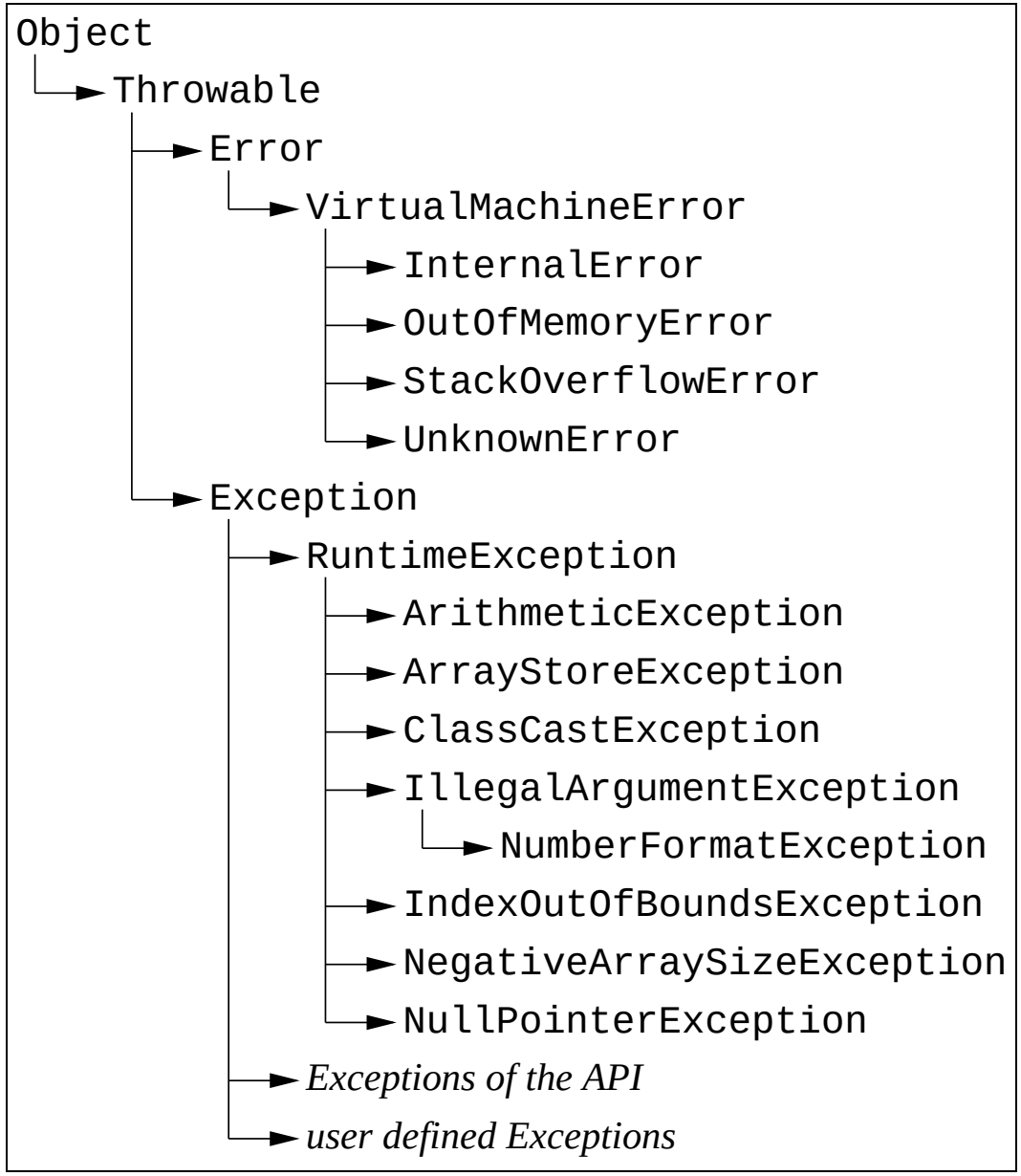
Errors and exceptions in Java$_{\mathcal{E}}$



Checked exceptions and `throws` clauses

Definition. A class E is called a checked exception class iff

- $E \preceq_{\mathrm{h}} \texttt{Throwable}$
- $E \not\preceq_{\mathrm{h}} \texttt{Error}$
- $E \not\preceq_{\mathrm{h}} \texttt{RuntimeException}$

Syntax of `throws` clauses:

$$meth(D_1\, x_1, \ldots, D_n\, x_n)\, \texttt{throws}\, E_1, \ldots, E_n\, body$$

Constraint: $E_i \preceq_{\mathrm{h}} \texttt{Throwable}$

Definition. A Method $msig$ throws more specific exceptions in A than in B, iff for each class E occurring in the `throws` clause of $msig$ in A there exists a class F in the `throws` clause of $msig$ in B such that $E \preceq_{\mathrm{h}} F$.

