test

test code:

hello.exlang:

```
Floor test = 3 + 5 * 6 / (5 * 2) - 6;

test = test + 10;

Show>>test;
```

output:



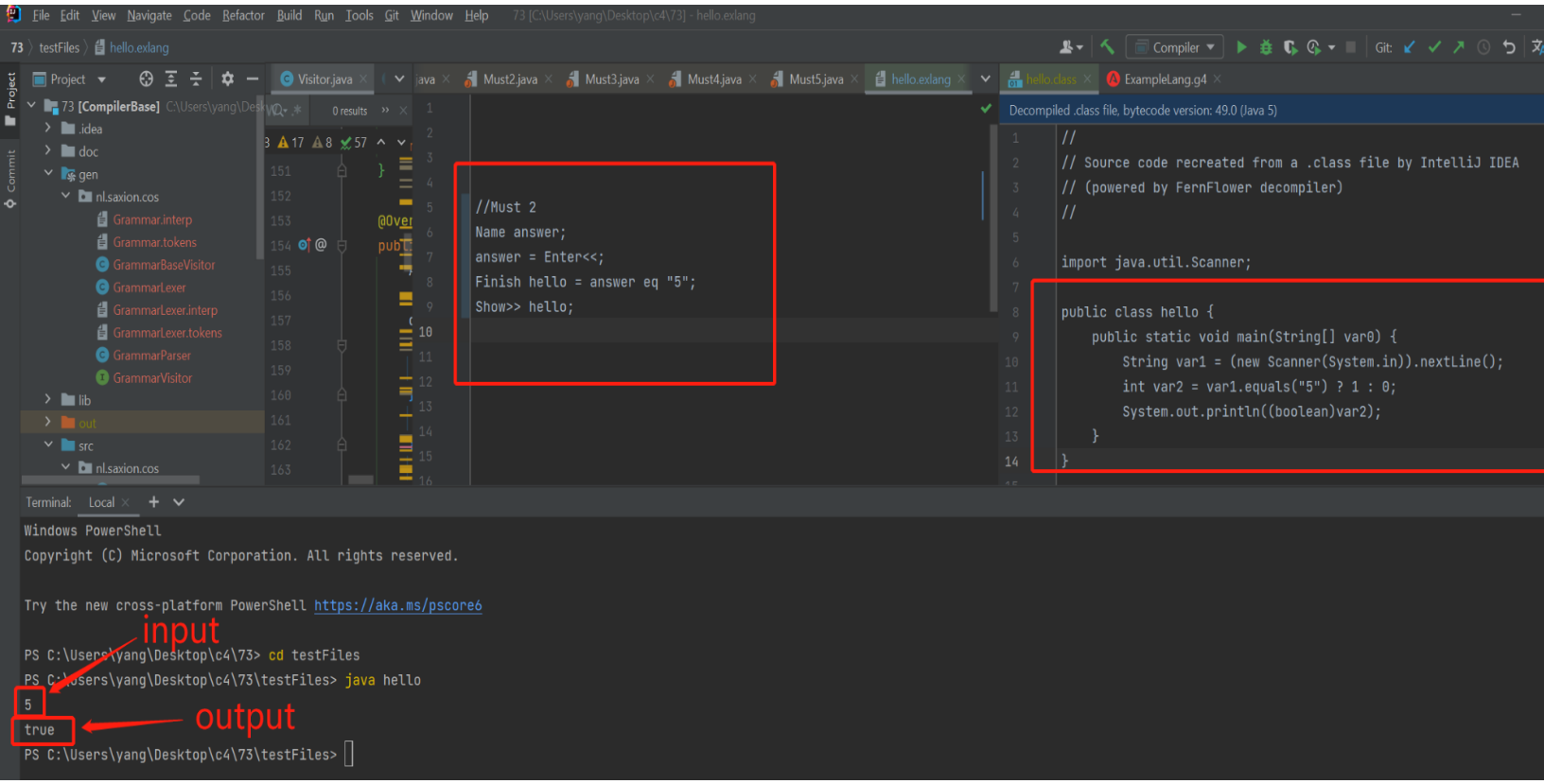
3. Scanner test

test code:

hello.exlang:

```
Name answer;
answer = Enter<<;
Finish hello = answer eq "5";
Show>> hello;
```

output:



