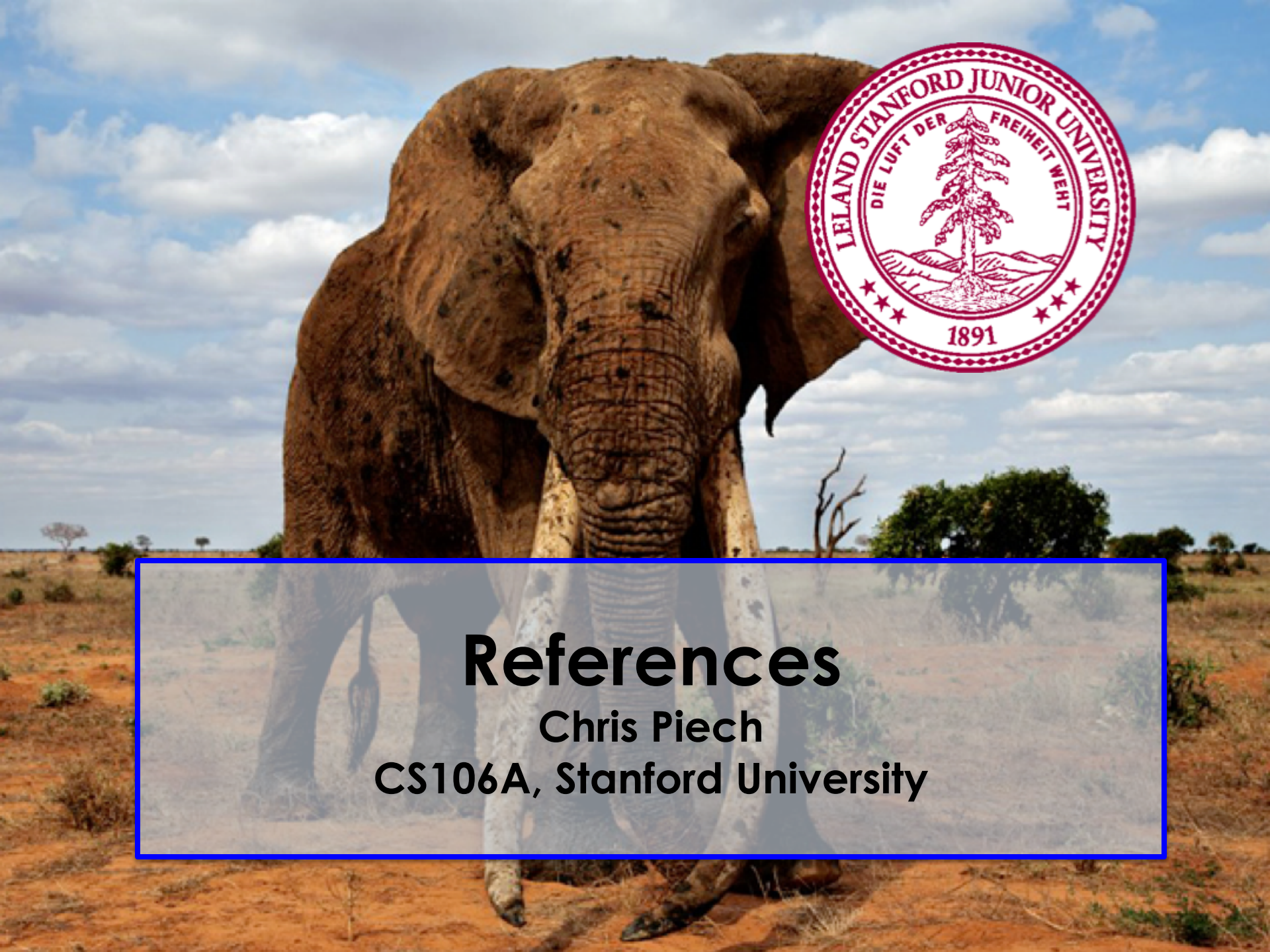


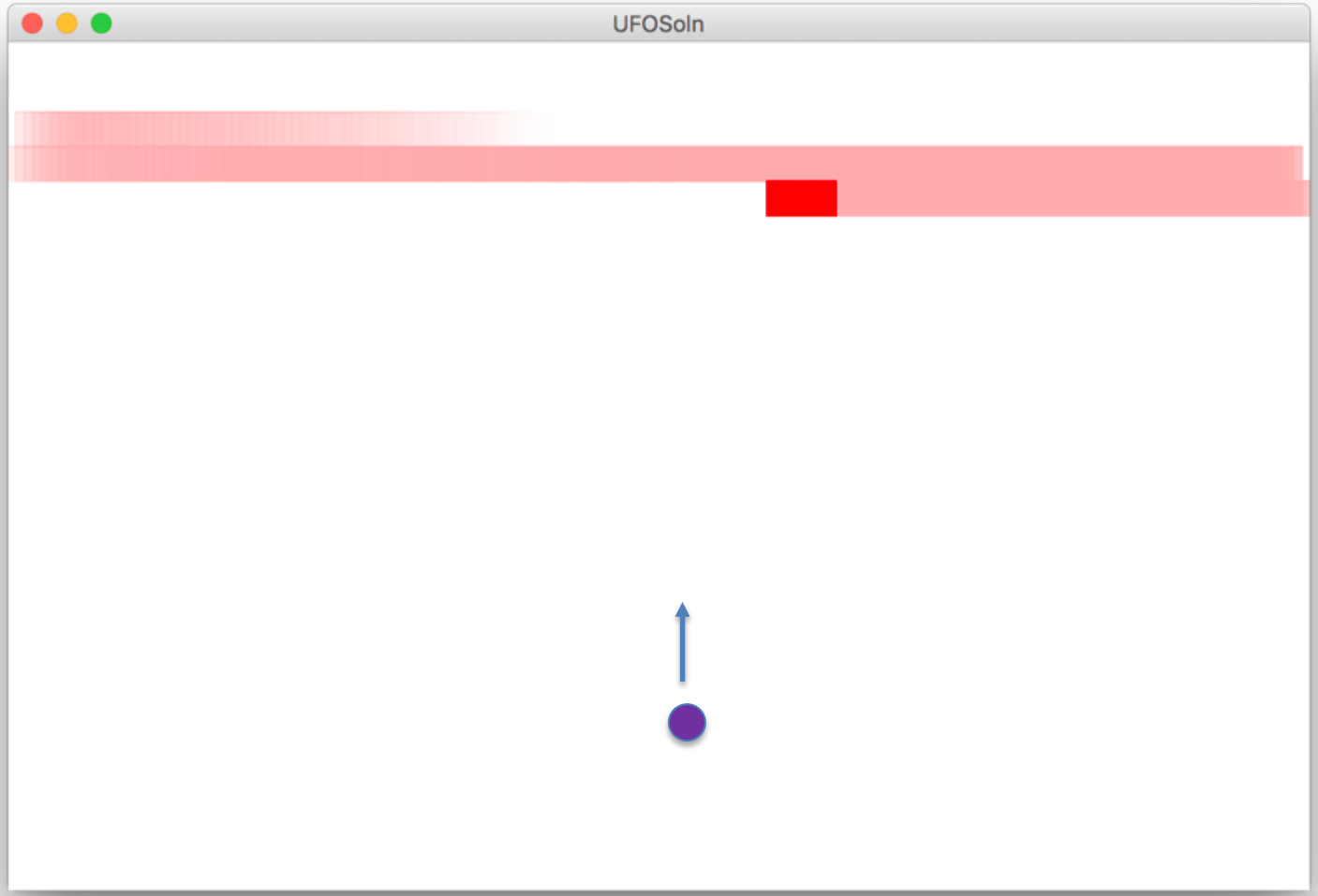
References

Chris Piech
CS106A, Stanford University

