We take the common nonterminals *digit*, *digits*, *lower*, *upper*, *alphanum*, and *alphanums* to be predefined with their obvious meaning (plurality such as digits implies at least one digit). Then the grammar for the language is:

• *Classs may extend another class or default to extending Object*
⟨class⟩ ⇒
    class ⟨class id⟩⟨extend⟩⟨scope⟩⟨class decls⟩
⟨extend⟩ ⇒
    ϵ
  | **extends** ⟨class id⟩

• *A class can be split up into five sections – private protected public refinements and main*
⟨class decls⟩ ⇒
    ⟨class decl⟩
  | ⟨class decls⟩⟨class decl⟩
⟨class decl⟩ ⇒
    ⟨refinement⟩
  | ⟨access group⟩
  | ⟨main⟩

• *Refinements are named method dot refinement*
⟨refinement⟩ ⇒
    refinement ⟨scope⟩⟨refines⟩
⟨refines⟩ ⇒
    ⟨refine⟩
  | ⟨refines⟩⟨refine⟩
⟨refine⟩ ⇒
    ⟨refine sig⟩⟨scope⟩⟨statements⟩
⟨refine sig⟩ ⇒
    ⟨return type⟩⟨var id⟩.⟨var id⟩⟨params⟩

• *Access groups contain all the members of a class*
⟨access group⟩ ⇒
    ⟨access type⟩⟨scope⟩⟨members⟩
⟨access type⟩ ⇒
    private
  | **protected**
  | **public**
⟨members⟩ ⇒
    ⟨member⟩
  | ⟨members⟩⟨member⟩
⟨member⟩ ⇒

⟨ivar⟩
| ⟨method⟩
| ⟨init⟩

⟨ivar⟩ ⇒
⟨var decl⟩

⟨method⟩ ⇒
⟨method sig⟩⟨scope⟩⟨statements⟩

⟨method sig⟩ ⇒
⟨return type⟩⟨var id⟩
| ⟨params⟩

⟨init⟩ ⇒
⟨init sig⟩⟨scope⟩⟨statements⟩

⟨init sig⟩ ⇒
init ⟨params⟩

• *The main is special – it belongs to no object and is what can be used to start an execution*

⟨main⟩ ⇒
⟨main sig⟩⟨scope⟩⟨statements⟩

⟨main sig⟩ ⇒
main
| **main (String[]** ⟨var id⟩ **)**

• *Finally the meat and potatoes*

⟨statements⟩ ⇒
⟨statement⟩
| ⟨statements⟩⟨statement⟩

⟨statement⟩ ⇒
⟨local⟩
| ⟨refine stmt⟩
| ⟨assignment⟩
| ⟨conditional⟩
| ⟨loop⟩
| ⟨expression⟩

• *Local variables can be uninitialized (end up null) or initialized*

⟨local⟩ ⇒
⟨var decl⟩
| ⟨var decl⟩ **:=** ⟨expression⟩

• *Refine statements need not return type information*

⟨refine stmt⟩ ⇒

refine ⟨specialize⟩

• *Assignment works as in most languages – lvalues receive the results of expressions*
⟨assignment⟩ ⇒
⟨lvalue⟩ **:=** ⟨expression⟩
⟨lvalue⟩ ⇒
⟨var id⟩
| **this .** ⟨var id⟩
| ⟨var id⟩ **.** ⟨var id⟩

• *If and else are the same as in every other language*
⟨conditional⟩ ⇒
⟨if⟩
| ⟨if⟩⟨else⟩
⟨if⟩ ⇒
if (⟨expression⟩ **)**⟨scope⟩⟨statements⟩
⟨else⟩ ⇒
else⟨scope⟩

• *C and java style loop constructs*
⟨loop⟩ ⇒
⟨loop head⟩⟨scope⟩⟨statements⟩
⟨loop head⟩ ⇒
for (⟨assigment⟩ **,** ⟨expression⟩ **,** ⟨assigment⟩ **)**
| **while (** ⟨expression⟩ **)**

