cd ::= class C extends D { $\overline{fd}$; kd $\overline{md}$ } class
fd ::= q C f field
kd ::= q C ( t C g, t C f ) { super(g); this.f = f; } constructor
md ::= t C m ( t C this, t C x ) { $\overline{t\ C\ y}$ s; return z } instance method
e ::= x | x.f expression
s ::= x = e | x.f = y | x = y.m(z) | super(g) | this(g) | x = new t() | s;s statement
t ::= k q qualifier type
k ::= initialized | underinitialized | unknowninitialized | fbcbottom
initializatioin qualifier
q ::= readonly | mutable | polymutable | receiverdependantmutable | immutable | bottom
immutability qualifier

**Qualifier Hierarchy**

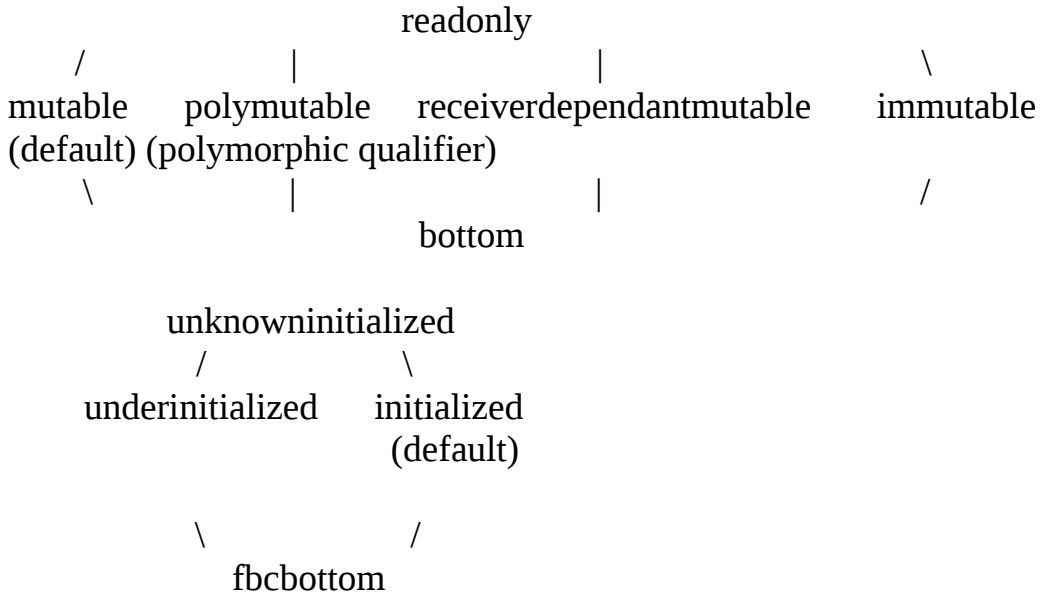


**Figure 1 Combination of qualifiers.** Two qualifier hierarchies are orthogonal. If an object is under initialization, its immutability guarantee is not satisfied. So even immutable and polyimmutable objects can also be modifed when under initialization. We don't have readonly objects, so there is no need to initialize readonly objects. Therefore, readonly doesn't have such exception when under initialization.

**Subtype relations**

$k_1\ q_1 <: k_2\ q_2 <=> k_1 <: k_2 \wedge q_1 <: q_2$

**Helper Functions**

$$\frac{q\ C\ f}{fType(f) = q}$$

*Note:*
*1)* No initialization modifier on field declarations. In actual implementation, to have circular initialization, *@NotOnlyInitialized* can be used on field declaration. However, it doesn't belong to initialization qualifier hierarchy.
2) The field is unique within the whole type hierarchy

fields(C) returns all fields directly declared in C.
cBody(kd) returns constructor body of kd.
mBody(md) returns method body of md.
isStatic(f) returns true if field f is static.
isStatic(md) returns true if method md is static.