Learning Goals

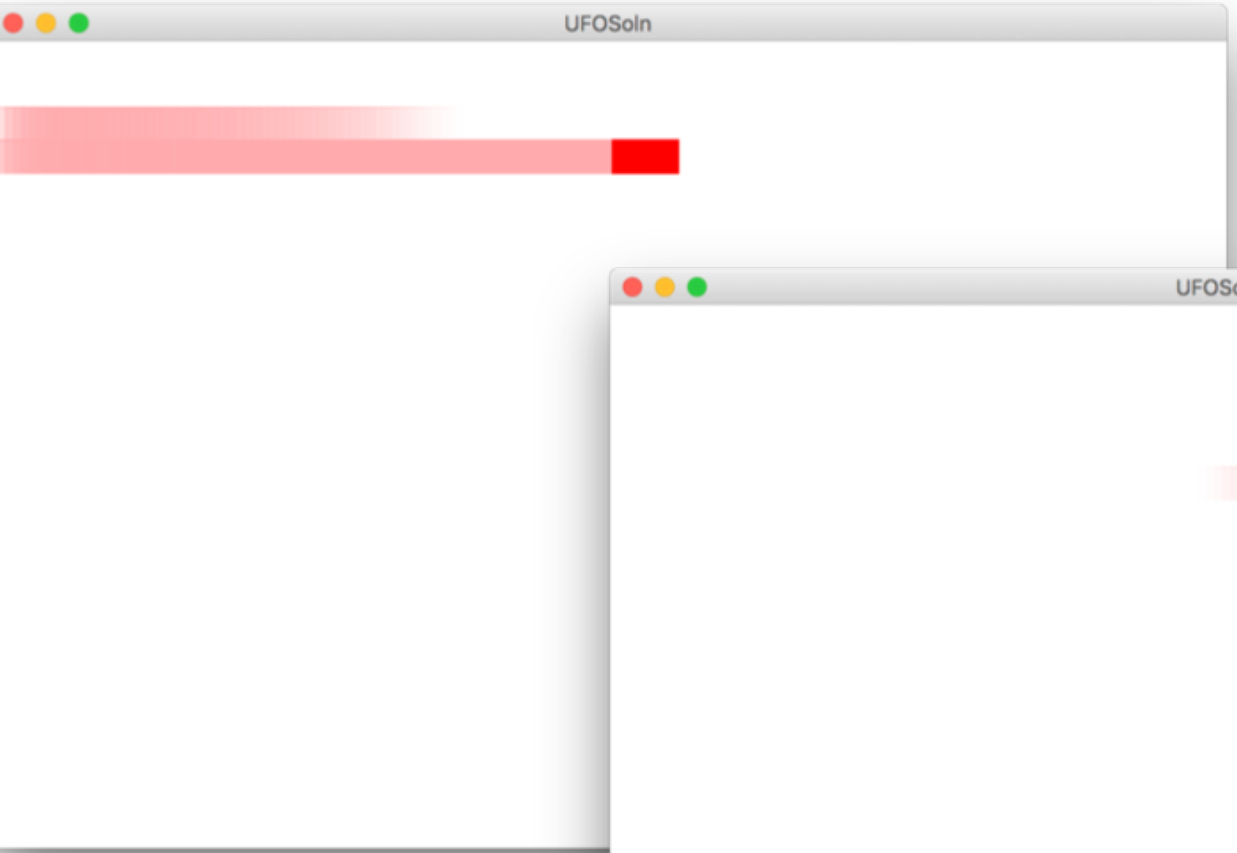
1. Be able to write a large program
2. Be able to trace memory with references

