Using Abstractions to Make Concepts Concrete

Kim Bruce
Williams College
on leave at UC - Santa Cruz

My mother,
Jane Bruce

My wife, Fatma Kassamali

Why am I here?

Work with lots of smart people!

Williams CS Colleagues

Tom Murtagh Andrea Danyluk

Liberal Arts Computer Science Consortium


Arcadia High School
Pomona College
My Teachers
MIT
UC Santa Cruz
University of Wisconsin

My students

And of course, NSF!

Abstraction

- Computer Scientists build abstractions to
  - Extract common features
  - Hide irrelevant details / control complexity
  - Protect integrity
- Abstraction used to present simple consistent model

Teachers & Abstraction

- As teachers, we do the same thing
  - Abstract away from the complexities of material
  - Present simple consistent model
  - Slowly add complexity
  - Filling in exceptional cases later

A Few Examples

- Classes and Objects
- Semantics of Assignments & Parameter Passing
- Recursion

Classes & Objects

Classes & Objects

- Need simple, but compelling examples
  - Not bank accounts
  - Not strings
- Should have both state and behavior
- The more concrete the better

Possibilities

- Microworlds -- e.g., Karel J. Robot, Alice, etc.
- Environments that support exploration and visualization -- e.g., BlueJ
- Graphic objects using simplifying libraries:
  - Objectdraw, Java PowerTools, acm Java library, Breezy Swing

Objects-Early Tools - Friday @ 4 p.m.
Java Task Force Status report - today @ 10:30 a.m.

Objectdraw Library

- Predefined graphics classes
- WindowController
  - extension of JApplet with drawing canvas in center
  - pre-wired as mouse listener for canvas
- Support for concurrency w/o exceptions

Support for First 3 Weeks

- Graphics are concrete objects
  - Appear when created
  - Redrawn automatically when changed
- Event-driven programming gives fine control over when commands executed.
- Makes programs reactive.

Run examples here

Examples

- Advantages:
  - Use of constructors and methods
  - Instant and visible feedback
  - Methods short and clear
  - Interesting programs with no loops

Assignment Semantics

Assignment Semantics

- What is meaning of simple assignment

  `id = exp;`

- In Pascal, C, or C++: Copy Semantics
  - Evaluate `exp` to obtain value `v`
  - Store copy of `v` into location corresponding to `id`
- Important to know what is pointer!

Java Assignment Semantics

- Similar for Java primitives.
- Java objects. Can give similar definition:
  - Evaluate `exp` to obtain reference to object `obj`.
  - Copy reference to `obj` into location corresponding to `id`.
- Assignment as sharing, not copying.

New Assignment Semantics

- What is meaning of

```
id = exp;
```

- For all types of values
  - Evaluate `exp` to obtain associated object `obj`
  - Associate `id` with object `obj`

New Assignment Semantics

- Subtle difference
  - Copy reference to `obj` into location corresponding to `id`

versus

  - Associate `id` with object `obj`
- More than one identifier can be associated with the same object (sharing!)

New Assignment Semantics

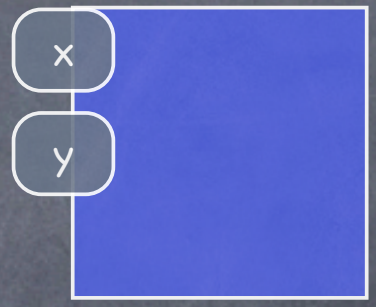
```
FramedRect x,y;
     new FramedRect(...);
```

New Assignment Semantics


```
FramedRect x,y;
x = new FramedRect(...);
```

New Assignment Semantics

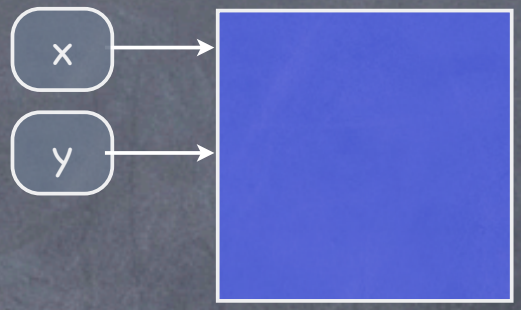
```
FramedRect x,y;
x = new FramedRect(...);
y = x;
```



Identifiers as labels

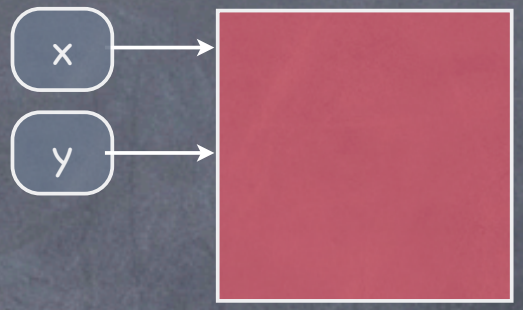
New Assignment Semantics

```
FramedRect x,y;
x = new FramedRect(...);
y = x;
```



Identifiers as tags

New Assignment Semantics




```
FramedRect x,y;
x = new FramedRect(...);
y = x;
y.setColor(Color.RED);
... x.getColor() ...;
```

Same explanation for primitives

No need to refer to memory

Recursion

Structural Recursion Before Arrays

Friday @ 10:55 a.m.

Recursion

- Recursive functions:

```
int fact(int n) {
    if (n <= 1) return 1;
    else return n*fact(n-1);
}
```

- Best understood via mathematical induction
- Successive recursive calls represented by activation records holding parameters and local variables, but not instance variables.

In O-O world:

- Recursive structures easier:

- Recursive substructures visible.
- Best understood via mathematical induction, but now visibly trace calls

Interface + 2 classes

Interface

```
public interface NestedRectsInterface {
  void moveTo(double x, double y);  }
```

Base class

```
public class EmptyRects implements
                          NestedRectsInterface {
  public EmptyRects() { }
  public void moveTo(double x, double y) { }
}
```

```
public class NestedRects implements NestedRectsInterface {
  private FramedRect outerRect;       // outer rect
  private NestedRectsInterface rest;  // inner nested rects

  public NestedRects(..., double width, double ht, ...) {
    outerRect = new FramedRect(x, y, width, ht, ...);
    if (width >= 8 && height >= 8) {
      rest = new NestedRects( x+4, y+4, width-8, ht-8, ...);
    } else {
      rest = new EmptyRects(); }
  }

  public void moveTo(double x, double y) {
    outerRect.moveTo(x, y);
    rest.moveTo(x + 4, y + 4);
  }
```

Structural Recursion

- More concrete than activation records.
- Dynamic method invocation via interfaces.
- Example really recursive linked lists!

Abstraction & CS

- In some ways, CS asks most of novices.
- In math students select and follow algorithms to solve problems.
- In CS students develop algorithms to solve problems.
- Requires higher level of cognitive processing.
- Help make things more concrete.

Making Concepts Concrete

- Creating new abstraction can make concepts more concrete.
  - Classes & Object examples w/ microworlds, visualization tools, or graphics
  - Reinterpret semantics of assignment - abstract from memory
  - Structural recursion vs functional recursion
- Use languages, software, and tools that represent & support simple models

Thank You!

http://eventfuljava.cs.williams.edu/