**Viewpoint Adaptation Rules**

_ ▷ mutable = mutable
_ ▷ readonly = readonly
_ ▷ immutable = immutable
_ ▷ bottom = bottom
_ ▷ polymutable = polymutable
q ▷ receiverdependantmutable = q

**Special Rules**

- Forbid polymutable fields; readonly or polymutable constructor return type and readonly instantiation of objects
- In constructor, $q_{this} = q_{ret}$
- Forbid initialization modifier on fields, constructor return type and new statement
- Forbid bottom except on (implicit/explicit) lower bounds and null literal.

**Typing Rules**

$$\frac{x \in \Gamma}{\Gamma \vdash x : \Gamma(x)} \text{ (T-VAR)}$$

$$\frac{\Gamma(x) = k_x\, q_x \quad \text{fType}(f) = q_f \quad q = q_x \triangleright q_f \qquad k = \begin{cases} \text{initialized} & \textit{if } k_x = \textit{initialized} \\ \text{unknowninitialized} & \textit{otherwise} \end{cases}}{\Gamma \vdash x.f : k\, q} \text{ ( T-FLD )}$$

**Figure 2 Expression typing**

$$\frac{\Gamma \vdash e = t_e \quad t_e <: \Gamma(x)}{\Gamma \vdash x = e} \text{ ( T-VARASS )}$$

$$\frac{\begin{array}{l}\Gamma(x) = k_x\, q_x \quad \Gamma(y) = k_y\, q_y \quad \text{typeof}(f) = q_f \\ q_x = \text{mutable} \vee \\ q_f = \text{mutable} \vee \\ (k_x = \text{underinitialized} \wedge q_x = \text{immutable}) \vee \\ (k_x = \text{underinitialized} \wedge q_x = \text{polyimmutable}) \\ q_y <: q_x \triangleright q_f \\ k_x = \text{underinitialized} \vee k_y = \text{initialized}\end{array}}{\Gamma \vdash x.f = y} \text{ ( T-FLDASS)}$$

*__TODO__ Discuss: if $q_f$ = readonly, f also doesn't belong to abstract state of x. Should we allow reassigning this field? (I prefer not to.)

$\Gamma(x) = k_x\ q_x \quad \Gamma(y) = k_y\ q_y \quad \Gamma(\bar{z}) = \bar{k}_z\ \bar{q}_z \quad \text{typeof}(m) = k_{this}\ q_{this}, \bar{k}_p\ \bar{q}_p \rightarrow k_{ret}\ q_{ret}$

$k_y <: k_{this} \quad \bar{k}_z <: \bar{k}_p \quad k_{ret} <: k_x$

$q_y <: q_y \triangleright q_{this} \quad \bar{q}_z <: q_y \triangleright \bar{q}_p \quad q_y \triangleright q_{ret} <: q_x$

---

$\Gamma \vdash x = y.m(\bar{z})$ ( T-CALL )

$kd \text{ in } C \quad C <: D \quad \text{typeof}(D) = \bar{k}_{p\text{-}D}\ \bar{q}_{p\text{-}D} \rightarrow q_{ret\text{-}D} \quad \text{typeof}(kd) = \overline{\_\ \_} \rightarrow q_{ret\text{-}C}$

$q_{ret\text{-}C} =$
- $\_$ *if* $q_{ret\text{-}D} =$ receiverdependantmutable
- immutable *if* $q_{ret\text{-}D} =$ immutable
- mutable *if* $q_{ret\text{-}D} =$ mutable

$\Gamma(z) = k_z\ q_z \quad \bar{k}_z <: \bar{k}_{p\text{-}D} \quad \bar{q}_z <: q_{ret\text{-}C} \triangleright \bar{q}_{p\text{-}D}$

---

$\Gamma \vdash \text{super}(\bar{z}) \text{ in } kd$ ( T-SUPER )

