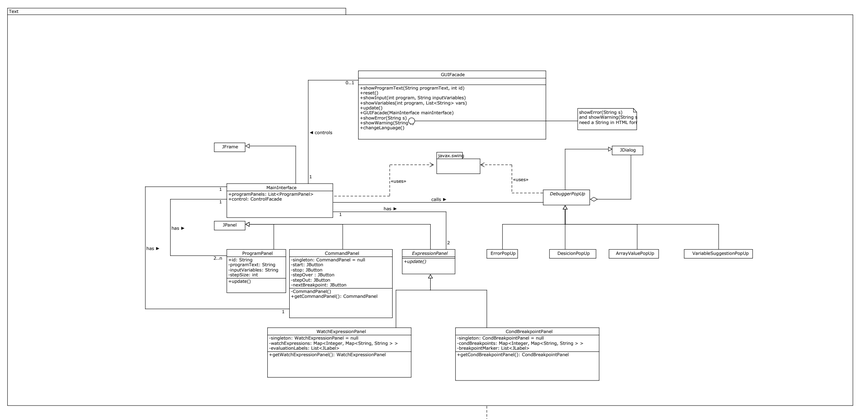
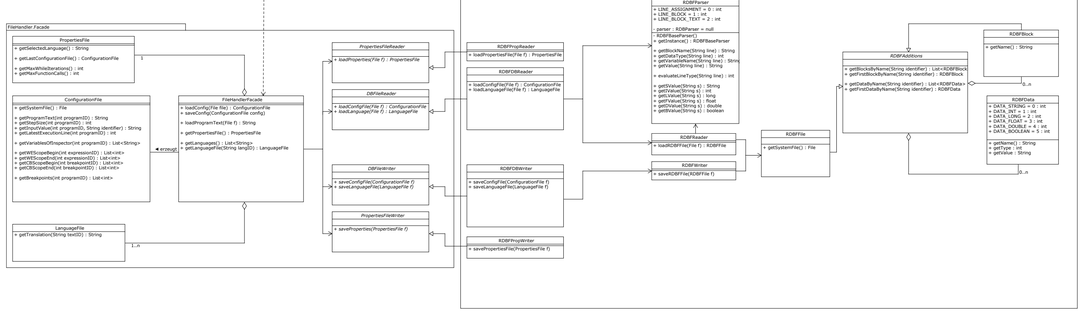
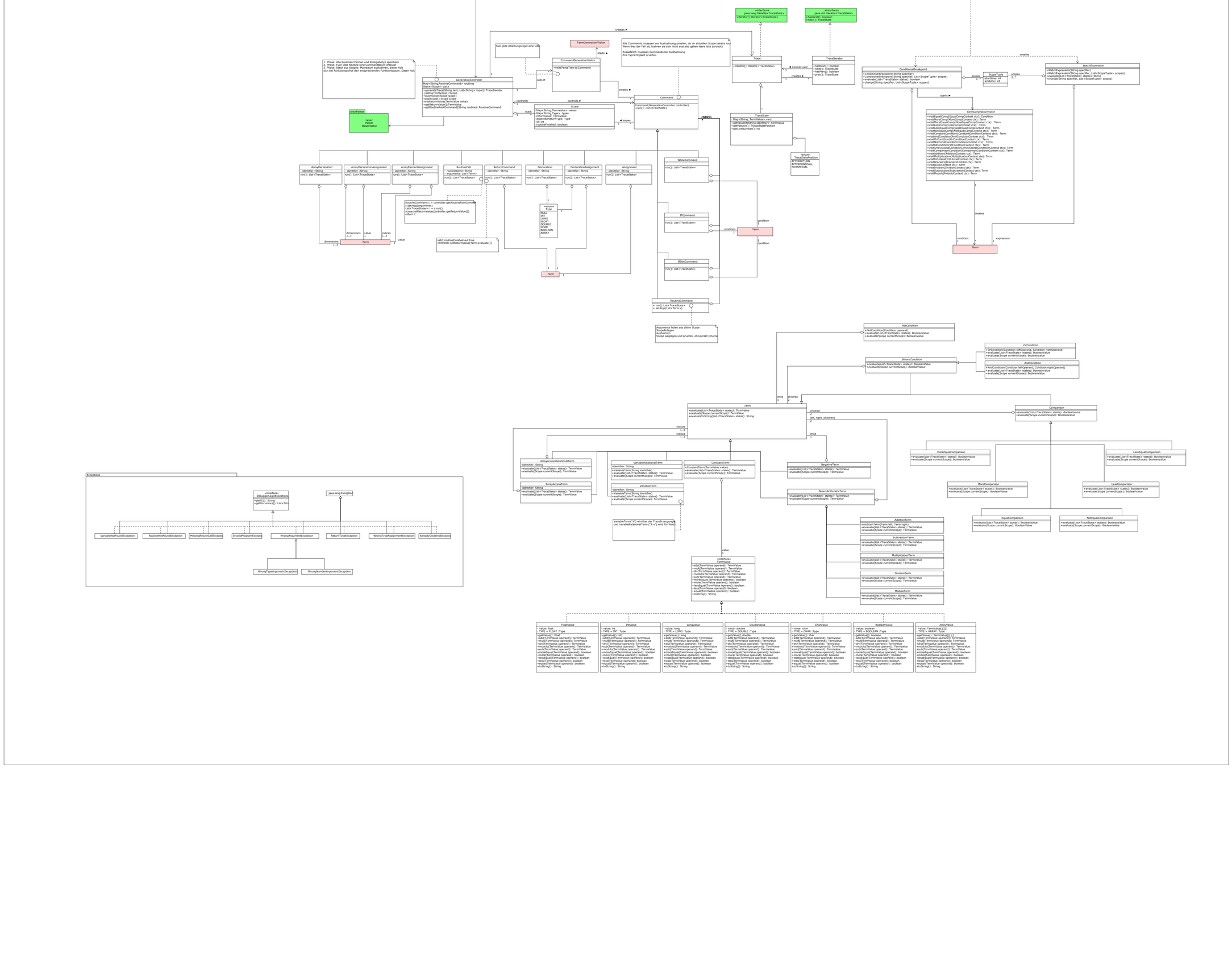

Text
GUIFacade
+showProgramText(String programText, int id)
+reset()
+showInputs(int program, String inputVariables)
+showBreakpoints(int program, List<String> vars)
+update()
+GUIFacade(MainInterface mainInterface)
+showError(String s)
+showWarning(String s)
+changeLanguage()
showError(String s)
and showWarning(String s)
need a String in HTML tag
0..1
controls
JFrame
1
MainInterface
+programPanels: List<ProgramPanel>
+control: ControlFacade
1
1
has
has
1
JPanel
2..n
ProgramPanel
-programText: String
-inputVariables: String
-stepSize: int
+update()
CommandPanel
-singleton: CommandPanel = null
-start: JButton
-stop: JButton
-stepOver : JButton
-stepOut: JButton
-stepBack: JButton
-singleStep: JButton
+CommandPanel()
+getCommandPanel(): CommandPanel
2
ExpressionPanel
+update()
1
WatchExpressionPanel
-singleton: WatchExpressionPanel = null
-watchExpressions: List<String>, Map<String, String > >
-evaluationLabels: List<JLabel>
+getWatchExpressionPanel(): WatchExpressionPanel
CondBreakpointPanel
-singleton: CondBreakpointPanel = null
-conditionsMap: List<String>, Map<String, String > >
-breakpointMarker: List<JLabel>
+getCondBreakpointPanel(): CondBreakpointPanel
calls
has
uses
uses
JDialog
DebuggerPopUp
ErrorPopUp
DescionPopUp
ArrayValuePopUp
VariableSuggestionPopUp