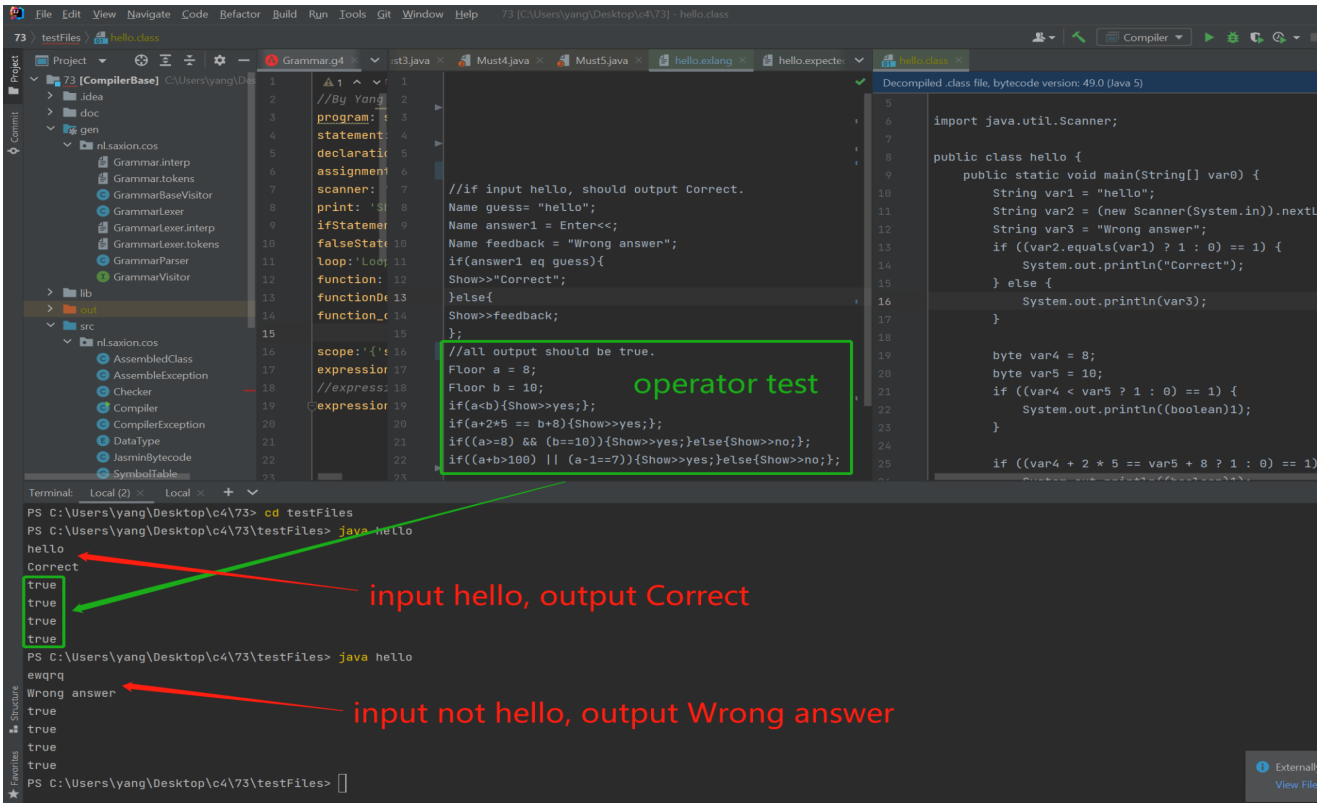
4. If statement test

test code:

hello.exlang:

```
//if input hello, should output Correct.
Name guess= "hello";
Name answer1 = Enter<<;
Name feedback = "Wrong answer";
if(answer1 eq guess){
Show>>"Correct";
}else{
Show>>feedback;
};
//Operator test, all output should be true.
Floor a = 8;
Floor b = 10;
if(a<b){Show>>yes;};
if(a+2*5 == b+8){Show>>yes;};
if((a>=8) && (b==10)){Show>>yes;}else{Show>>no;};
if((a+b>100) || (a-1==7)){Show>>yes;}else{Show>>no;};
```

output:



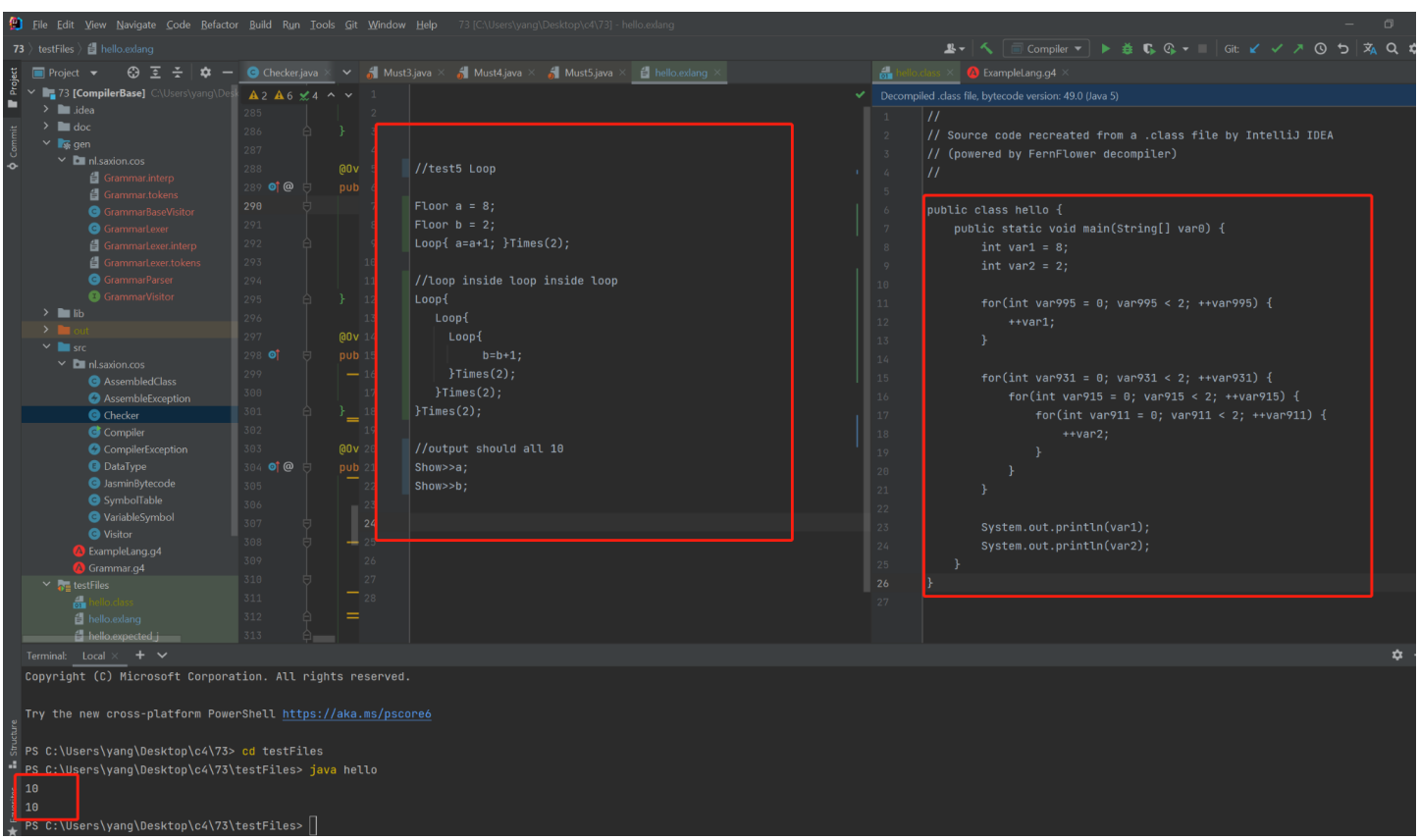
5. Loop test

test code:

hello.exlang:

```
Floor a = 8;
Floor b = 2;
Loop{ a=a+1; }Times(2);

//loop inside loop inside loop
Loop{
   Loop{
     Loop{
          b=b+1;
     }Times(2);
   }Times(2);
}Times(2);

//output should all 10
Show>>a;
Show>>b;
```

output:



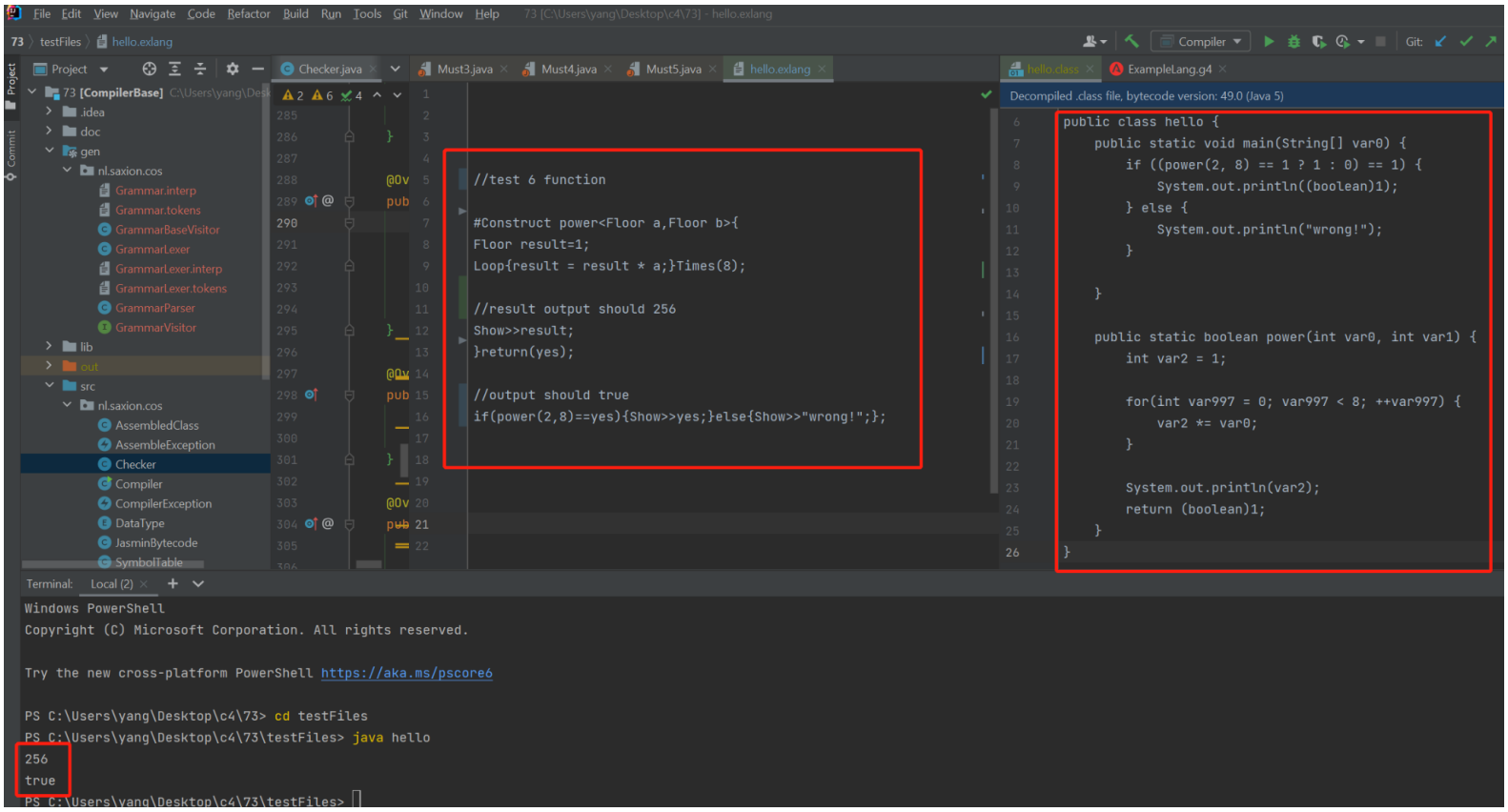
6. Method test

test code:

hello.exlang:

```
#Construct power<Floor a,Floor b>{
Floor result=1;
Loop{result = result * a;}Times(8);

//result output should 256
Show>>result;
}return(yes);

//output should true
if(power(2,8)==yes){Show>>yes;}else{Show>>"wrong!";};
```

output:



7. Scope test

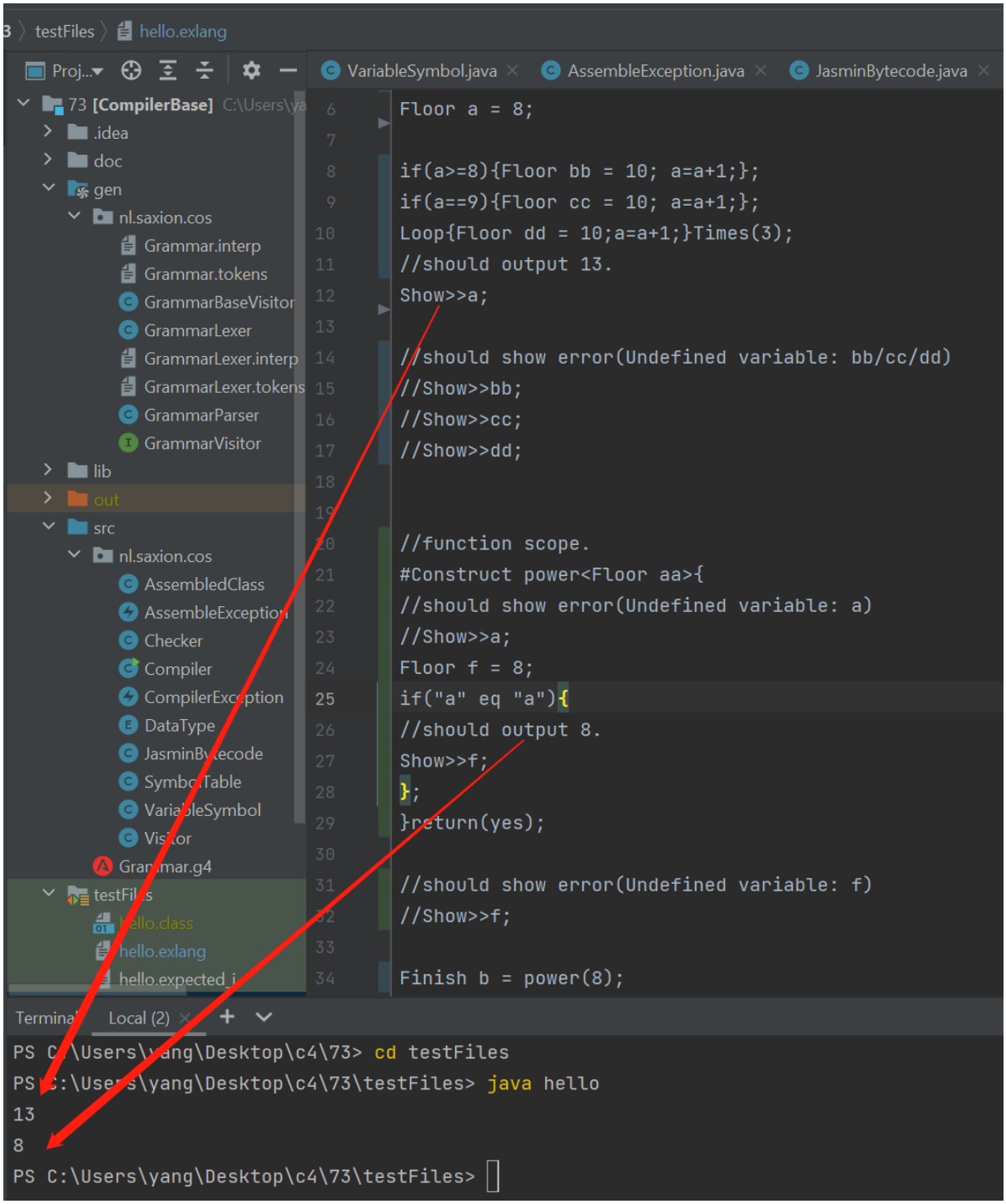
test code:

hello.exlang:

```
//Scope test
Floor a = 8;

if(a>=8){Floor bb = 10; a=a+1;};
if(a==9){Floor cc = 10; a=a+1;};
Loop{Floor dd = 10;a=a+1;}Times(3);
//should output 13.
Show>>a;

//should show error(Undefined variable: bb/cc/dd)
//Show>>bb;
//Show>>cc;
//Show>>dd;


//method scope.
#Construct power<Floor aa>{
//should show error(Undefined variable: a)
//Show>>a;
Floor f = 8;
if("a" eq "a"){
//should output 8.
Show>>f;
};
}return(yes);

//should show error(Undefined variable: f)
//Show>>f;

Finish b = power(8);
```

