8 Programming Paradigms

OH WOW! PARADIGM SHIFT!

Programming Paradigms
Granularity of Classification

The principal programming paradigms

"More is not better (or worse) than less, just different."

Programming Paradigms

Concepts, Techniques, and Models of Computer Programming

Peter van Roy and Seif Haridi

v1.08 © 2008 by Peter Van Roy

CS 381 • Spring 201

v1.08 © 2008 by Peter Van Roy
What is a Programming Paradigm?
Imperative Programming
Functional Programming
Logic Programming
Object-Oriented Programming
(Programming) Paradigm

Paradigm: A conceptual model underlying the theories and practice of a scientific subject (Oxford)

scientific subject = programming

Programming Paradigm: A conceptual model underlying the theories and practice of programming
Programming Paradigm

Programming is the process of creating programs

A program describes a particular *computation*

*Programming Paradigm*: A conceptual model underlying the theories and practice of computing

*Programming Paradigm*: A model of computation
Imperative Paradigm

Data are represented by a collection of state variables

Computation is a transformation of state (variables)

Formal definition of the imperative paradigm:

```
type State = [(Name, Val)]
type Computation = State -> State
```

Examples:
- Turing Machine
- Fortran, Pascal, C, Perl, ...
Imperative Programming Languages

Need two sublanguages:

1. Language of expressions to describe values to be stored in variables: `Expr`
2. Language of statements to describe state changes and control flow: `Stmt`

Semantics are given by two functions:

\[
\text{evalE} :: \text{Expr} \rightarrow \text{State} \rightarrow \text{Val} \\
\text{evalS} :: \text{Stmt} \rightarrow \text{Computation}
\]
Haskell Demo ...

Imp.hs
Functional Paradigm

Data are represented by values

Computation is a function

Formal definition of the functional paradigm:

\[
\text{type Computation} = \text{Val} \rightarrow \text{Val}
\]

Examples:
Lambda Calculus
Lisp, Scheme, ML, Haskell, ...
Functional Programming Languages

One language of expressions to describe values and functions: \( \text{Expr} \)

Semantics are given by a function:

\[
\text{eval} :: \text{Expr} \rightarrow \text{Val}
\]
Haskell Demo ...

FunStatScope.hs

FunRec.hs
Logic Paradigm

Data are represented by values & relations

Computation is a relation

Formal definition of the logic paradigm:

```haskell
type Computation = (Val, ..., Val)
```

Examples:
Predicate Calculus
Prolog, Mercury, Curry, Twelf, ..., (SQL)

Quel, Datalog
Logic Programming Languages

One language of relations to describe values and relations: \texttt{Rel}

Semantics are given by a function:

\texttt{eval :: Rel \rightarrow (Val, ..., Val)}
Object-Oriented Paradigm

*Data* are represented by a *collection of objects with state*

*Computation* is *evolution of objects* (through method calls)

Formal definition of the object-oriented paradigm:

```plaintext
type State   = [(Name,Val)]
type Methods = [(Name,State -> State)]

type Object   = (State,Methods)
type Objects  = [Object]
type Computation = Objects -> Objects
```

*Examples:*
  - Featherweight Java (& other Object Calculi)
  - Simula, Smalltalk, CLOS, C++, Java, C#, ...
Object-Oriented Programming Languages

Need two sublanguages for *expressions*, and *statements* (like for imperative languages), but:

The statement language needs constructs to:
(a) create objects (a group of state and methods)
(b) invoke methods (execute local state transformations)

Semantics are again given by two functions:

```
evalE :: Expr -> State -> Val
evals :: Stmt -> Computation
```
Haskell Demo ...

Obj.hs
## Comparison of Paradigms

<table>
<thead>
<tr>
<th>Paradigm</th>
<th>Computation Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>imperative</strong></td>
<td>... <em>state transformation</em> that changes an input state into an output state</td>
</tr>
<tr>
<td><strong>functional</strong></td>
<td>... <em>function</em> that maps input to output</td>
</tr>
<tr>
<td><strong>logic</strong></td>
<td>... <em>relation</em> between input and output</td>
</tr>
<tr>
<td><strong>object-oriented</strong></td>
<td>... <em>simulation</em> through a set of interacting objects</td>
</tr>
</tbody>
</table>
More declarative

Programming Paradigms

The principal programming paradigms

"More is not better (or worse) than less, just different."

QUERY