Text
+access+
ControlFacade
+ ControlFacade(GuiFacade guiFacade)
+ setStepSize(int program, int size)
+ step(int type)
+ step()
+ singleStep(int programID)
+ stepBack()
+ createWatchExpression(int id, String expr)
+ changeWatchExpression(int id, String expr, List<ScopeTuple> scope)
+ deleteWatchExpression(int id)
+ createConditionalBreakpoint(int id, String cond)
+ changeConditionalBreakpoint(int id, String cond, List<ScopeTuple> scope)
+ deleteConditionalBreakpoint(int id)
+ createSynchronousBreakpoint(int line)
+ createBreakpoint(int program, int line)
+ deleteBreakpoint(int program, int line)
+ deleteAllBreakpoints()
+ saveText(List<String> programtexts, List<String> inputvar)
+ startDebug()
+ stopDebug()
+ reset()
+ loadConfiguration(File configFile)
+ saveConfiguration(File configFile)
+ loadProgramText(File file)
+ getAvailableLanuages() : List<String>
+ setMaximumIterations(int i)
+ setMaximumFunctionCalls(int i)
+ suggestStepSize() : String
+ suggestWatchExpression() : String
+ suggestConditionalBreakpoint(): String
+ suggestInputValue(String id, String range, int type): String
+ selectStepSizeStrategy(int id)
+ selectRelationalExpressionStrategy(int id)
+ selectInputValueStrategy(int id)
+ changeLanguage(String lang)
debugLogicFacade : debuglogic.debugger::DebugLogicFacade
= new debuglogic.debugger::DebugLogicFacade()
debugLogicFacade.attach(guiFacade);
...