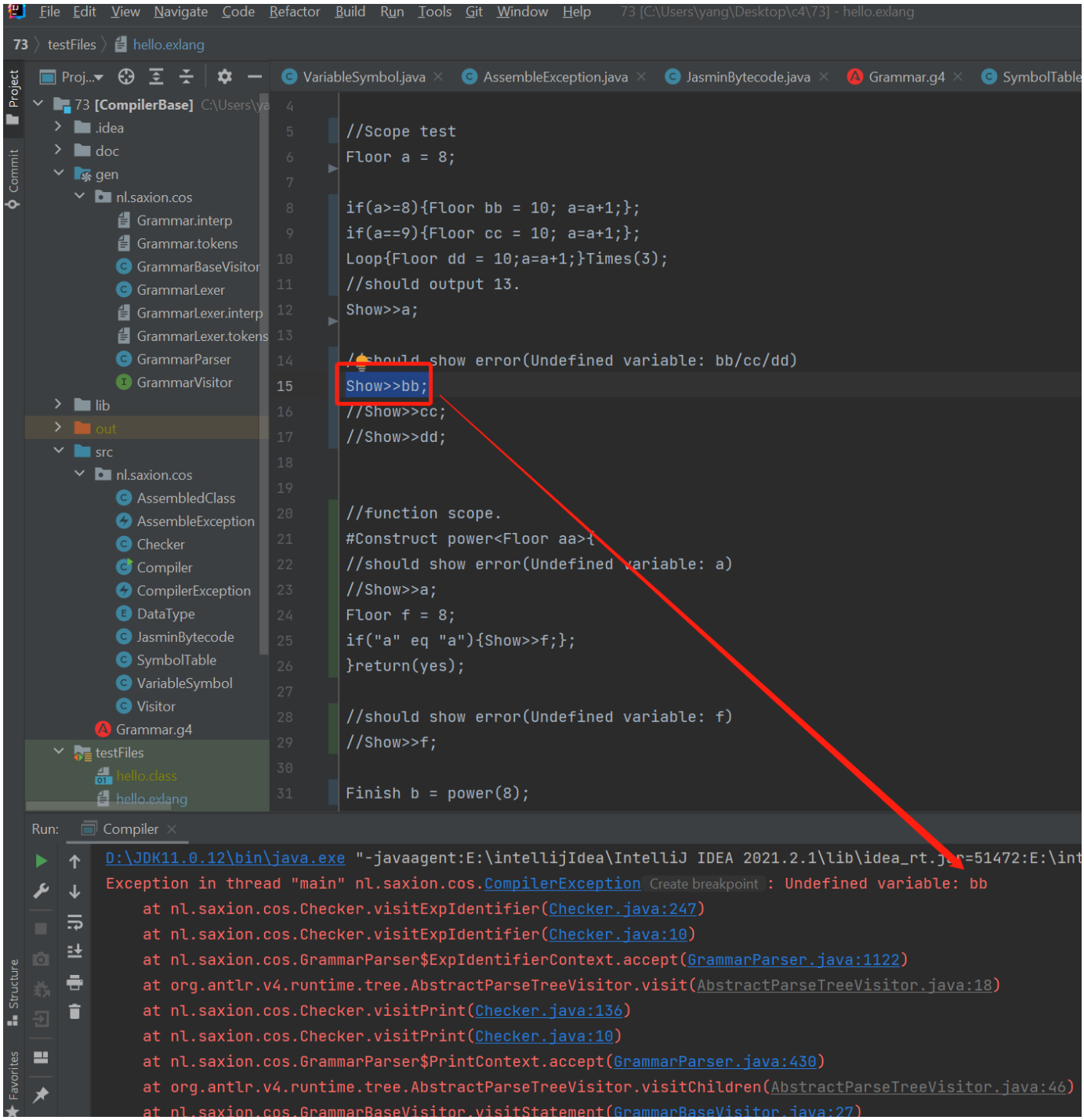
output:

1. comment all errors, should output 13 and 8. because In the scope of if statement and loop, it can look for the variable from the parent scope when the variable is not found in the current scope.



2. uncomment “Show>>bb”, show error(Undefined variable: bb)

because bb is created in the if statement, cannot be used outside the if statement.



3. uncomment “Show>>a”, show error(Undefined variable: a)

because In method scope, method cannot find variables from other methods or main method.

```
//Scope test
Floor a = 8;

if(a>=8){Floor bb = 10; a=a+1;};
if(a==9){Floor cc = 10; a=a+1;};
Loop{Floor dd = 10;a=a+1;}Times(3);
//should output 13.
Show>>a;

//should show error(Undefined variable: bb/cc/dd)
//Show>>bb;
//Show>>cc;
//Show>>dd;


//function scope.
#Construct power<Floor aa>{
//should show error(Undefined variable: a)
Show>>a;
Floor f = 8;
if("a" eq "a"){
//should output 8.
Show>>f;
};
}return(yes);

//should show error(Undefined variable: f)
//Show>>f;

Finish b = power(8);
```

```
D:\JDK11.0.12\bin\java.exe "-javaagent:E:\intellijIdea\IntelliJ IDEA 2021.2.1\lib\idea_rt.jar=51600:E:\
Exception in thread "main" nl.saxion.cos.CompilerException Create breakpoint : Undefined variable: a
    at nl.saxion.cos.Checker.visitExpIdentifier(Checker.java:247)
    at nl.saxion.cos.Checker.visitExpIdentifier(Checker.java:10)
    at nl.saxion.cos.GrammarParser$ExpIdentifierContext.accept(GrammarParser.java:1122)
    at org.antlr.v4.runtime.tree.AbstractParseTreeVisitor.visit(AbstractParseTreeVisitor.java:18)
    at nl.saxion.cos.Checker.visitPrint(Checker.java:136)
    at nl.saxion.cos.Checker.visitPrint(Checker.java:10)
```

