

```

Syntax
...

SYNTAX #Id ::= object
      | null
SYNTAX Variable ::= #Int
SYNTAX VariableName ::= v #Int
SYNTAX Selector ::= <init>()#Int
SYNTAX TypeName ::= #Id
      | #Id / TypeName
SYNTAX TypeReference ::= < #Id , TypeName >
SYNTAX FieldReference ::= < #Id , TypeName , #Id , TypeReference >
SYNTAX MethodReference ::= < #Id , TypeName , Selector >
SYNTAX Params ::= List(Variable, " " )
SYNTAX NewInstructionBase ::= Variable =new TypeReference @ #Int
SYNTAX NewInstruction ::= NewInstructionBase ( Params )
      | NewInstructionBase ( Params )
SYNTAX GetInstruction ::= Variable =getField FieldReference Variable
      | Variable =getstatic FieldReference
SYNTAX PutInstruction ::= putfield Variable = Variable FieldReference
      | putstatic Variable FieldReference
SYNTAX PhiInstruction ::= Variable =phi( Params )
SYNTAX PhiPhiInstruction ::= Variable =phiphi( Params )
SYNTAX InvokeSpecialInstruction ::= invokespecial MethodReference Params @ #Int exception: Variable
SYNTAX Instruction ::= NewInstruction
      | GetInstruction
      | PutInstruction
      | PhiInstruction
      | PhiPhiInstruction
      | return
      | InvokeSpecialInstruction
      | noinstruction
      | main

SYNTAX BBEdge ::= #Id -> #Id ;
SYNTAX BlockBody ::= List(Instruction, " " )
SYNTAX Block ::= #Id : { BlockBody }
SYNTAX TaskUnit ::= BBEdge
      | Block
      | start
      | analysis
      | done

SYNTAX Task ::= TaskUnit
      | Task Task

END MODULE

```

MODULE KWALA

IMPORTS KWALA-SYNTAX

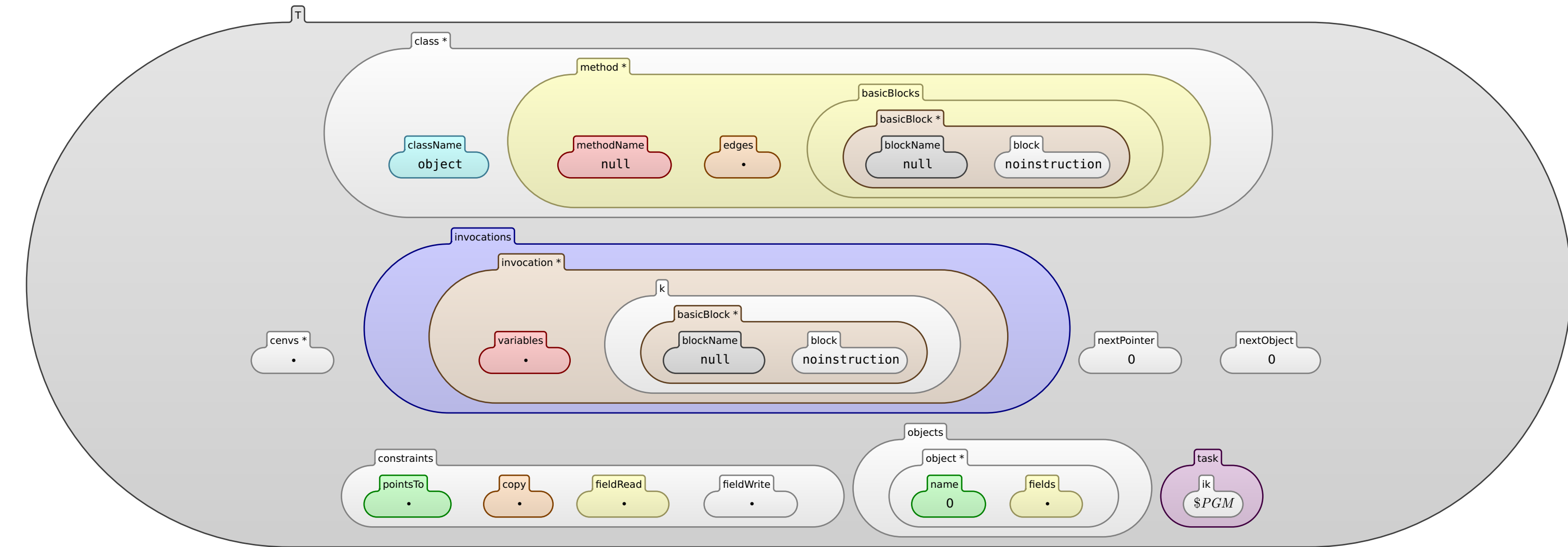
```

Semantics
...

Configuration
...