Today, we build!

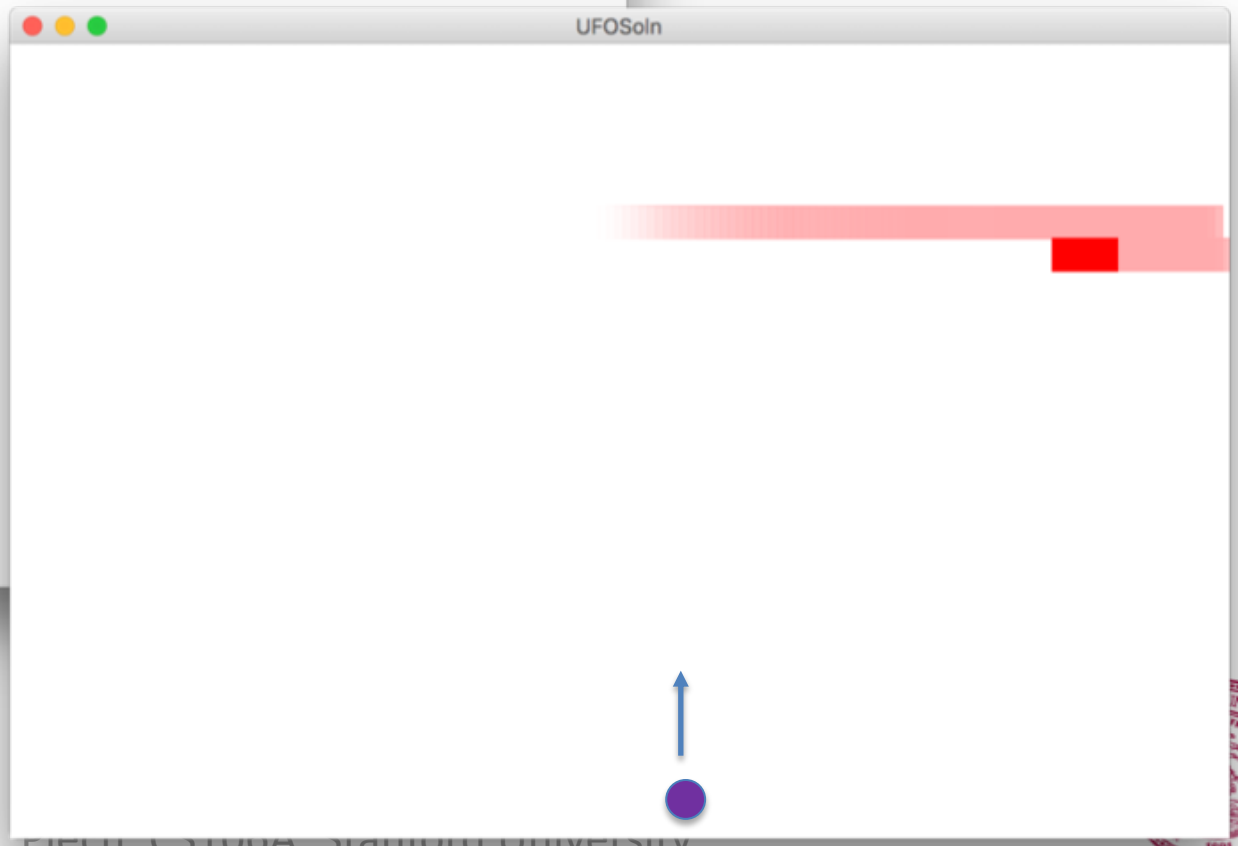


Milestones

Milestone 1



Milestone 2

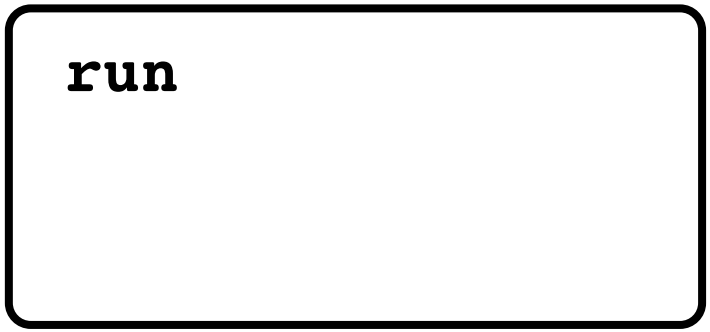


Advanced memory model

Core memory model

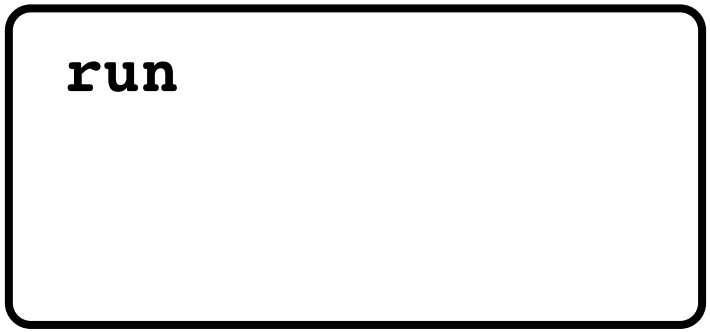
Stack Diagrams

```
public void run() {
    println(toInches(5));
