• *Classs may extend another class or default to extending Object*
⟨class⟩ ⇒
    class ⟨class id⟩⟨extend⟩ **{** ⟨class section⟩* **}**
⟨extend⟩ ⇒
    ϵ
  | **extends** ⟨class id⟩

• *Sections – private protected public refinements and main*
⟨class section⟩ ⇒
    ⟨refinement⟩
  | ⟨access group⟩
  | ⟨main⟩

• *Refinements are named method dot refinement*
⟨refinement⟩ ⇒
    refinement { ⟨refine⟩* **}**
⟨refine⟩ ⇒
    ⟨return type⟩⟨var id⟩**.**⟨var id⟩⟨params⟩ **{** ⟨statement⟩* **}**

• *Access groups contain all the members of a class*
⟨access group⟩ ⇒
    ⟨access type⟩ **{** ⟨member⟩* **}**
⟨access type⟩ ⇒
    private
  | **protected**
  | **public**
⟨member⟩ ⇒
    ⟨var decl⟩
  | ⟨method⟩
  | ⟨init⟩
⟨method⟩ ⇒
    ⟨return type⟩⟨var id⟩⟨params⟩ **{** ⟨statement⟩* **}**
⟨init⟩ ⇒
    init ⟨params⟩ **{** ⟨statement⟩* **}**

• *Main is special – not instance data starts execution*
⟨main⟩ ⇒
    main (String[] ⟨var id⟩ **) {** ⟨statement⟩* **}**

• *Finally the meat and potatoes*
⟨statement⟩ ⇒
    ⟨var decl⟩ **;**

| ⟨assignment⟩ **;**
| ⟨conditional⟩
| ⟨loop⟩
| ⟨expression⟩ **;**

• *Accessing a variable possibly by index*
⟨lpiece⟩ ⇒
⟨var id⟩
| ⟨lpiece⟩ **[** ⟨expression⟩ **]**

• *Accessing a member's member's member etc etc*
⟨lvalue⟩ ⇒
⟨lpiece⟩
| ⟨lvalue⟩ **.** ⟨lpiece⟩

• *Assignment – lvalues receive the results of expressions*
⟨assignment⟩ ⇒
⟨lvalue⟩ **:=** ⟨expression⟩

• *Basic control structures*
⟨conditional⟩ ⇒
if (⟨expression⟩ **) {** ⟨statement⟩* **} else {** ⟨statement⟩* **}**
⟨loop⟩ ⇒
while (⟨expression⟩ **) {** ⟨statement⟩* **}**

• *Anything that can result in a value*
⟨expression⟩ ⇒
⟨invocation⟩
| ⟨field⟩
| ⟨variable⟩
| ⟨arithmetic⟩
| ⟨test⟩
| ⟨instantiate⟩
| ⟨refine expr⟩
| ⟨literal⟩
| **(** ⟨expression⟩ **)**
| **null**

• *Method invocations always have a receiver*
⟨invocation⟩ ⇒
⟨expression⟩ **.** ⟨invoke⟩
⟨invoke⟩ ⇒

⟨var id⟩ **()**
| ⟨var id⟩ **(** ⟨args⟩ **)**

• *Field of some foreign object (or this)*
⟨field⟩ ⇒
⟨expression⟩ **.** ⟨variable⟩

• *Variable values can be indexed or not*
⟨variable⟩ ⇒
⟨var id⟩
| ⟨variable⟩ **[** ⟨expression⟩ **]**

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

• *Common boolean predicates*
⟨test⟩ ⇒
⟨expression⟩⟨bin pred⟩⟨expression⟩
| ⟨unary pred⟩⟨expression⟩
| **refinable (** ⟨var id⟩ **)**
⟨bin pred⟩ ⇒
and
| **or**
| **xor**
| **nand**
| **nor**
| **<**
| **<=**
| **=**
| **!=**
| **>=**
| **>**
⟨unary pred⟩ ⇒

!

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

• *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⟩ ⇒
    (⟨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⟩