Constraint. If a method $A/msig$ directly overrides $B/msig$, then $msig$ throws more specific exceptions in A than in B.

Syntax of Java$_{\mathcal{E}}$

Stm ::= ... | `throw` *Exp*;
| `try` *Block*
`catch` (*Class*$_1$ *Loc*$_1$) *Block*$_1$ ... `catch` (*Class*$_n$ *Loc*$_n$) *Block*$_n$
| *Stm* `finally` *Block*

Definition. An exception E is allowed at a position α, iff one of the following conditions is true:

- $E \preceq_{\mathrm{h}}$ `Error`
- $E \preceq_{\mathrm{h}}$ `RuntimeException`
- The position α is in a `try` block and at least one `catch` clause of the `try` statement has a parameter of type F such that $E \preceq_{\mathrm{h}} F$
- The position α is in the body of a method or constructor declaration and there exists a class F in the `throws` clause of the declaration such that $E \preceq_{\mathrm{h}} F$

Type constraints for Java$_{\mathcal{E}}$

`catch` $(E\ loc)\ block$	$E \preceq_{\mathrm{h}}$ `Throwable`.
`throw` $^{\alpha} exp;$	$\mathcal{T}(\alpha) \preceq_{\mathrm{h}}$ `Throwable` and $\mathcal{T}(\alpha)$ is allowed at position α.
$^{\alpha}(exp_0.C/msig(exps))$	Each class E occurring in the `throws` clause of $msig$ in C is allowed at position α.
$^{\alpha}C.msig(exps)$	Each class E occurring in the `throws` clause of $msig$ in C is allowed at position α.

Vocabulary of the ASM for Java$_{\mathcal{E}}$

$$\begin{aligned} \mathit{Abr} = {} & \mathit{Break}(\mathit{Lab}) \mid \mathit{Continue}(\mathit{Lab}) \\ & \mid \mathit{Return} \mid \mathit{Return}(\mathit{Val}) \mid \mathit{Exc}(\mathit{Ref}) \end{aligned}$$

Transition rules for Java$_{\mathcal{E}}$

$$\begin{aligned} & \mathit{execJava}_E = \\ & \quad \mathit{execJavaExp}_E \\ & \quad \mathit{execJavaStm}_E \end{aligned}$$

$$\begin{aligned} \mathit{failUp}(\mathit{exc}) &= \mathit{yieldUp}(\texttt{throw new } \mathit{exc}();) \\ \mathit{fail}(\mathit{exc}) &= \mathit{yield}(\texttt{throw new } \mathit{exc}();) \end{aligned}$$

Transition rules for Java$_{\mathcal{E}}$ (Expressions)

$$
\begin{array}{l}
execJavaExp_E = \textbf{case } context(pos) \textbf{ of} \\
\quad {}^{\alpha}val_1\ bop\ {}^{\blacktriangleright}val_2 \quad \rightarrow \textbf{if } bop \in divMod \wedge isZero(val_2) \textbf{ then} \\
\qquad\qquad\qquad\qquad failUp(\texttt{ArithmeticException}) \\
\quad {}^{\blacktriangleright}ref.c/f \qquad\quad \rightarrow \textbf{if } ref = null \textbf{ then } failUp(\texttt{NullPointerException}) \\
\quad {}^{\alpha}ref.c/f = {}^{\blacktriangleright}val \rightarrow \textbf{if } ref = null \textbf{ then } failUp(\texttt{NullPointerException}) \\
\quad {}^{\alpha}ref.c/m{}^{\blacktriangleright}(vals) \rightarrow \textbf{if } ref = null \textbf{ then } failUp(\texttt{NullPointerException}) \\
\quad (c){}^{\blacktriangleright}ref \qquad\quad \rightarrow \textbf{if } ref \neq null \wedge classOf(ref) \not\leq c \textbf{ then} \\
\qquad\qquad\qquad\qquad failUp(\texttt{ClassCastException})
\end{array}
$$

Transition rules for Java$_{\mathcal{E}}$ (Statements)