}

private int toInches(int feet){
    int result = feet * 12;
    return result;
}
```



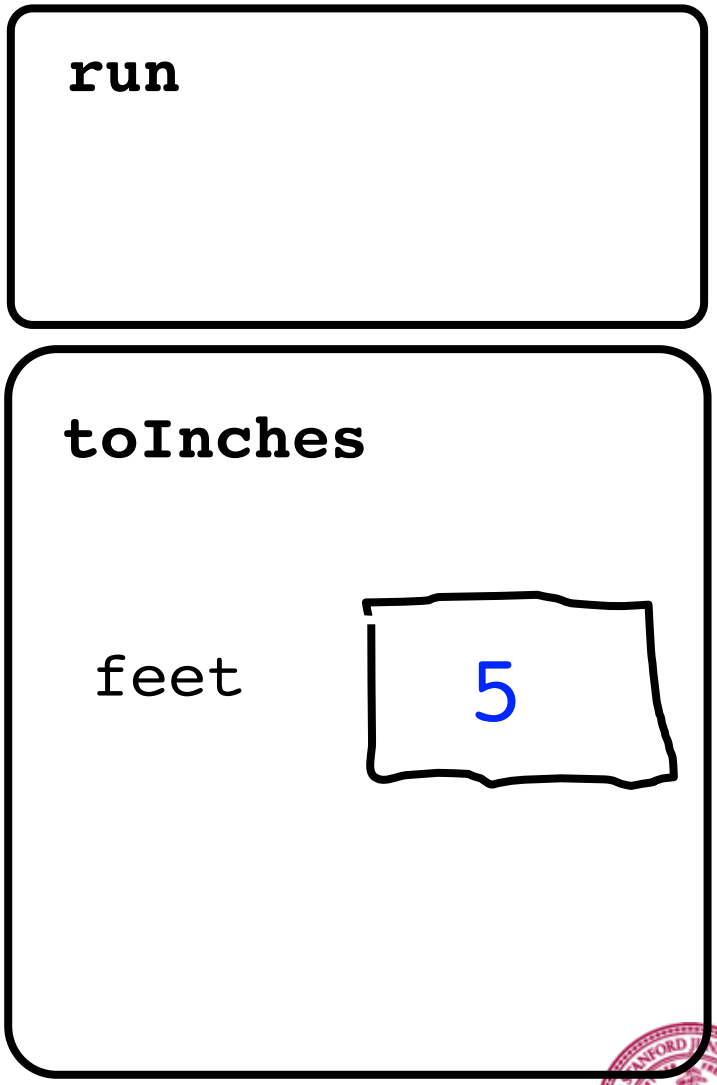
Stack Diagrams

```
public void run() {
    println(toInches(5));
}

private int toInches(int feet){
    int result = feet * 12;
    return result;
}
```