ExceptionHandler
+ ExceptionHandler(ControlFacade facade)
+ handle(Exception exception)
1
delegates to
delegates to
controller
FileHandler Interactor
0..1
FileHandlerInteractor
+ loadConfigFile(configFile)
+ saveConfigFile(configFile)
+ loadProgramText(File file)
+ changeLanguage(String lang)
+access+
1
trigger change in presentation
Filehandler.facade::LanguageFile
userinterface::GuiFacade
DebugLogicController
+ setStepSize(int program, int size)
+ step(int type)
+ step()
+ singleStep()
+ stepBack()
+ createWatchExpression(int id, String expr)
+ changeWatchExpression(int id, String expr, List<ScopeTuple> scope)
+ deleteWatchExpression(int id)
+ createCondBreakpoint(int id, String cond)
+ changeCondBreakpoint(int id, String cond, List<ScopeTuple> scope)
+ deleteCondBreakpoint(int id)
+ createSyncBreakpoint(int line)
+ createBreakpoint(int program, int line)
+ deleteBreakpoint(int program, int line)
+ deleteAllBreakpoints()
+ saveText(List<String> programtexts, List<String> inputvar)
+ startDebug()
+ stopDebug()
+ reset()
+ suggestStepSize() : String
+ suggestWatchExpression() : String
+ suggestConditionalBreakpoint(): String
+ suggestInputValue(): String
+ selectStepSizeStrategy()
+ selectRelationalExpressionStrategy()
+ selectInputValueStrategy()
debugLogicFacade.setStepSize(int program, int size)
...
FileHandler.Exceptions
«interface»
FileHandlerException
+getDE() : String
java.lang.Exception
«throws»
LanguageNotFoundException
ParseBlockException
ParseAssignmentException
InvalidLineTypeException
Filehandler.Facade
PropertiesFile
+ getSelectedLanguage() : String
+ getLastConfigurationFile() : ConfigurationFile
+ getMaxWhileIterations() : int
+ getMaxFunctionCalls() : int
1
ConfigurationFile
+ getSystemFile() : File
+ getProgramText(int programID) : String
+ getInputVariables(int programID) : String
+ getInputValue(int programID, String identifier) : String
+ getLatestExecutionLine(int programID) : int
+ getVariableOfInspector(int programID) : List<String>
+ getWESScopeBegin(int expressionID) : List<int>
+ getWESScopeEnd(int expressionID) : List<int>
+ getBPScopeBegin(int breakpointID) : List<int>
+ getBPScopeEnd(int breakpointID) : List<int>
+ getBreakpoints(int programID) : List<int>
FileHandlerFacade
+ loadConfig(File file) : ConfigurationFile
+ saveConfig(ConfigurationFile config)
+ loadProgramText(File f) : String
+ getPropertiesFile() : PropertiesFile
+ getLanguages() : List<String>
+ getLanguageFile(String lang[0]) : LanguageFile
+errorcode
LanguageFile
+ getTranslation(String text[0]) : String
1..n
PropertiesFileReader
+ loadPropertiesFile(File f) : PropertiesFile
DBFileReader
+ loadConfigFile(File f) : ConfigurationFile
+ loadLanguageFile(File f) : LanguageFile
DBFileWriter
+ saveConfigFile(ConfigurationFile f)
+ saveLanguageFile(LanguageFile f)
PropertiesFileWriter
+ saveProperties(PropertiesFile f)
Filehandler.RDBF
RDBFPropReader
+ loadPropertiesFile(File f) : PropertiesFile
RDBFDBReader
+ loadConfigFile(File f) : ConfigurationFile
+ loadLanguageFile(File f) : LanguageFile
RDBFDBWriter
+ saveConfigFile(ConfigurationFile f)
+ saveLanguageFile(LanguageFile f)
RDBFPropWriter
+ savePropertiesFile(PropertiesFile f)
RDBFParser
+ LINE_ASSIGNMENT = 0 : int
+ LINE_BLOCK = 1 : int
+ LINE_BLOCK_TEXT = 2 : int
