CS 365 — Programming Language Concepts

Going classless

Apr 2, 2008

JayO Concrete Syntax

Program	$\longrightarrow$	*Interfaces* void main () '{' *Declarations Statements* '}'
Interfaces	$\longrightarrow$	{ *Interface* }*
Interface	$\longrightarrow$	interface *Identifier* { *Signatures* }
Signatures	$\longrightarrow$	{ *Signature* }*
Signature	$\longrightarrow$	*Type Identifier* (*Parameters*) ;
Parameters	$\longrightarrow$	*Parameter* { , *Parameter* }
Parameter	$\longrightarrow$	*Type Identifier*
Declarations	$\longrightarrow$	{ *Declaration* }*
Declaration	$\longrightarrow$	*Type Identifiers;*
Type	$\longrightarrow$	int | boolean | *Identifier*
Identifiers	$\longrightarrow$	*Identifer* { , *Identifier* }*

JayO Concrete Syntax, continued

$$
\begin{array}{rcl}
\mathit{Statements} & \longrightarrow & \{\ \mathit{Statement}\ \}* \\
\mathit{Statement} & \longrightarrow & ;\ |\ \mathit{Block}\ |\ \mathit{Assignment}\ |\ \mathit{IfStatement}\ |\ \mathit{WhileStatement}\ | \\
 & & \mathit{PrintStatement}\ |\ \mathit{InvocationStatement}\ |\ \mathit{ReturnStatement} \\
\mathit{Block} & \longrightarrow & \text{'}\{\text{'}\ \mathit{Declarations}\ \mathit{Statements}\ \text{'}\}\text{'} \\
\mathit{Assignment} & \longrightarrow & \mathit{Identifier} = \mathit{Expression}\ ; \\
\mathit{IfStatement} & \longrightarrow & \mathsf{if}\ (\ \mathit{Expression}\ )\ \mathit{Statement}\ \{\ \mathsf{else}\ \mathit{Statement}\ \}_{opt} \\
\mathit{WhileStatement} & \longrightarrow & \mathsf{while}\ (\ \mathit{Experssion}\ )\ \mathit{Statement} \\
\mathit{PrintStatement} & \longrightarrow & \mathsf{System.out.println}\ (\ \mathit{Expression}\ )\ ; \\
\mathit{InvocationStatement} & \longrightarrow & \mathit{Invocation}\ ; \\
\mathit{ReturnStatement} & \longrightarrow & \mathsf{return}\ \mathit{Expression}\ ;
\end{array}
$$

JayO Concrete Syntax, continued

$$
\begin{array}{rcl}
\textit{Expression} & \longrightarrow & \textit{Conjunction}\ \{\ \texttt{||}\ \textit{Conjunction}\ \}* \\
\textit{Conjunction} & \longrightarrow & \textit{Relation}\ \{\ \texttt{\&\&}\ \textit{Relation}\ \}* \\
\textit{Relation} & \longrightarrow & \textit{Addition}\ \{\ [\ \texttt{|}\ <\ \texttt{|}\ <=\ \texttt{|}\ >\ \texttt{|}\ >=\ \texttt{|}\ ==\ \texttt{|}\ !\ =]\ \textit{Addition}\ \}_{opt} \\
\textit{Addition} & \longrightarrow & \textit{Term}\ \{\ [\ +\ \texttt{|}\ \texttt{-}]\ \textit{Term}\ \}* \\
\textit{Term} & \longrightarrow & \textit{Negation}\ \{\ [\ \texttt{'*'}\ \texttt{|}\ \texttt{/}]\ \textit{Negation}\ \}* \\
\textit{Negation} & \longrightarrow & \{\ !\ \}_{opt}\ \textit{Factor} \\
\textit{Factor} & \longrightarrow & \textit{Identifier}\ |\ \textit{Literal}\ |\ (\ \textit{Expression}\ )\ |\ \textit{Invocation}\ |\ \textit{Instantiation}\ |\ \underline{\textit{Clone}} \\
\textit{Invocation} & \longrightarrow & \textit{Identifier}\ .\ \textit{Identifier}\ (\ \textit{Arguments}\ ) \\
\textit{Arguments} & \longrightarrow & \textit{Expression}\ \{\ ,\ \textit{Expression}\ \}* \\
\textit{Instantiation} & \longrightarrow & \texttt{new}\ \textit{Identifier}\ ()\ \underline{\texttt{'\{'}\ \textit{Fields}\ \textit{Methods}\ \texttt{'\}'}} \\
\textit{Field} & \longrightarrow & \texttt{private}\ \textit{Type}\ \textit{Identifier}\ \underline{=\ \textit{Expression};} \\
\textit{Method} & \longrightarrow & \texttt{public}\ \textit{Type}\ \textit{Identifier}\ (\ [\ \textit{Parameters}\ ]\ )\ \underline{\textit{Block}} \\
\underline{\textit{Clone}} & \longrightarrow & \underline{\textit{Identifier}.\ \texttt{clone}\ ()}
\end{array}
$$