```

CONFIGURATION:



```

Processing Basic Blocks
...

```

RULE $T_1 \quad T_2 \Rightarrow T_1 \curvearrowright T_2$ RULE $I_1 ; BBl_2 \Rightarrow I_1 \curvearrowright BBl_2$ SYNTAX $ListItem ::= [\#Id , \#Id]$

RULE $\frac{ik}{BB_1 \rightarrow BB_2} ; \{ BB_1, BB_2 \}$

RULE $\frac{ik}{BB : \{ Bl \}}$

RULE $\frac{ik}{BB : \{ \}}$

RULE $\frac{ik}{start}$

RULE $\frac{ik}{analysis}$

```

Gathering Constraints
...

```

SYNTAX $K ::= wrappedList(Bag)$

```

Phi functions

```

RULE $\frac{invocation}{\frac{block}{V_i = \phi(P)} \quad \frac{variables}{V_i = \phi(P)} \quad \frac{nextPointer}{NP \rightarrow NP + Int \ 1}}$

RULE $\frac{invocation}{\frac{block}{V_i = \phi(V_2 , P)} \quad \frac{variables}{V_i = \phi(V_2 , P)} \quad \frac{copy}{P_2 \mapsto wrappedList(\frac{censvs}{P_i} RestSet)}}$ when $\neg_{Bool} P_1$ in $RestSet$

RULE $\frac{invocation}{\frac{block}{V_i = \phi(V_2 , P)} \quad \frac{variables}{V_i = \phi(V_2 , P)} \quad \frac{copy}{P_2 \mapsto wrappedList(\frac{censvs}{P_i} Rest)}}$ when $\neg_{Bool} P_2$ in keys $Rest$

RULE $\frac{invocation}{V_i = \phi()}$

```

Get field

```

SYNTAX $SetItem ::= (FieldReference |> \#Int)$

RULE $\frac{invocation}{\frac{block}{V_i = getField \ F \ V_2} \quad \frac{variables}{V_i = getField \ F \ V_2} \quad \frac{nextPointer}{NP \rightarrow NP + Int \ 1} \quad \frac{fieldRead}{P_2 \mapsto wrappedList(\frac{censvs}{(F |> NP)} Rest)}}$ when $\neg_{Bool} P_2$ in keys $Rest$

RULE $\frac{invocation}{\frac{block}{V_i = getField \ F \ V_2} \quad \frac{variables}{V_i = getField \ F \ V_2} \quad \frac{nextPointer}{NP \rightarrow NP + Int \ 1} \quad \frac{fieldRead}{P_2 \mapsto wrappedList(\frac{censvs}{(F |> NP)} RestSet)}}$

```

Put field

```

RULE $\frac{invocation}{\frac{block}{putfield \ V_i = V_2 \ F} \quad \frac{variables}{V_i = V_2 \ F} \quad \frac{fieldWrite}{P_1 \mapsto wrappedList(\frac{censvs}{(F |> P_2)} Rest)} \quad \frac{pointsTo}{P_1 \mapsto wrappedList(\frac{censvs}{O})} \quad \frac{object}{\frac{name}{O} \quad \frac{fields}{F \mapsto NP} \quad \frac{nextPointer}{NP \rightarrow NP + Int \ 1}}$ when \neg_{Bool}

 P_1 in keys $Rest$

RULE $\frac{invocation}{\frac{block}{putfield \ V_i = V_2 \ F} \quad \frac{variables}{V_i = V_2 \ F} \quad \frac{fieldWrite}{P_1 \mapsto wrappedList(\frac{censvs}{(F |> P_2)} RestSet)} \quad \frac{pointsTo}{P_1 \mapsto wrappedList(\frac{censvs}{O})} \quad \frac{object}{\frac{name}{O} \quad \frac{fields}{F \mapsto NP} \quad \frac{nextPointer}{NP \rightarrow NP + Int \ 1}}$

```

New instruction

```

RULE $\frac{invocation}{\frac{block}{V_i = new \ TR \ @} \quad \frac{variables}{V_i = new \ TR \ @} \quad \frac{nextPointer}{NP \rightarrow NP + Int \ 1} \quad \frac{nextObject}{NO \rightarrow NO + Int \ 1} \quad \frac{pointsTo}{NP \mapsto wrappedList(\frac{censvs}{NO})} \quad \frac{objects}{\frac{object}{\frac{name}{NO} \quad \frac{fields}{\cdot}}}}$

END MODULE