-parser : RDBFParser = null
- RDBFBaseParser()
+ getInstance() : RDBFBaseParser
+ getBlockName(String line) : String
+ getDataType(String line) : int
+ getVariableName(String line) : String
+ getValue(String line) : String
+ evaluateLineType(String line) : int
+ getValue(String s) : String
+ getValue(String s) : int
+ getLvalue(String s) : long
+ getValue(String s) : float
+ getValue(String s) : double
+ getValue(String s) : boolean
RDBFReader
+ loadRDBFFile(File f) : RDBFFile
RDBFWriter
+ saveRDBFFile(RDBFFile f)
RDBFFile
+ getSystemFile() : File
RDBFAddStore
+ getBlockByName(String identifier) : List<RDBFBlock>
+ getFirstBlockByName(String identifier) : RDBFBlock
+ getDataByName(String identifier) : List<RDBFData>
+ getFirstDataByName(String identifier) : RDBFData
RDBFBlock
+ getName() : String
0..n
RDBFData
+ DATA_STRING = 0 : int
+ DATA_INT = 1 : int
+ DATA_LONG = 2 : int
+ DATA_FLOAT = 3 : int
+ DATA_DOUBLE = 4 : int
+ DATA_BOOLEAN = 5 : int
+ getName() : String
+ getType() : int
+ getValue : String
0..n
Text
controls model via
Subject
+attach(Observer observer)
+detach(Observer observer)
+notify()
model's facade
DebugLogicFacade
+ STRAT_STEP_SIZE_SIMPLE = 0 : int
+ STRAT_RES_SIMPLE = 1 : int
+ STRAT_INPUT_SIMPLE = 2 : int
+ setStepSize(int program, int size)
+ step(int type)
+ singleStep(int programID)
+ continue()
+ createWatchExpression(int id, String expr)
+ changeWatchExpression(int id, String expr, List<ScopeTuple> sco
+ deleteWatchExpression(int id)
+ createConditionalBreakpoint(int id, String cond)
+ changeConditionalBreakpoint(int id, String cond, List<ScopeTuple
+ deleteConditionalBreakpoint(int id)
+ createBreakpoint(int program, int line)
+ deleteBreakpoint(int program, int line)
+ launchRun(List<DebugLogic.ProgramInput> programInput)
+ getProgramCounter(): List<int>
+ reset()
+ setMaximumIterations(int i)
+ setMaximumFunctionCalls(int i)
+ deleteAllBreakpoints()
+ suggestStepSize(List<String> programText) : String
+ suggestWatchExpression(List<String> programText) : String
+ suggestConditionalBreakpoint(List<String> programText): String
+ suggestInputValue(String identifier, String range, int type): String
+ selectStepSizeStrategy(int id)
+ selectRelationalExpressionStrategy(int id)
+ selectInputValueStrategy(int id)
ProgramInput
- text : String
- inputVars : String>
- counter : int
+setA()
delegates
delegates
DebugControl
+ STEP_NORMAL = 0 : int
+ STEP_OUT = 1 : int
+ STEP_OVER = 2 : int
+ STEP_BACK = 3 : int
+ launchRun(List<ProgramInput> programs)
+ step(int type)
+ singleStep(int programID)
+ continue()
+ createWatchExpression(int id, String expr)
+ changeWatchExpression(int id, String expr, List<ScopeTuple> scope)
+ deleteWatchExpression(int id)
+ createCondBreakpoint(int id, String cond)
+ changeCondBreakpoint(int id, String cond, List<ScopeTuple> scope)
+ deleteCondBreakpoint(int id)
+ createBreakpoint(int program, int line)
+ deleteBreakpoint(int program, int line)
+ getProgramCounter(): List<int>
+ setMaximumIterations(int i);
+ setMaximumFunctionCalls(int i);
+ deleteAllBreakpoints()
Breakpoint
+ getLine() : int
+ getProgramID() : int
list_breakpoints
Interpreter.WatchExpressions
list_watchExpressions
Interpreter.ConditionalBreakpo
list_condBreakpoints
Interpreter.TraceIterator
list_traceIterator