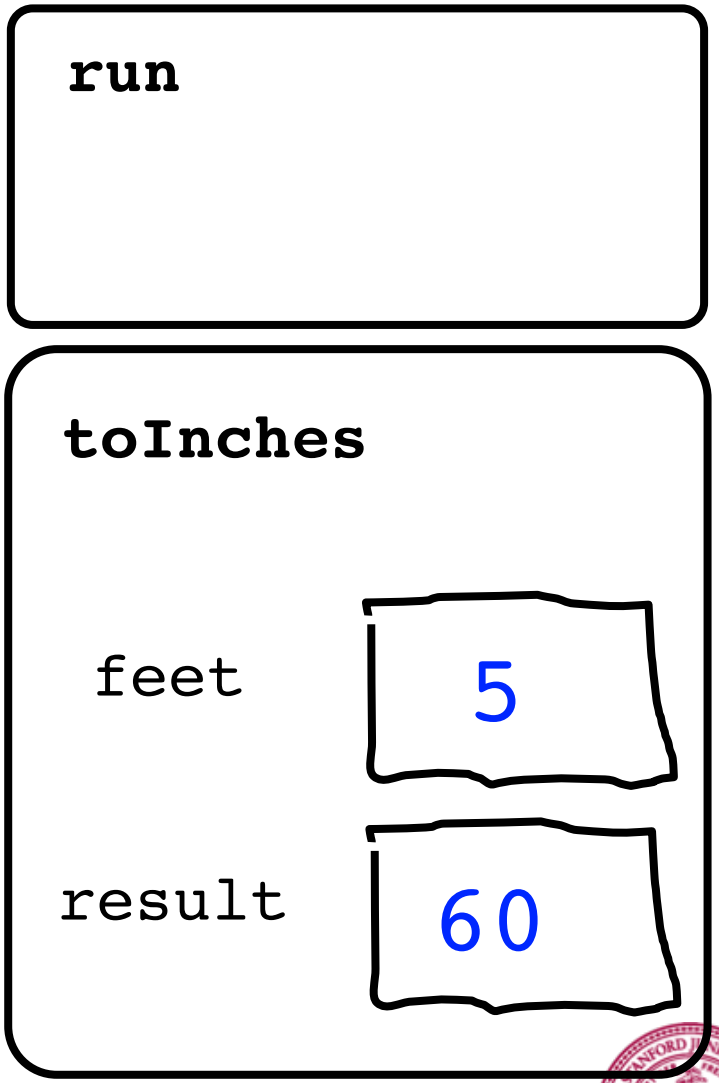
Stack Diagrams

```
public void run() {
    println(toInches(5));
}

private int toInches(int feet){
    int result = feet * 12;
    return result;
}
f
```



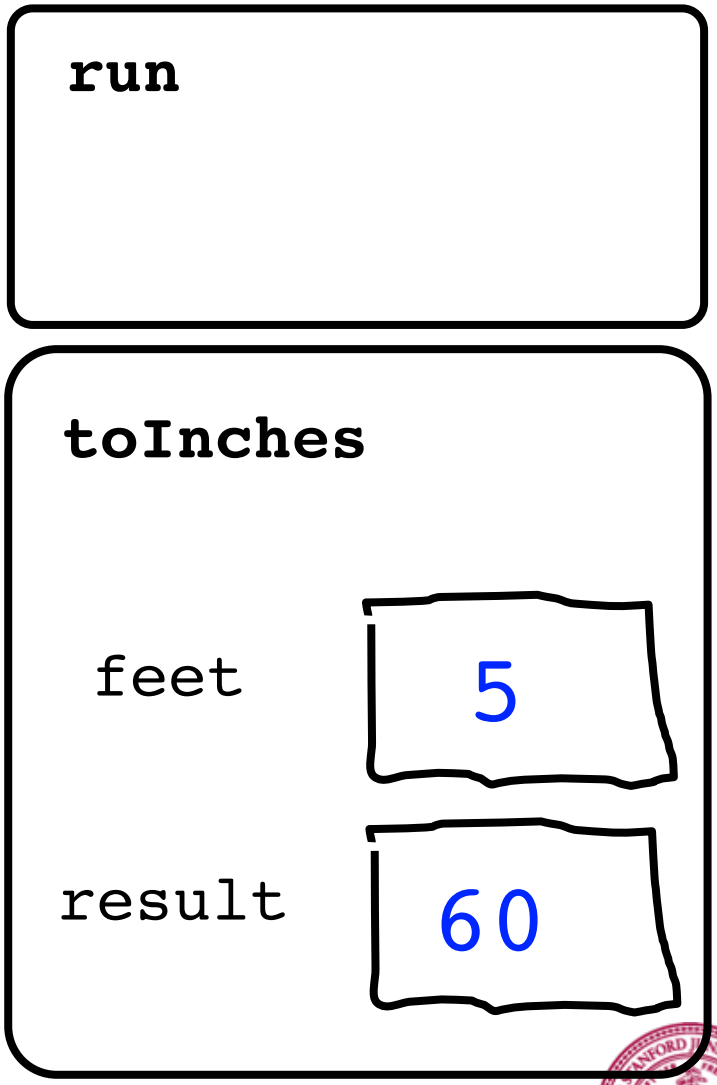
Stack Diagrams

```
public void run() {
    println(toInches(5));
}

private int toInches(int feet){
    int result = feet * 12;
    return result;
}
f
```



