Example: Expression Hierarchy

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

- Anticipate additions to library

```java
interface ThermometerObserver {
    void changed(Thermometer t);
}

class Thermometer {
    protected int current; // mutable state
    protected Vector<ThermometerObserver> observers;
    void addObserver(ThermometerObserver observer) {
        observers.add(observer);
    }
    void setCurrent(int newValue) {
        current = newValue;
        for (int i = 0; i < observers.size(); i++) {
            observers.get(i).changed(this);
        }
    }
    int getCurrent() { return current; }
}

class DigitalThermometer implements ThermometerObserver {
    protected JTextField label;
    void changed(Thermometer t) {
        label.setText(t.getCurrent() + " °C");
    }
}
class MercuryThermometer implements ThermometerObserver {
    ... void changed(Thermometer t) {... }
}

Thermometer therm = new Thermometer();
ThermometerObserver t1 = new DigitalThermometer(...);
therm.addObserver(t1);
ThermometerObserver t2 = new MercuryThermometer(...);
therm.addObserver(t2);
```

Dahl and Nygaard
Why Objects For Simulation?

interface Event {
    void process();
}

while (!queue.isEmpty()) {
    Event e = queue.dequeue();
    e.process(); // may add new events to queue...
}

Event ButtonPushed...
DoorsOpened...
PersonEntered

Simula Point Class

class Point(x,y):
    real x,y;
    begin
        boolean procedure equals(q): ref(Point) q;
        if q /= none then
            equals := abs(x - q.x) + abs(y - q.y) < 0.001;
        else error();
        real procedure distance(q): ref(Point) q;
        if q == none then
            error();
        else
            distance := sqrt((x-q.x)**2 + (y-q.y)**2);
        end *** Point ***
    p1 := new Point(1.0, 2.5);
    p2 := new Point(2.0, 3.5);
    if p1.distance(p2) < 2 then
        c := new Circle(...)

Representation of Objects

Object is represented by activation record with access link to find global variables according to static scoping

Simula Line Class

class Line(a,b,c):
    real a,b,c;
    begin
        boolean procedure parallelto(l): ref(Line) l;
        if l /= none then  parallelto := ...
        ref(Point) procedure meets(l): ref(Line) l;
        if l /= none and ~parallelto(l) then ...
        end;
        real d;
        d := sqrt(a**2 + b**2)/2;
        if d = 0.0 then error else 
        begin
            d := 1/d;
            a := a*d; b := b*d; c := c*d;
        end;
        end *** Line ***

Initialization: "normalize" a,b,c

Call to p1.distance(p2)

Local variables
line determined by ax+by+c=0
Derived Classes in Simula

class A(str); begin
  procedure moo ... end;
A class B(x); begin
  procedure moo ... end;
A class C(y,z); begin end;
B class D(m,n); begin
  procedure moo ... end;

Object Creation

class A(str); begin
  procedure moo ... end;
A class B(x); begin
  procedure moo ... end;
A class C(y,z); begin end;
B class D(m,n); begin
  procedure moo ... end;

Object Creation

Object Creation

Subtyping

The type of an object is its class
Subclasses are subtypes:

Subtyping and Inheritance
Subtyping

class Point(x,y): begin proc draw() ... Point class ColorPoint(c): begin proc draw()... ref(Point) a :- new Point(...) ref(ColorPoint) b :- new ColorPoint(...) a := b // legal: ColorPt is subclass of Pt ... b := a // also legal, but run-time test. b.setColor(red);

Java Array Covariant Subtyping Rule

class Point { ... } class ColorPoint extends Point { ... }
Point pts[] = new Point [100];
pts[0] = new Point (10,10); pts[1] = new Point (20,20);

Java Array Covariant Subtyping Rule

class Point { ... } class ColorPoint extends Point { ... }
ColorPoint cpts = new ColorPoint[100]; Point pts[] = cpts;
pts[0] = new Point (10,10); pts[1] = new Point (20,20);
... cpts[0].setColor(RED);

Why Did They Add It?

static void arrayCopy(Point src[], Point dst[]) {
  for (int i = 0; i < src.length; i++)
    dst[i] = src[i];
}

Point p[];
Point q[];
arrayCopy(p,q);
String s[];
String t[];
arrayCopy(s,t);

General arrayCopy Operation

static void arrayCopy(Object src[], Object dst[]) {
  for (int i = 0; i < src.length; i++)
    dst[i] = src[i];
}

Point p[];
Point q[];
arrayCopy(p,q);
String s[];
String t[];
arrayCopy(s,t);

From Bill Joy (Sun Cofounder)

Date: Fri, 09 Oct 1998 09:41:05 -0600
From: bill joy
Subject: ...[discussion about java genericity]

actually, java array covariance was done for less noble reasons ... it made some generic "bcopy" (memory copy) and like operations much easier to write...
I proposed to take this out in 95, but it was too late (...). i think it is unfortunate that it wasn’t taken out...
it would have made adding genericity later much cleaner, and [array covariance] doesn’t pay for its complexity today.