4. uncomment “Show>>f”, show error(Undefined variable: f), because f is created in the method and cannot be used outside the method.

```
if(a>=8){Floor bb = 10; a=a+1;};
if(a==9){Floor cc = 10; a=a+1;};
Loop{Floor dd = 10;a=a+1;}Times(3);
//should output 13.
Show>>a;

//should show error(Undefined variable: bb/cc/dd)
//Show>>bb;
//Show>>cc;
//Show>>dd;


//function scope.
#Construct power<Floor aa>{
//should show error(Undefined variable: a)
//Show>>a;
Floor f = 8;
if("a" eq "a"){
//should output 8.
Show>>f;
};
}return(yes);

//should show error(Undefined variable: f)
Show>>f;

Finish b = power(8);
```

```
D:\JDK11.0.12\bin\java.exe "-javaagent:E:\intellijIdea\IntelliJ IDEA 2021.2.1\lib\idea_rt.jar=51568:E
Exception in thread "main" nl.saxion.cos.CompilerException Create breakpoint : Undefined variable: f
    at nl.saxion.cos.Checker.visitExpIdentifier(Checker.java:247)
    at nl.saxion.cos.Checker.visitExpIdentifier(Checker.java:10)
    at nl.saxion.cos.GrammarParser$ExpIdentifierContext.accept(GrammarParser.java:1122)
    at org.antlr.v4.runtime.tree.AbstractParseTreeVisitor.visit(AbstractParseTreeVisitor.java:18)
```

Exception test

1. Calculation exception

Addition, subtraction, multiplication and division first need to check whether the types on the left and right sides of the operator are the same. Then whether the type is int or double.

test code:

```
//test types on the left and right sides are not same.
Show>>"a"+8;
```

output:

```
aagent:E:\intellijIdea\IntelliJ IDEA 2021.2.1\lib\idea_rt.jar=53853:E:\intellijIdea\IntelliJ IDEA 2021.2.1\bin" -D
xion.cos.CompilerException Create breakpoint : add or sub error! Left and right type are not the same! at: "a"+8
itExpAdd(Checker.java:41)
itExpAdd(Checker.java:10)
er$ExpAddContext.accept(GrammarParser.java:1041)
.AbstractParseTreeVisitor.visit(AbstractParseTreeVisitor.java:18)
itPrint(Checker.java:136)
```

test code:

```
//test type not int or double.
Show>> "a"+“b”;
```

output:

```
IntelliJ IDEA 2021.2.1\lib\idea_rt.jar=53869:E:\intellijIdea\IntelliJ IDEA 2021.2.1\bin" -Dfile.encoding=UTF-8
ion Create breakpoint : add or sub error! not valid data type: STRING. Required type: int or double! at: "a"+"b"
6)
0)
(GrammarParser.java:1041)
```

2. Negate exception

check whether the type is int or double.

test code:

```
//test type not int or double.
Show>>-”e”;
```

output:

```
ijIdea\IntelliJ IDEA 2021.2.1\lib\idea_rt.jar=54059:E:\intellijIdea\IntelliJ IDEA 2021.2.1\bin" -Dfile.enco
erException Create breakpoint : negate error! at: "a". Provided type is: STRING. Required type: int or double.
cker.java:80)
cker.java:10)
text.accept(GrammarParser.java:1102)
eeVisitor.visit(AbstractParseTreeVisitor.java:18)
java:136)
```

3. Compare exception

Using comparison operators, you need to check whether the left and right types are the same. If using eq, need to check whether the left and right sides are Name(string) type.

test code:

```
//test types on the left and right sides are not the same.
Show>>7>"c";
```

output:

```
xe "-javaagent:E:\intellijIdea\IntelliJ IDEA 2021.2.1\lib\idea_rt.jar=65232:E:\intellijIdea\IntelliJ IDEA 2021.2.1\bi
n" nl.saxion.cos.CompilerException Create breakpoint : Compare error! at: (7>"c"). Diff datatype.
cker.visitExpCompare(Checker.java:114)
cker.visitExpCompare(Checker.java:10)
```

test code:

```
//when using eq, check types on the left and right sides are not Name(String).
Show>> 7 eq 8;
```

output:

```
ntellijIdea\IntelliJ IDEA 2021.2.1\lib\idea_rt.jar=62396:E:\intellijIdea\IntelliJ IDEA 2021.2.1\bin" -Dfile.
mpilerException Create breakpoint : eq error! not valid data type: INT. Required type: String! at: 7 eq 8
Checker.java:128)
Checker.java:10)
Context.accept(GrammarParser.java:1074)
rseTreeVisitor.visit(AbstractParseTreeVisitor.java:18)
cker.java:136)
```

4. logic operator exception

Finish(boolean) type is required on the left and right sides of the symbol && or ||.
It can be to compare the value first and then compare the logic, because using the compare operator will return the Finish(boolean) type like: (xx>xxx)&&(xx<=xxx).