• *An expression is anything that can result in a value – note that assignment is not an expression (should we change?)*
⟨expression⟩ ⇒
⟨invocation⟩
| ⟨arithmetic⟩
| ⟨array expression⟩
| ⟨test⟩
| ⟨instantiate⟩
| ⟨refine expr⟩
| ⟨literal⟩
| ⟨var id⟩
| **(** ⟨expression⟩ **)**
| **null**

• *Method invocations always have a receiver*

⟨invocation⟩ ⇒
    ⟨receiver⟩ **.** ⟨invoke⟩
⟨receiver⟩ ⇒
    this
  | ⟨var id⟩
⟨invoke⟩ ⇒
    ⟨var id⟩ **()**
  | ⟨var id⟩ **(** ⟨args⟩ **)**

• *Basic arithmetic can and will be done!*
⟨arithmetic⟩ ⇒
    ⟨expression⟩⟨bin op⟩⟨expression⟩
  | ⟨unary op⟩⟨expression⟩
⟨bin op⟩ ⇒
    +
  | **-**
  | ***
  | /
  | %
⟨unary op⟩ ⇒
    -

• *Build an array on the fly (should we leave this out? TODO – ARRAY ASSIGNMENT)*
⟨array expression⟩ ⇒
    [⟨args⟩]

• *Common boolean predicates*
⟨test⟩ ⇒
    ⟨expression⟩⟨bin pred⟩⟨expression⟩
  | ⟨unary pred⟩⟨expression⟩
  | **refinable (** ⟨var id⟩ **)**
⟨bin pred⟩ ⇒
    and
  | **or**
  | **xor**
  | <
  | <=
  | =
  | **!=**
  | >=
  | >
⟨unary pred⟩ ⇒

!

• *Making something*
⟨instantiate⟩ ⇒
    ⟨object instantiate⟩
  | ⟨array instantiate⟩
⟨object instantiate⟩ ⇒
    new ⟨class id⟩
  | **new** ⟨class id⟩ **(** ⟨args⟩ **)**
⟨array instantiate⟩ ⇒
    new ⟨type⟩ [⟨digits⟩]

• *Refinement takes a specific specialization and notes the required return type*
⟨refine expr⟩ ⇒
    refine ⟨specialize⟩ **to** ⟨type⟩
⟨specialize⟩ ⇒
    ⟨var id⟩ **()**
  | ⟨var id⟩ **(** ⟨args⟩ **)**

• *Literally necessary*
⟨literal⟩ ⇒
    ⟨int lit⟩
  | ⟨bool lit⟩
  | ⟨float lit⟩
  | ⟨string lit⟩
⟨float lit⟩ ⇒
    ⟨digits⟩ **.** ⟨digits⟩
⟨int lit⟩ ⇒
    ⟨sign⟩⟨digits⟩
⟨bool lit⟩ ⇒
    true
  | **false**
⟨string lit⟩ ⇒
    “⟨string escape seq⟩”

• *Params and args are as expected*
⟨params⟩ ⇒
    ϵ
  | **(** ⟨paramlist⟩ **)**
⟨paramlist⟩ ⇒
    ⟨var decl⟩
  | ⟨paramlist⟩ **,** ⟨var decl⟩

⟨args⟩ ⇒
    ⟨expression⟩
  | ⟨args⟩ **,** ⟨expression⟩

• *All the basic stuff we've been saving up until now*
⟨var decl⟩ ⇒
    ⟨type⟩⟨var id⟩
⟨return type⟩ ⇒
    unit
  | ⟨type⟩
⟨type⟩ ⇒
    ⟨class id⟩
  | ⟨type⟩[]
⟨class id⟩ ⇒
    ⟨upper⟩
  | ⟨upper⟩⟨alphanums⟩
⟨var id⟩ ⇒
    ⟨lower⟩
  | ⟨lower⟩⟨alphanums⟩