CS 365 — Programming Language Concepts

Inheritance

Mar 31, 2008

IPOJay Concrete Syntax

Program	→	*Classes* void main () '{' *Declarations Statements* '}'
Classes	→	{ *Class* }*
Class	→	class *Identifier* extends *Identifier* '{' { *Field* }* { *Method* }* '}'
Field	→	private *Type Identifiers* ;
Method	→	public *Type Identifier* ([*Parameters*]) '{' *Declarations Statements* '}'
Parameters	→	*Parameter* { , *Parameter* }
Parameter	→	*Type Identifier*
Declarations	→	{ *Declaration* }*
Declaration	→	*Type Identifiers;*
Type	→	int | boolean | *Identifier*
Identifiers	→	*Identifer* { , *Identifier* }*

IPOJay Concrete Syntax, continued

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

IPOJay Concrete Syntax, continued

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

IPOJay Abstract Syntax

Program	$\longrightarrow$	*Class*$*$ *Block*
Class	$\longrightarrow$	**Identifier Identifier** *Declaration*$*$ *Method*$*$
Method	$\longrightarrow$	**Type Identifier** *Parameter*$*$ *Block*
Parameter	$\longrightarrow$	**Type Identifier**
Declaration	$\longrightarrow$	**Type Identifier**$*$

POJay Abstract Syntax, continued

Statement	$\longrightarrow$	*Skip* | *Block* | *Assignment* | *Conditional* | *Loop* | *Print* | *InvkStmt* | *Return*
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*

POJay Abstract Syntax, continued

Expression	$\longrightarrow$	*Variable* | *IntLitExpr* | *BoolLitExpr* | *BinaryExpr* | *UnaryExpr* | *Invk* | *Instantiation* | *Cast*
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**
Cast	$\longrightarrow$	**Identifier** *Expression*