StepSizeSuggestion
+ suggest(List<String> programText)
RelationalSuggestion
+ suggestWatchExpression() : String
+ suggestConditionalBreakpoint() : St
InputValueSuggestion
+ TYPE_CHAR = 0 : int
+ TYPE_INT = 1 : int
+ TYPE_LONG = 2 : int
+ TYPE_FLOAT = 3 : int
+ TYPE_DOUBLE = 4 : int
+ TYPE_BOOLEAN = 5 : int
+ suggest(String identifier, String range, int type)
SimpleInputSuggestion
+ suggest(List<String> programText)
SimpleRelationalSuggestion
+ suggestWatchExpression() : String
+ suggestConditionBreakpoint() : Stri
SimpleInputSuggestion
+ suggest(String identifier, String range, int type) : String


Filehandler.Facade
PropertiesFile
+ getSelectedLanguage() : String
+ getLastConfigurationFile() : ConfigurationFile
+ getMaxWhileIterations() : int
+ getMaxFunctionCalls() : int
1
ConfigurationFile
+ getSystemFile() : File
+ getProgramText(int programID) : String
+ getInputVariables(int programID) : String
+ getInputValue(int programID, String identifier) : String
+ getLatestExecutionLine(int programID) : int
+ getVariableOfInspector(int programID) : List<String>
+ getWESScopeBegin(int expressionID) : List<int>
+ getWESScopeEnd(int expressionID) : List<int>
+ getBPScopeBegin(int breakpointID) : List<int>
+ getBPScopeEnd(int breakpointID) : List<int>
+ getBreakpoints(int programID) : List<int>
FileHandlerFacade
+ loadConfig(File file) : ConfigurationFile
+ saveConfig(ConfigurationFile config)
+ loadProgramText(File f) : String
+ getPropertiesFile() : PropertiesFile
+ getLanguages() : List<String>
+ getLanguageFile(String lang[0]) : LanguageFile
+errorcode
LanguageFile
+ getTranslation(String text[0]) : String
1..n
PropertiesFileReader
+ loadPropertiesFile(File f) : PropertiesFile
DBFileReader
+ loadConfigFile(File f) : ConfigurationFile
+ loadLanguageFile(File f) : LanguageFile
DBFileWriter
+ saveConfigFile(ConfigurationFile f)
+ saveLanguageFile(LanguageFile f)
PropertiesFileWriter
+ saveProperties(PropertiesFile f)


Text
call
creates
1: Phase: Alle Routinen trennen und Rückgabetyp speichern
2: Phase: Alle Variablen und Parameter speichern
3: Phase: Stack aus Scopes: Mainbaum aufbauen, dieber holt sich bei Funktionsaufruf den entsprechenden Funktionsbaum. Daten holt
GenerationController
Map<String,RoutineCommand> routines
Stack<Scope> stack
+generateTrace(String text, List<String> input): TraceIterator
+getCurrentScope():Scope
+getGlobalScope():Scope
+popScope():Scope scope
+setReturnValue(TermValue value)
+getReturnValue():TermValue
+getRoutineNeed(Command)(String routine): RoutineCommand
controller
stack
controlls
Scope
-Map<String,TermValue> values
-Map<String,Type> types
-returnValue: TermValue
-returnValueExpectedType: Type
-id: int
-routineFinished: boolean
knows
Command
Command(GenerationController controller)
+run(): List<TraceState>
children
TraceState
- Map<String, TermValue> vars
+getVariable(String identifier): TermValue
+getPosition(): TraceStatePosition
+getLineNumber(): int
«enum»
TraceStatePosition
BEFORE
AFTERRETURN
AFTERFUNCCALL
NOTSPECIAL
Trace
+Iterator():Iterator<TraceState>
iterates over
creates
TraceIterator
+hasNext(): boolean
+next(): TraceState
+peek(): TraceState
+prev(): TraceState
«interface»
java.lang.Iterable<TraceState>
+Iterator():Iterator<TraceState>
«interface»
java.util.Iterator<TraceState>