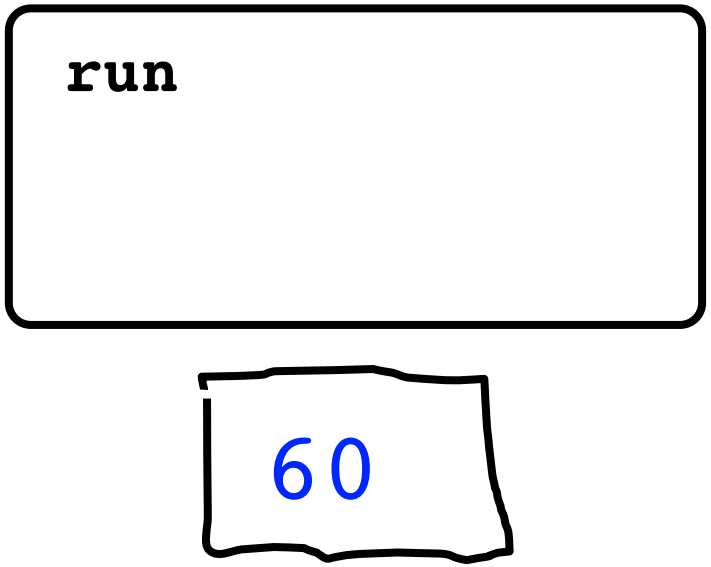
Stack Diagrams

```
public void run() {
    println(toInches(5));
}

private int toInches(int feet){
    int result = feet * 12;
    return result;
}
f
```



Stack Diagrams

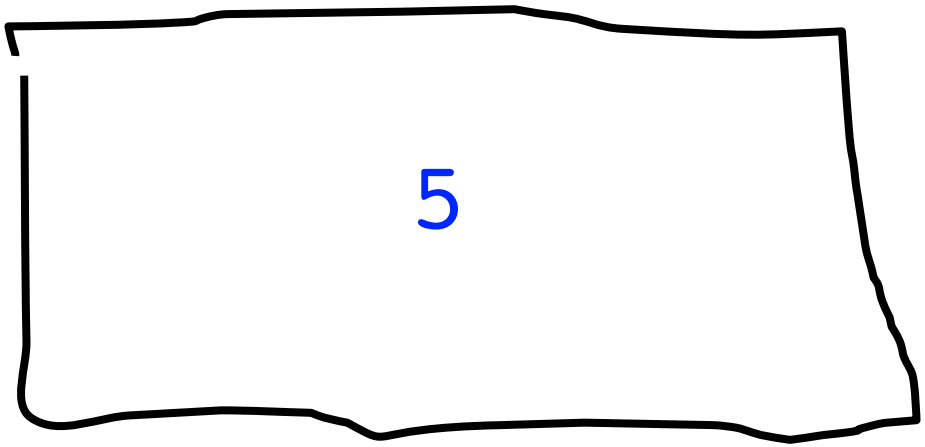
```
public void run() {
    println(toInches(5));
}

private int toInches(int feet){
    int result = feet * 12;
    return result;
}
f
```



Aside: Actual Memory

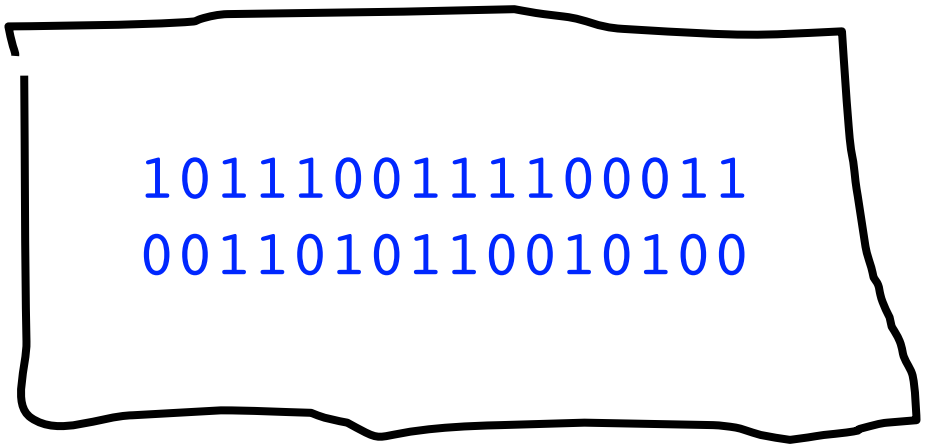
What is a bucket

feet



What is a bucket

feet



* Each bucket or “word” holds 64 bits

#0: don’t think on the binary level (yet)

End aside

Primitives vs Classes

Primitive Variable Types

```
int
double
char
boolean
```

Class Variable Types

```
GRect
GOval
Gline
Color
```

Class variables (aka objects)

1. Have upper camel case types
2. You can call methods on them
3. Are constructed using `new`
4. Are stored in a special way

How do you share wikipedia articles?

