CS 365 — Programming Language Concepts

The Visitor Pattern for Language System Implementation

Jan 30, 2008

Functional vs OO

```
interface Animal {
    String happyNoise();
    String excitedNoise();
}

class Dog implements Animal {
    String happyNoise() { return "pant pant"; }
    String excitedNoise() { return "bark"; }
}

class Cat implements Animal {
    String happyNoise() { return "purrrrr"; }
    String excitedNoise() { return "meow"; }
}
```

Functional vs OO

```
interface Animal {
    String happyNoise();
    String excitedNoise();
}

class Dog implements Animal {
    String happyNoise() { return "pant pant"; }
    String excitedNoise() { return "bark"; }
}

class Cat implements Animal {
    String happyNoise() { return "purrrrr"; }
    String excitedNoise() { return "meow"; }
}

class Chicken implements Animal {
    String happyNoise() { return "cluck cluck"; }
    String excitedNoise() { return "cockadoodledoo"; }
}
```

Functional vs OO

```
interface Animal {
    String happyNoise();
    String excitedNoise();
    String angryNoise();
}

class Dog implements Animal {
    String happyNoise() { return "pant pant"; }
    String excitedNoise() { return "bark"; }
    String angryNoise() { return "grrrrr"; }
}

class Cat implements Animal {
    String happyNoise() { return "purrrrr"; }
    String excitedNoise() { return "meow"; }
    String angryNoise() { return "hissss"; }
}

class Chicken implements Animal {
    String happyNoise() { return "cluck cluck"; }
    String excitedNoise() { return "cockadoodledoo"; }
    String angryNoise() { return "squaaaack"; }
}
```

Functional vs OO

```
datatype Animal = Dog | Cat ;

fun happyNoise(Dog) = "pant pant"
  | happyNoise(Cat) = "purrrr"

fun excitedNoise(Dog) = "bark"
  | excitedNoise(Cat) = "meow"
```

Functional vs OO

```
datatype Animal = Dog | Cat ;

fun happyNoise(Dog) = "pant pant"
  | happyNoise(Cat) = "purrrr"

fun excitedNoise(Dog) = "bark"
  | excitedNoise(Cat) = "meow"

fun angryNoise(Dog) = "grrrrr"
  | angryNoise(Cat) = "hisssss"
```

Functional vs OO

```
datatype Animal = Dog | Cat | Chicken;

fun happyNoise(Dog) = "pant pant"
  | happyNoise(Cat) = "purrrr"
  | happyNoise(Chicken) = "cluck cluck";

fun excitedNoise(Dog) = "bark"
  | excitedNoise(Cat) = "meow"
  | excitedNoise(Chicken) = "cockadoodledoo";

fun angryNoise(Dog) = "grrrrr"
  | angryNoise(Cat) = "hisssss"
  | angryNoise(Chicken) = "squaaaack";
```

Functional vs OO

	Dog	Cat	Chicken
happyNoise	pant pant	purrrr	cluck cluck
excitedNoise	bark	meow	cockadoodledoo
angryNoise	grrrrr	hisssss	squaaaaack

Jay

$$
\begin{array}{rcl}
\mathit{Program} & \rightarrow & \texttt{void main () '\{'}\ \mathit{Declarations}\ \mathit{Statements}\ \texttt{'\}'} \\
\mathit{Declarations} & \rightarrow & \{\ \mathit{Declaration}\ \}* \\
\mathit{Declaration} & \rightarrow & \mathit{Type}\ \mathit{Identifiers};\ \texttt{;} \\
\mathit{Type} & \rightarrow & \texttt{int} \mid \texttt{boolean} \\
\mathit{Identifiers} & \rightarrow & \mathit{Identifer}\ \{\ \texttt{,}\ \mathit{Identifier}\ \}* \\
\mathit{Statements} & \rightarrow & \{\ \mathit{Statement}\ \}* \\
\mathit{Statement} & \rightarrow & \texttt{;} \mid \mathit{Block} \mid \mathit{Assignment} \mid \mathit{IfStatement} \\
 & & \mid \mathit{WhileStatement} \mid \mathit{PrintStatement} \\
\mathit{Block} & \rightarrow & \texttt{'\{'}\ \mathit{Statements}\ \texttt{'\}'} \\
\mathit{Assignment} & \rightarrow & \mathit{Identifier} = \mathit{Expression}\ \texttt{;}
\end{array}
$$

Jay, continued

$$
\begin{array}{rcl}
\mathit{IfStatement} & \rightarrow & \texttt{if (}\ \mathit{Expression}\ \texttt{)}\ \mathit{Statement}\ \{\ \texttt{else}\ \mathit{Statement}\ \}_{opt} \\
\mathit{WhileStatement} & \rightarrow & \texttt{while (}\ \mathit{Experssion}\ \texttt{)}\ \mathit{Statement} \\
\mathit{PrintStatement} & \rightarrow & \texttt{System.out.println (}\ \mathit{Expression}\ \texttt{) ;} \\
\mathit{Expression} & \rightarrow & \mathit{Conjunction}\ \{\ \texttt{||}\ \mathit{Conjunction}\ \}\texttt{*} \\
\mathit{Conjunction} & \rightarrow & \mathit{Relation}\ \{\ \texttt{\&\&}\ \mathit{Relation}\ \}\texttt{*} \\
\mathit{Relation} & \rightarrow & \mathit{Addition}\ \{\ [\ |\ <\ |\ <=\ |\ >\ |\ >=\ |\ ==\ |\ !=]\ \mathit{Addition}\ \}_{opt} \\
\mathit{Addition} & \rightarrow & \mathit{Term}\ \{\ [\ +\ |\ \texttt{-}]\ \mathit{Term}\ \}\texttt{*} \\
\mathit{Term} & \rightarrow & \mathit{Negation}\ [\ \texttt{*}\ |\ /]\ \mathit{Negation} \\
\mathit{Negation} & \rightarrow & \{\ !\ \}_{opt}\ \mathit{Factor} \\
\mathit{Factor} & \rightarrow & \mathit{Identifier}\ |\ \mathit{Literal}\ |\ (\ \mathit{Expression}\ )
\end{array}
$$