+hasNext(): boolean
+next(): TraceState
ConditionalBreakpoint
+ConditionalBreakpoint(String specifier)
+ConditionalBreakpoint(String specifier, List<ScopeTuple> scopes)
+evaluate(List<TraceState> states): TermValue
+changeQ(String specifier, List<ScopeTuple> scopes)
scopes
ScopeTuple
-startLine: int
-endLine: int
WatchExpression
+WatchExpression(String specifier)
+WatchExpression(String specifier, List<ScopeTuple> scopes)
+evaluate(List<TraceState> states): TermValue
+changeQ(String specifier, List<ScopeTuple> scopes)
starts
TermGenerationVisitor
+visitEqualCompC(EqualCompContext ctx): Condition
+visitNotEqualCompC(NotEqualCompContext ctx): Condition
+visitMoreEqualCompC(MoreEqualCompContext ctx): Term
+visitLessCompC(LessCompContext ctx): Term
+visitMoreCompC(MoreCompContext ctx): Term
+visitNotEqualCompC(NotEqualCompContext ctx): Term
+visitConstantCondition(ConstantCondition Context ctx): Term
+visitAndCondition(AndConditionContext ctx): Term
+visitOrCondition(OrConditionContext ctx): Term
+visitNotCondition(NotConditionContext ctx): Term
+visitIdCondition(IdConditionContext ctx): Term
+visitArrayAccessCondition(ArrayAccessConditionContext ctx): Term
+visitBinaryCondition(BinaryConditionCondition Context ctx): Term
+visitAdditionC(AdditionC ontext ctx): Term
+visitMultiplicationC(MultiplicationC ontext ctx): Term
+visitDivisionC(DivisionC ontext ctx): Term
+visitBracketC(BracketC ontext ctx): Term
+visitIDC(IDC ontext ctx): Term
+visitAdditionC(AdditionC ontext ctx): Term
+visitSubtractionC(SubtractionC ontext ctx): Term
+visitModuloC(ModuloC ontext ctx): Term
condition
expression
Term
ArrayDeclaration
- identifier: String
run(): List<TraceState>
ArrayDeclarationAssignment
- identifier: String
run(): List<TraceState>
ArrayElementAssignment
- identifier: String
run(): List<TraceState>
RoutineCall
- routineName: String
- arguments: List<Term>
run(): List<TraceState>
ReturnCommand
- identifier: String
run(): List<TraceState>
Declaration
- identifier: String
run(): List<TraceState>
DeclarationAssignment
- identifier: String
run(): List<TraceState>
Assignment
- identifier: String
run(): List<TraceState>
WhileCommand
run(): List<TraceState>
IfCommand
run(): List<TraceState>
IfElseCommand
run(): List<TraceState>
RoutineCommand
+ run(): List<TraceState>
+ setArgs(List<Term>)
Argumente holen aus altem Scope
Scopeändern
ausführen
Scope wegjagen und prüefen, ob korrekt returnt
RoutineCommand c = controller.getRoutineRoot(Conte
c.setArgs(arguments)
List<TraceState> l = c.run()
scope.setReturnValue(controller.getReturnValue())
return l;
setzt routineFinished auf true
controller.setReturnValue(Term.evaluate())
«enum»
Type
NULL
INT
LONG
FLOAT
DOUBLE
CHAR
BOOLEAN
ARRAY
dimensions
1..3
value
1
indices
1...3
dimensions
1..3
value
1
condition
condition
condition
condition
NotCondition
+NotCondition(Condition operand)
+evaluate(List<TraceState> states): BooleanValue
+evaluate(Scope currentScope): BooleanValue
BinaryCondition
+evaluate(List<TraceState> states): BooleanValue
+evaluate(Scope currentScope): BooleanValue
OrCondition
+OrCondition(Condition leftOperand, Condition rightOperand)
+evaluate(List<TraceState> states): BooleanValue
+evaluate(Scope currentScope): BooleanValue
AndCondition
+AndCondition(Condition leftOperand, Condition rightOperand)