Antelope Canyon Article

Antelope Canyon is a slot canyon in the American Southwest. It is located on Navajo land east of Page, Arizona. Antelope Canyon includes two separate, photogenic slot canyon sections, referred to individually as *Upper Antelope Canyon* or *The Crack*; and *Antelope Canyon* or *The Corkscrew*.[2]

The Navajo name for Upper Antelope Canyon is Tsé bighánílíní, which means "the place where water runs through rocks." Lower Antelope Canyon is Hazdistazí (advertised as *"Hasdestwazi"* by the Navajo Parks and Recreation Department), or "spiral rock arches." Both are located within the LeChee Chapter of the Navajo Nation.[4]

Contents [hide]

1 Geology
2 Tourism and photography
2.1 Upper Antelope Canyon

Antelope Canyon

Tsé bighánílíní dóó Hazdistazí (Navajo)

A beam of light in Upper Antelope Canyon

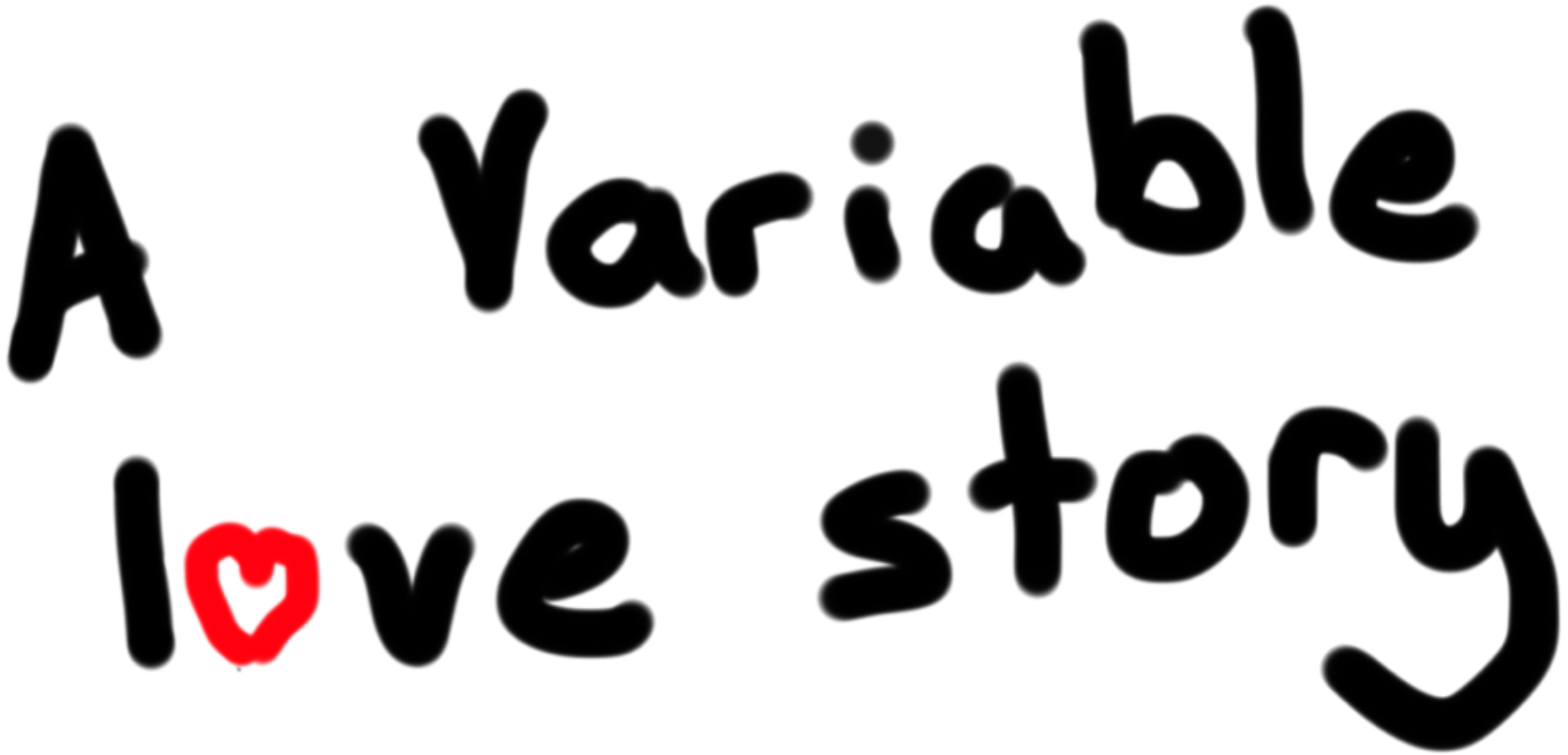
https://en.wikipedia.org/wiki/Antelope_Canyon

Objects store addresses
(which are like URLs)

What does an object store?