* Previously, when $q_{ret\text{-}D} =$ mutable, $q_{ret\text{-}C}$ can still be immutable. Because at that time, immutable constructors only have immutable or polyimmutable(does not exist anymore), thus any mutable objet created locally cannot escape and be captured by outside objects; Neither outside mutable objects will be captured by the receiverdependantmutable field when invoking mutable super constructor in immutable constructor. But now, immutable and receiverdependantmutable constructors don't have such restrictions(mutable parameters are allowed in both cases) any more, so outside mutable objects can be captured by receiverdependantmutable field. If we allow calling mutable super() in immutable subclass constructor, when we use $this_{sub}$.rdmf to access the field, the result is not guarantee to be immutable(may be the mutable object assigned in super mutable constructor). Therefore, we don't allow this kinds of flexibility and require subclass and superclass constructors should have the exact same qualifier if $q_{ret\text{-}D} \neq$ receiverdependantmutable
* *Note:* In real Java code, one class can have multiple overloaded consturctors. One constructor can invoke the other by "this(..., ...)". The type rule T-THIS is very much the same as T-SUPER except that the constructor invoked by "this(..., ...)" comes from the same class.

$$\frac{\Gamma(x) = k_x\ q_x \quad \Gamma(\bar{y}) = \bar{k}_y\ \bar{q}_y \quad \text{typeof}(C) = \bar{k}_p\ \bar{q}_p \rightarrow q_{ret} \quad \bar{q}_y <: q \triangleright \bar{q}_P \quad q <: q \triangleright q_{ret} \quad q \neq \text{readonly} \quad \bar{k}_y <: \bar{k}_p \quad q <: q_x \quad k <: k_x \quad k = \begin{cases} \text{initialized} & \textit{if } \bar{k}_p = \textit{initialized} \\ \text{underinitialized} & \textit{otherwise} \end{cases}}{\Gamma \vdash x = \text{new } q\ C(\bar{y})} \text{ (T-NEW)}$$

$$\frac{\Gamma \vdash s_1 \quad \Gamma \vdash s_2}{\Gamma \vdash s_1;s_2} \text{ (T-SEQ )}$$

**Figure 3 Statement typing**

**Well-formdness Rules**

$$\frac{\text{fType}(f) \neq \text{polymutable} \quad C <: D \quad f \notin \text{fields}(D) \quad \text{isStatic}(f) \rightarrow \text{fType}(f) \neq \text{receiverdependantmutable}}{\vdash_C f \text{ is OK}} \text{ (WF-FLD)}$$

$$\frac{}{\vdash_{\text{object}} kd \text{ is OK}} \text{ (WF-CONS-OBJECT)}$$

$cBody(kd) = super(g);s \quad typeof(kd) = \overline{k_p}\ \overline{q_p} \rightarrow q_{ret}$
$q_{ret} \neq readonly \wedge q_{ret} \neq polymutable$
$\Gamma = (this : underinitialized\ q_{ret},\ \overline{p} : \overline{k_p}\ \overline{q_p},\ \overline{y} : \overline{k_{local}}\ \overline{q_{local}})$
$\Gamma \vdash super(\overline{y})\ in\ kd \quad \Gamma \vdash s$

(WF-CONS)

$\vdash_C kd\ is\ OK$

*Note*: $\vdash_{C\ kd}$ reads "constructor kd in class C is well-formed".

$mBody(md) = s;return\ z \quad typeof(md) = k_{this}\ q_{this},\ \overline{k_p}\ \overline{q_p} \rightarrow t_{ret}$
$\Gamma = (this : k_{this}\ q_{this},\ \overline{p} : \overline{k_p}\ \overline{q_p},\ \overline{y} : \overline{k_{local}}\ \overline{q_{local}}) \quad \Gamma \vdash s \quad \Gamma(z) <: t_{ret}$
$isStatic(md) \rightarrow q_{this} \neq receiverdependantmutable \wedge \overline{q_p} \neq receiverdependantmutable \wedge \overline{q_{ret}} \neq receiverdependantmutable$

(WF-METH)

$\vdash md\ is\ OK$

*TODO Write in formalisation: in static blocks, receiverdependantmutable is forbidden(like peer and rep are forbidden in static blocks)

$\vdash_C \overline{f}\ is\ OK \qquad \vdash_C kd\ is\ OK \qquad \vdash \overline{\overline{md}}\ is\ OK$

(WF-CLASS)

$\vdash C\ is\ OK$

**Figure 4 Well-formdness typing**