+evaluate(List<TraceState> states): BooleanValue
+evaluate(Scope currentScope): BooleanValue
Term
+evaluate(List<TraceState> states): TermValue
+evaluate(Scope currentScope): TermValue
+evaluateToString(List<TraceState> states): String
children
left, right (children)
child
1
child
1
children
2
Comparison
+evaluate(List<TraceState> states): BooleanValue
+evaluate(Scope currentScope): BooleanValue
MoreEqualComparison
+evaluate(List<TraceState> states): BooleanValue
+evaluate(Scope currentScope): BooleanValue
LessEqualComparison
+evaluate(List<TraceState> states): BooleanValue
+evaluate(Scope currentScope): BooleanValue
MoreComparison
+evaluate(List<TraceState> states): BooleanValue
+evaluate(Scope currentScope): BooleanValue
LessComparison
+evaluate(List<TraceState> states): BooleanValue
+evaluate(Scope currentScope): BooleanValue
EqualComparison
+evaluate(List<TraceState> states): BooleanValue
+evaluate(Scope currentScope): BooleanValue
NotEqualComparison
+evaluate(List<TraceState> states): BooleanValue
+evaluate(Scope currentScope): BooleanValue
indices
1..2
indices
1..2
ArrayAccessRelationalTerm
-identifier: String
+evaluate(List<TraceState> states): TermValue
+evaluate(Scope currentScope): TermValue
ArrayAccessTerm
-identifier: String
+evaluate(List<TraceState> states): TermValue
+evaluate(Scope currentScope): TermValue
VariableRelationalTerm
-identifier: String
+VariableTerm(String identifier)
+evaluate(List<TraceState> states): TermValue
+evaluate(Scope currentScope): TermValue
VariableTerm
-identifier: String
+VariableTerm(String identifier)
+evaluate(List<TraceState> states): TermValue
+evaluate(Scope currentScope): TermValue
VariableTerm("x") wird bei der TraceErzeugung
und VariableRelationalTerm ("A.x") wird für Wert
ConstantTerm
+ConstantTerm(TermValue value)
+evaluate(List<TraceState> states): TermValue
+evaluate(Scope currentScope): TermValue
value
1
«interface»
TermValue
+add(TermValue operand): TermValue
+mult(TermValue operand): TermValue
+div(TermValue operand): TermValue
+sub(TermValue operand): TermValue
+add(TermValue operand): TermValue
+moreEqual(TermValue operand): boolean
+more(TermValue operand): boolean
+lessEqual(TermValue operand): boolean
+less(TermValue operand): boolean
+equal(TermValue operand): boolean
+toString(): String
NegativeTerm
+evaluate(List<TraceState> states): TermValue
+evaluate(Scope currentScope): TermValue
BinaryArithmeticTerm
+evaluate(List<TraceState> states): TermValue
+evaluate(Scope currentScope): TermValue
AdditionTerm
+AdditionTerm(Term left, Term right)
+evaluate(List<TraceState> states): TermValue
+evaluate(Scope currentScope): TermValue
SubtractionTerm
+evaluate(List<TraceState> states): TermValue
+evaluate(Scope currentScope): TermValue
MultiplicationTerm
+evaluate(List<TraceState> states): TermValue
+evaluate(Scope currentScope): TermValue
DivisionTerm
+evaluate(List<TraceState> states): TermValue
+evaluate(Scope currentScope): TermValue
ModuloTerm
+evaluate(List<TraceState> states): TermValue
+evaluate(Scope currentScope): TermValue
Exceptions
«interface»
DBuggerLogicExceptions
+getDE(): String
+getOccurence(): List<Stri
java.lang.Exception
VariableNotFoundException
RoutineNotFoundException
MissingReturnCallExcepti
InvalidProgramExcepti