$execJavaStm_E = \textbf{case}\ context(pos)\ \textbf{of}$

$\texttt{throw}\ {}^{\alpha}exp; \rightarrow pos := \alpha$

$\texttt{throw}\ {}^{\blacktriangleright}ref; \rightarrow \textbf{if}\ ref = null\ \textbf{then}\ failUp(\texttt{NullPointerException})$
$\qquad\qquad\qquad\qquad\ \textbf{else}\ yieldUp(Exc(ref))$

$\texttt{try}\ {}^{\alpha}stm\ \texttt{catch}\ \ldots \quad \rightarrow pos := \alpha$

$\texttt{try}\ {}^{\blacktriangleright}Norm\ \texttt{catch}\ \ldots \rightarrow yieldUp(Norm)$

$\texttt{try}\ {}^{\blacktriangleright}Exc(ref)\ \texttt{catch}\ (c_1\, x_1)\ {}^{\beta_1}stm_1 \ldots \texttt{catch}\ (c_n\, x_n)\ {}^{\beta_n}stm_n \rightarrow$
$\quad \textbf{if}\ \exists\, 1 \le j \le n : classOf(ref) \preceq_{\mathrm{h}} c_j\ \textbf{then}$
$\qquad \textbf{let}\ j = \min\{i \mid classOf(ref) \preceq_{\mathrm{h}} c_i\}$
$\qquad pos := \beta_j$
$\qquad locals := locals \oplus \{(x_j, ref)\}$
$\quad \textbf{else}\ yieldUp(Exc(ref))$

$\texttt{try}\ {}^{\blacktriangleright}abr\ \texttt{catch}\ (c_1\, x_1)\ {}^{\beta_1}stm_1 \ldots \texttt{catch}\ (c_n\, x_n)\ {}^{\beta_n}stm_n \rightarrow yieldUp(abr)$

$\texttt{try}\ {}^{\alpha}Exc(ref) \ldots \texttt{catch}\ (c_i\, x_i)\ {}^{\blacktriangleright}Norm \ldots \rightarrow yieldUp(Norm)$

$\texttt{try}\ {}^{\alpha}Exc(ref) \ldots \texttt{catch}\ (c_i\, x_i)\ {}^{\blacktriangleright}abr \ldots \quad \rightarrow yieldUp(abr)$

${}^{\alpha}stm_1\ \texttt{finally}\ {}^{\beta}stm_2 \rightarrow pos := \alpha$

${}^{\blacktriangleright}Norm\ \texttt{finally}\ {}^{\beta}stm \rightarrow pos := \beta$

${}^{\blacktriangleright}abr\ \texttt{finally}\ {}^{\beta}stm \rightarrow pos := \beta$

${}^{\alpha}s\ \texttt{finally}\ {}^{\blacktriangleright}Norm \rightarrow yieldUp(s)$

${}^{\alpha}s\ \texttt{finally}\ {}^{\blacktriangleright}abr \rightarrow yieldUp(abr)$

Transition rules for Java$_{\mathcal{E}}$ (Statements continued)

$execJavaStm_E$ = **case** $context(pos)$ **of**
  lab : $^{\blacktriangleright}Exc(ref)$ $\longrightarrow$ $yieldUp(Exc(ref))$
  `static` $^{\alpha}Exc(ref)$ $\longrightarrow$
    if $classOf(ref) \preceq_{\mathrm{h}}$ `Error` **then**
      $yieldUp(Exc(ref))$
    else
      $failUp($`ExceptionInInitializerError`$)$
  $Exc(ref)$ $\longrightarrow$ **if** $pos = firstPos \wedge \neg null(frames)$ **then**
      $exitMethod(Exc(ref))$
      if $methNm(meth) =$ `"<clinit>"` **then**
        $classState(className(meth)) := Unusable$

Transition rules for Java$_{\mathcal{E}}$ (continued)

$$
\begin{array}{l}
\mathit{initialize}(c) = \\
\quad \ldots \\
\quad \textbf{if } \mathit{classState}(c) = \mathit{Unusable} \textbf{ then} \\
\quad\quad \mathit{fail}(\texttt{NoClassDefFoundErr})
\end{array}
$$

$$
\begin{array}{l}
\mathit{propagatesAbr}(\mathit{phrase}) = \\
\quad \mathit{phrase} \neq \mathit{lab} : s \wedge \\
\quad \mathit{phrase} \neq \texttt{static}\ s \wedge \\
\quad \mathit{phrase} \neq \texttt{try} \ldots \wedge \\
\quad \mathit{phrase} \neq s_1\ \texttt{finally}\ s_2
\end{array}
$$