Example: Expression Hierarchy
• Define general concept Expression
• Implement two forms: Number, Sum
• Methods on implemented types of exprs evaluate, toString, draw, ...
• Ex:
  e = new Sum(new Number(23), new Number(2));
  print e.toString() + " = " + e.evaluate();

• Anticipate additions to library

abstract class Expr {
  public abstract String toString();
  public abstract int eval();
}

class Number extends Expr {
  private int n;
  public Number(int n) { this.n = n; }
  public String toString() { return "" + n; }
  public int eval() { return n; }
}

class Sum extends Expr {
  private Expr left, right;
  public Sum(...) { .... } // constructor
  public String toString() {
    return left.toString() + " + " + right.toString();
  }
  public int eval() { return left.eval() + right.eval(); }
}

Steve Jobs on Touring Xerox PARC
And they showed me really three things. But I was so blinded by the first one I didn’t even really see the other two. One of the things they showed me was object orienting programming—they showed me that but I didn’t even see that. The other one they showed me was a networked computer system...they had over a hundred Alto computers all networked using email etc., etc., I didn’t even see that. I was so blinded by the first thing they showed me which was the graphical user interface...within you know ten minutes it was obvious to me that all computers would work like this some day.
Dynabook

"A Personal Computer for Children of All Ages", Alan Kay, 1972

Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I

John McCarthy, 1960

A programming system called LISP (for LISt Processor) has been developed for the IBM 704 computer by the Artificial Intelligence group at M.I.T. ... In this article, we first describe a formalism for defining functions recursively.

Smalltalk in Unix Lab

• Directions in HW
  - makes big files, so please follow directions to work on scratch disk.
  - (You can always use this disk for other purposes as well.)

• To find Objects/Widgets/etc:
  - right-click on background
  - select "flaps..."
  - click on "Supplies", "Tools", "Widgets", ...

Example: Point Class

• Class definition written in tabular form

<table>
<thead>
<tr>
<th>class name</th>
<th>Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>super class</td>
<td>Object</td>
</tr>
<tr>
<td>class vars</td>
<td>pt</td>
</tr>
<tr>
<td>instance vars</td>
<td>x y</td>
</tr>
<tr>
<td>Class messages and methods</td>
<td>...names and code for methods...</td>
</tr>
<tr>
<td>Instance messages and methods</td>
<td>...names and code for methods...</td>
</tr>
</tbody>
</table>

Example: Point Class

Class definition written in tabular form

<table>
<thead>
<tr>
<th>class name</th>
<th>Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>super class</td>
<td>Object</td>
</tr>
<tr>
<td>class vars</td>
<td>pt</td>
</tr>
<tr>
<td>instance vars</td>
<td>x y</td>
</tr>
<tr>
<td>Class messages and methods</td>
<td>...names and code for methods...</td>
</tr>
<tr>
<td>Instance messages and methods</td>
<td>...names and code for methods...</td>
</tr>
</tbody>
</table>

Instance Messages and Methods

<table>
<thead>
<tr>
<th>Instance methods</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>moveDx: dx Dy: dy</td>
<td>pt.moveDx: 1 Dy: 1</td>
</tr>
<tr>
<td>x &lt;- dx + x</td>
<td></td>
</tr>
<tr>
<td>y &lt;- dy + y</td>
<td></td>
</tr>
</tbody>
</table>

In Java:

void moveDxDy(int dx, int dy) {
    x = x + dx;
    y = y + dy;
}

Instance Messages and Methods

<table>
<thead>
<tr>
<th>Instance methods</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>moveDx: dx Dy: dy</td>
<td>pt.moveDx: 1 Dy: 1</td>
</tr>
<tr>
<td>x &lt;- dx + x</td>
<td></td>
</tr>
<tr>
<td>y &lt;- dy + y</td>
<td></td>
</tr>
</tbody>
</table>

x: xcoord y: ycoord | pt: x:3 y:2
x <- xcoord |
y <- ycoord

void my(int xcoord, int ycoord) {
    x = xcoord;
    y = ycoord;
}
Instance Messages and Methods

Instance methods

moveDx: dx Dy: dy | |
  x <- dx + x
  y <- dy + y

x: xcoord y: ycoord | |
  x <- xcoord
  y <- ycoord

moveDx: Dy: | |
  x <- pt x + pt y
  (...draw point...)

Class Messages and Methods

Class methods

newX: xval Y: yval | |
  ^ self newX: xval y: yval

newOrigin | |
  ^ self new x: 0 y: 0

class Point {
  static Point newX(int xval, int yval) {
    Point temp = new Point();
    temp.xy(xval, yval);
    return temp;
  }
}

Examples

pt <- Point newX:3 Y:2

Class Meta Data

• Define colored points from points

Run-time Representation

• Three primary operations
  - object creation
  - method lookup
  - field lookup
**ColorPoint Methods**

**Instance Methods**

```plaintext
x: xcoord y: ycoord c:col

x <- xcoord
y <- ycoord
color <- c

color | | ^color
draw | | ...
```

**Class Methods**

```plaintext
newX: xv Y: yv C:cv

^self newX:xv y:yv c:cv

newOrigin ||
^self newX:0 Y:0 C:red
```

**Run-time Representation**

```
Point object
Point class
Template
Method dictionary

ColorPoint object
ColorPoint class
Template
Method dictionary
```

**Collection Hierarchy**

```
Collection
  isEmpty, size, includes: ...

Indexed
  at
  atPut

Updatable
  at
  atPut

Array
  replaceFrom:to:With:

SortedCollection
  sort

Dictionary
  contains:

Set
  add:
  remove:
```

**Ingrals Test for OO Languages**

- In an OO language, you should be able to:
  - Define a new kind of integer,
  - Put your new integers into a rectangle,
  - Ask the system to fill in the rectangle, and
  - Have it work.