WrongArgumentException
ReturnTypeException
WrongTypeAssignmentExcepti
AlreadyDeclaredExcepti
WrongTypeArgumentException
WrongNumberArgumentException
FloatValue
- value: float
- TYPE = FLOAT :Type
+getValue(): float
+add(TermValue operand): TermValue
+mult(TermValue operand): TermValue
+div(TermValue operand): TermValue
+sub(TermValue operand): TermValue
+sub(TermValue operand): TermValue
+moreEqual(TermValue operand): boolean
+more(TermValue operand): boolean
+lessEqual(TermValue operand): boolean
+less(TermValue operand): boolean
+equal(TermValue operand): boolean
+toString(): String
IntValue
- value: int
- TYPE = INT :Type
+getValue(): int
+add(TermValue operand): TermValue
+mult(TermValue operand): TermValue
+div(TermValue operand): TermValue
+sub(TermValue operand): TermValue
+modulo(TermValue operand): TermValue
+moreEqual(TermValue operand): boolean
+more(TermValue operand): boolean
+lessEqual(TermValue operand): boolean
+less(TermValue operand): boolean
+equal(TermValue operand): boolean
+toString(): String
LongValue
- value: long
- TYPE = LONG :Type
+getValue(): long
+add(TermValue operand): TermValue
+mult(TermValue operand): TermValue
+div(TermValue operand): TermValue
+sub(TermValue operand): TermValue
+sub(TermValue operand): TermValue
+moreEqual(TermValue operand): boolean
+more(TermValue operand): boolean
+lessEqual(TermValue operand): boolean
+less(TermValue operand): boolean
+equal(TermValue operand): boolean
+toString(): String
DoubleValue
- value: double
- TYPE = DOUBLE :Type
+getValue(): double
+add(TermValue operand): TermValue
+mult(TermValue operand): TermValue
+div(TermValue operand): TermValue
+sub(TermValue operand): TermValue
+sub(TermValue operand): TermValue
+moreEqual(TermValue operand): boolean
+more(TermValue operand): boolean
+lessEqual(TermValue operand): boolean
+less(TermValue operand): boolean
+equal(TermValue operand): boolean
+toString(): String
CharValue
- value: char
- TYPE = CHAR :Type
+getValue(): char
+add(TermValue operand): TermValue
+mult(TermValue operand): TermValue
+div(TermValue operand): TermValue
+sub(TermValue operand): TermValue
+sub(TermValue operand): TermValue
+moreEqual(TermValue operand): boolean
+more(TermValue operand): boolean
+lessEqual(TermValue operand): boolean
+less(TermValue operand): boolean
+equal(TermValue operand): boolean
+toString(): String
BooleanValue
- value: boolean
- TYPE = BOOLEAN :Type
+getValue(): boolean
+add(TermValue operand): TermValue
+mult(TermValue operand): TermValue
+div(TermValue operand): TermValue
+sub(TermValue operand): TermValue
+sub(TermValue operand): TermValue
+moreEqual(TermValue operand): boolean
+more(TermValue operand): boolean
+lessEqual(TermValue operand): boolean
+less(TermValue operand): boolean
+equal(TermValue operand): boolean
+toString(): String
ArrayValue
- value: TermValue[][][]
- TYPE = ARRAY :Type
+getValue(): TermValue[][][]
+add(TermValue operand): TermValue
+mult(TermValue operand): TermValue
+div(TermValue operand): TermValue
+sub(TermValue operand): TermValue
+sub(TermValue operand): TermValue
+moreEqual(TermValue operand): boolean
+more(TermValue operand): boolean
+lessEqual(TermValue operand): boolean
+less(TermValue operand): boolean
+equal(TermValue operand): boolean
+toString(): String