Smalltalk

CSCI 334
Stephen Freund

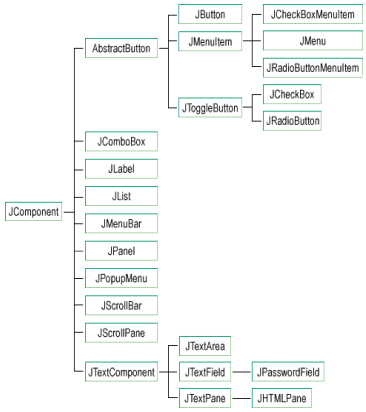
Example: Expression Hierarchy

- 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(...) { ... }
  public String toString() {
       return left.toString() + right.toString()
  }
  public int eval() { return left.eval() + right.eval(); }
}
```

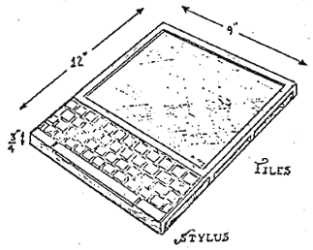


Steve Jobs on Touring Xerox PARC

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:
  - click on background
  - select "flaps..."
  - enable "Supplies", "Tools", "Widgets", ...

Example: Point Class

- Class definition written in tabular form

class name	Point
super class	Object
class vars	pi
instance vars	x y
class messages and methods	
(...names and code for methods...)	
instance messages and methods	
(...names and code for methods...)	

constructors

Instance Messages and Methods

Instance methods

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

Usage

```
pt moveDx: 1 Dy: 1
```

In Java:

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

```
pt.moveDxDy(1,1);
```

Instance Messages and Methods

Instance methods

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

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

Usage

```
pt moveDx: 1 Dy: 1

pt x:3 y:2
```

```
void xy(int xcoord,
        int ycoord) {
  x = xcoord;
  y = ycoord;
}
```

```
pt.xy(3,2);
```

Instance Messages and Methods

Instance methods

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

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

x | | ^x
y | | ^y
draw | |
    ⟨...draw point...⟩
```

Examples

```
pt moveDx: 1 Dy: 1

pt x:3 y:2

z <- pt x + pt y
```

Class Messages and Methods

Class methods

```
newX: xval Y: yval  | |
  ^ self new x: xval y: yval
```

Examples

```
p <- Point newX:3 Y:2
```

(new is method inherited from Object)

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

```
p = Point.newXY(3,2);
```

Class Messages and Methods

Class methods

```
newX: xval Y: yval  | |
  ^ self new x: xval y: yval

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

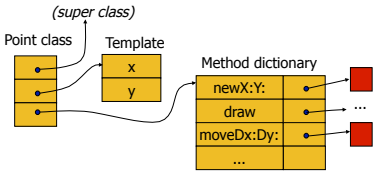
Examples

```
p <- Point newX:3 Y:2

p <- Point newOrigin
```

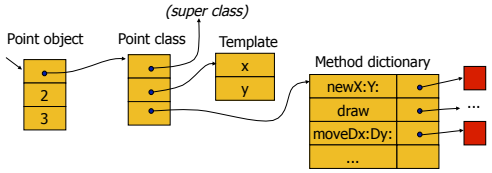
```
class Point {
  static Point newOrigin() {
    Point temp = new Point();
    temp.xy(0, 0);
    return temp;
  }
}
```

```
p = Point.newOrigin();
```

Class Meta Data



Run-time Representation



- Three primary operations
  - object creation
  - method lookup
  - field lookup

Inheritance

- Define colored points from points

class name	ColorPoint
super class	Point
class var	
instance var	color
class messages and methods	
newX:xv Y:yv C:cv	⟨ ... code ... ⟩
instance messages and methods	
color	| | ^color
draw	⟨ ... code ... ⟩

new instance variable

new method

override Point method

ColorPoint Methods

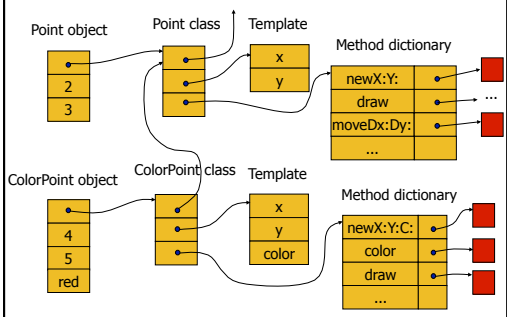
Instance Methods

```
x: xcoord y: ycoord c:col | |
    x <- xcoord
    y <- ycoord
    color <- c

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

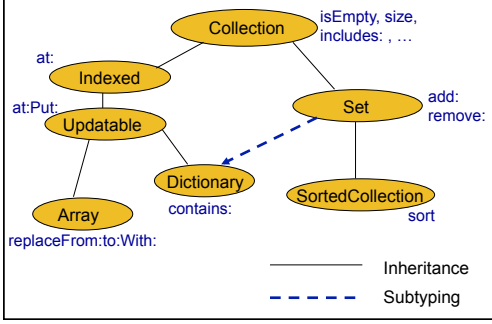
Class Methods

```
newX: xv Y: yv C:cv ||
   ^self new x:xv y:yv c:cv

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

Run-time Representation



Collection Hierarchy



Ingalls 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.