JayMU Concrete Syntax

Program	$\longrightarrow$	*Interfaces* void main () '{' *Declarations Statements* '}'
Interfaces	$\longrightarrow$	{ *Interface* }*
Interface	$\longrightarrow$	interface *Identifier* { *Signatures* }
Signatures	$\longrightarrow$	{ *Signature* }*
Signature	$\longrightarrow$	*Type Identifier* (*Parameters*) ;
Parameters	$\longrightarrow$	*Parameter* { , *Parameter* }
Parameter	$\longrightarrow$	*Type Identifier*
Declarations	$\longrightarrow$	{ *Declaration* }*
Declaration	$\longrightarrow$	*Type Identifiers;*
Type	$\longrightarrow$	int | boolean | *Identifier*
Identifiers	$\longrightarrow$	*Identifer* { , *Identifier* }*
Statements	$\longrightarrow$	{ *Statement* }*

JayMU Concrete Syntax, continued

Statement	$\longrightarrow$	; | *Block* | *Assignment* | *IfStatement* | *WhileStatement* | *PrintStatement* | *InvocationStatement* | *ReturnStatement* *MethodUpdate*
Block	$\longrightarrow$	'{' *Declarations Statements* '}'
Assignment	$\longrightarrow$	*Identifier* = *Expression* ;
IfStatement	$\longrightarrow$	if (*Expression*) *Statement* { else *Statement* }$_{opt}$
WhileStatement	$\longrightarrow$	while (*Experssion*) *Statement*
PrintStatement	$\longrightarrow$	System.out.println (*Expression*) ;
InvocationStatement	$\longrightarrow$	*Invocation* ;
ReturnStatement	$\longrightarrow$	return *Expression* ;
MethodUpdate	$\longrightarrow$	*Identifier* . *Identifier* = *Block*

JayMU Concrete Syntax, continued

$$
\begin{array}{rcl}
\textit{Expression} & \longrightarrow & \textit{Conjunction}\ \{\ \texttt{||}\ \textit{Conjunction}\ \}* \\
\textit{Conjunction} & \longrightarrow & \textit{Relation}\ \{\ \texttt{\&\&}\ \textit{Relation}\ \}* \\
\textit{Relation} & \longrightarrow & \textit{Addition}\ \{\ [\ |\ <\ |\ <=\ |\ >\ |\ >=\ |\ ==\ |\ !\ =]\ \textit{Addition}\ \}_{opt} \\
\textit{Addition} & \longrightarrow & \textit{Term}\ \{\ [\ +\ |\ \text{-}]\ \textit{Term}\ \}* \\
\textit{Term} & \longrightarrow & \textit{Negation}\ \{\ [\ \text{'*'}\ |\ /]\ \textit{Negation}\ \}* \\
\textit{Negation} & \longrightarrow & \{\ !\ \}_{opt}\ \textit{Factor} \\
\textit{Factor} & \longrightarrow & \textit{Identifier}\ |\ \textit{Literal}\ |\ (\ \textit{Expression}\ )\ |\ \textit{Invocation}\ |\ \textit{Instantiation}\ |\ \underline{\textit{Clone}} \\
\textit{Invocation} & \longrightarrow & \textit{Identifier}\ .\ \textit{Identifier}\ (\ \textit{Arguments}\ ) \\
\textit{Arguments} & \longrightarrow & \textit{Expression}\ \{\ ,\ \textit{Expression}\ \}* \\
\textit{Instantiation} & \longrightarrow & \texttt{new}\ \textit{Identifier}\ ()\ \text{'\{'}\ \textit{Methods}\ \text{'\}'} \\
\textit{Method} & \longrightarrow & \textit{Type}\ \textit{Identifier}\ (\ [\ \textit{Parameters}\ ]\ )\ \textit{Block} \\
\textit{Clone} & \longrightarrow & \textit{Identifier}.\ \texttt{clone}\ ()
\end{array}
$$

JayMU Abstract Syntax

Program	$\longrightarrow$	*Interface*$*$ *Block*
Interface	$\longrightarrow$	**Identifier** *Signature*$*$
Signature	$\longrightarrow$	**Type Identifier** *Parameter*$*$
Parameter	$\longrightarrow$	**Type Identifier**
Declaration	$\longrightarrow$	**Type Identifier**$*$
Statement	$\longrightarrow$	*Skip* | *Block* | *Assignment* | *Conditional* | *Loop* | *Print* | *InvkStmt* | *Return* | *Update*
Skip	$\longrightarrow$	
Block	$\longrightarrow$	*Declaration*$*$ *Statement*$*$
Assignment	$\longrightarrow$	**Identifier** *Expression*
Conditional	$\longrightarrow$	*Expression Statement Statement*
Loop	$\longrightarrow$	*Expression Statement*
Print	$\longrightarrow$	*Expression*
InvkStmt	$\longrightarrow$	*Invk*
Return	$\longrightarrow$	*Expression*
Update	$\longrightarrow$	**Identifier Identifer** *Block*

JayMU Abstract Syntax, continued

Expression	$\longrightarrow$	*Variable* | *IntLitExpr* | *BoolLitExpr* | *BinaryExpr* | *UnaryExpr* | *Invk* | *Instantiation* | *Clone*
Variable	$\longrightarrow$	**Identifier**
IntLitExpr	$\longrightarrow$	**IntLiteral**
BoolLitExpr	$\longrightarrow$	**BoolLiteral**
BinaryExpr	$\longrightarrow$	*Expression* **Operator** *Expression*
UnaryExpr	$\longrightarrow$	**Operator** *Expression*
Invk	$\longrightarrow$	**Identifier Identifier** *Expression*$*$
Instantiation	$\longrightarrow$	**Identifier** *Method*$*$
Clone	$\longrightarrow$	**Identifier**
Method	$\longrightarrow$	**Type Identifier** *Parameter*$*$ *Block*