test code:

```
//test when the left and right sides of the logical operator are not of Finish(boolean) type.
Show>> (7) && (8);
```

output:

```
Idea\IntelliJ IDEA 2021.2.1\lib\idea_rt.jar=55194:E:\intellijIdea\IntelliJ IDEA 2021.2.1\bin" -Dfi
xception Create breakpoint : logic error! not valid data type at: (7&&8). Required type: boolean
.java:144)
.java:10)
.accept(GrammarParser.java:1168)
```

5. Declaration exception

First check if the same variable name is declared repeatedly. Then If the user writes declaration and assignment together, like Floor a = 8; will check if the declared type matches the assignment type.

test code:

```
//When creating a variable repeatedly.
Name a = "hi";
Floor a =9;
```

output:

```
E:\intellijIdea\IntelliJ IDEA 2021.2.1\lib\idea_rt.jar=55331:E:\intellijIdea\Intelli
s.CompilerException Create breakpoint : Variable a already exist
ration(Checker.java:205)
ration(Checker.java:10)
arationContext.accept(GrammarParser.java:292)
```

test code:

```
//test if the declared type not matches the assignment type.
Name a = 8;
```

output:

```
a\IntelliJ IDEA 2021.2.1\lib\idea_rt.jar=55368:E:\intellijIdea\IntelliJ IDEA 2021.2
ption Create breakpoint : Declaration error! Incompatible datatypes. At Variable a.
.java:218)
.java:10)
.accept(GrammarParser.java:292)
itor.visitChildren(AbstractParseTreeVisitor.java:46)
```

6. Identifier exception

It will first check whether the value to be used is declared, and then check whether the value has been assigned.

test code:

```
//test If the value is not declared.
Show>>a;
```

output:

```
ellijIdea\IntelliJ IDEA 2021.2.1\lib\idea_rt.jar=55399:E
ilerException Create breakpoint : Undefined variable: a
er(Checker.java:247)
er(Checker.java:10)
ierContext.accept(GrammarParser.java:1122)
```

test code

```
//test If the value is not assigned.
Room a;
Show>>a;
```

output:

```
Exception Create breakpoint : Unassigned variable: a
hecker.java:250)
hecker.java:10)
ontext.accept(GrammarParser.java:1122)
```

7. Assignment exception

First check whether the value to be assigned has been declared, and then check whether the value matches the type of the variable

test code:

```
//test If the value is not declared.
a=8;
```

output:

```
pilerException Create breakpoint : Undefined variable in assignment: a
(Checker.java:262)
(Checker.java:10)
tContext.accept(GrammarParser.java:355)
```

test code:

```
// test if the declared type not matches the assignment type.
Name a;
a=8;
```

output:

```
lijIdea\IntelliJ IDEA 2021.2.1\lib\idea_rt.jar=55487:E:\intellijIdea\In
erException Create breakpoint : Incompatible datatypes. At variable: a
ecker.java:267)
ecker.java:10)
ntext.accept(GrammarParser.java:355)
```

8. If statement exception

In the if statement, check whether the argument is boolean type.

test code:

```
//test if argument not boolean type.
if(9){Show>>"hi";};
```

output:

```
t:E:\intellijIdea\IntelliJ IDEA 2021.2.1\lib\idea_rt.jar=55624:E:\intellijIdea\IntelliJ
cos.CompilerException Create breakpoint : if statement arg is not a Boolean data type
tatement(Checker.java:184)
tatement(Checker.java:10)
StatementContext.accept(GrammarParser.java:479)
```

9. Method exception

Check if the method return value is boolean type.

testcode:

```
//test if the method return value not a boolean type.
#Construct power<Floor a,Floor b>{
Floor result=1;
Loop{result = result * a;}Times(8);
//result output should 256
Show>>result;
Floor uu = 5;
}return(5);
```

output:

```
tellijIdea\IntelliJ IDEA 2021.2.1\lib\idea_rt.jar=64387:E:\intellijIdea\IntelliJ IDEA 2021.2.1\bin" -
pilerException Create breakpoint : Method return error! Required type:boolean! At method name: power@ii
cker.java:298)
cker.java:10)
text.accept(GrammarParser.java:694)
```

10. Method_call exception

Check if method is declared

test code:

```
//test if the method is not defined.
Finish a = power(8,9);
```

output:

```
CompilerException Create breakpoint : Method power(Floor,Floor) is undefined.
call(Checker.java:358)
call(Checker.java:10)
_callContext.accept(GrammarParser.java:844)
```