Objects store addresses
(which are like URLs)

A Variable love story



By Chris Piech

A Variable origin ~~love~~ story

Nick Troccoli

By ~~Chris Piech~~

Once upon a time…

...a variable x was born!

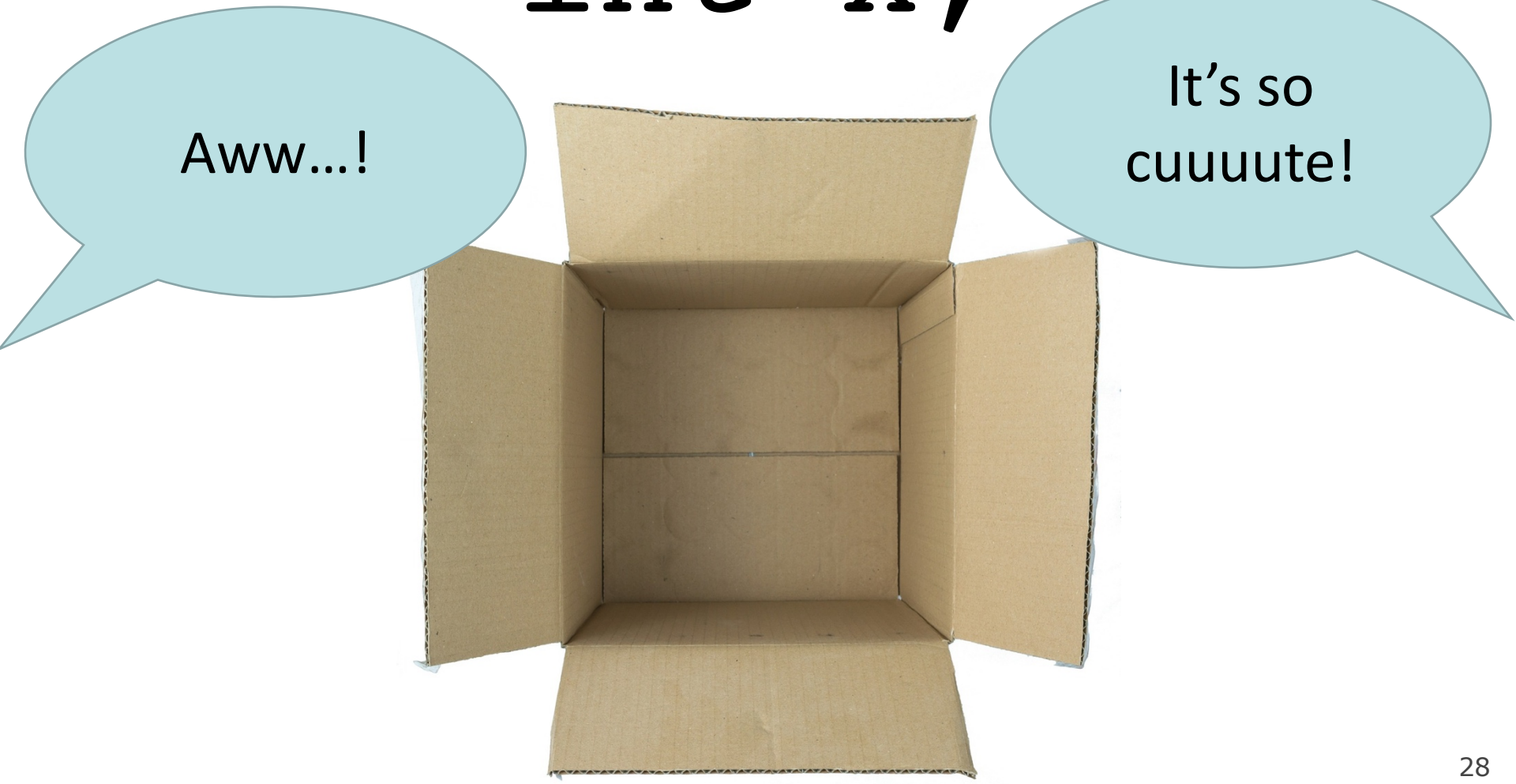
```
int x;
```

...a variable x was born!

```
int x;
```

x was a primitive variable…

```
int x;
```



...and its parents loved it very much.

```
int x;
```

We should give it…. value 27!

...and its parents loved it very much.

```
x = 27;
```

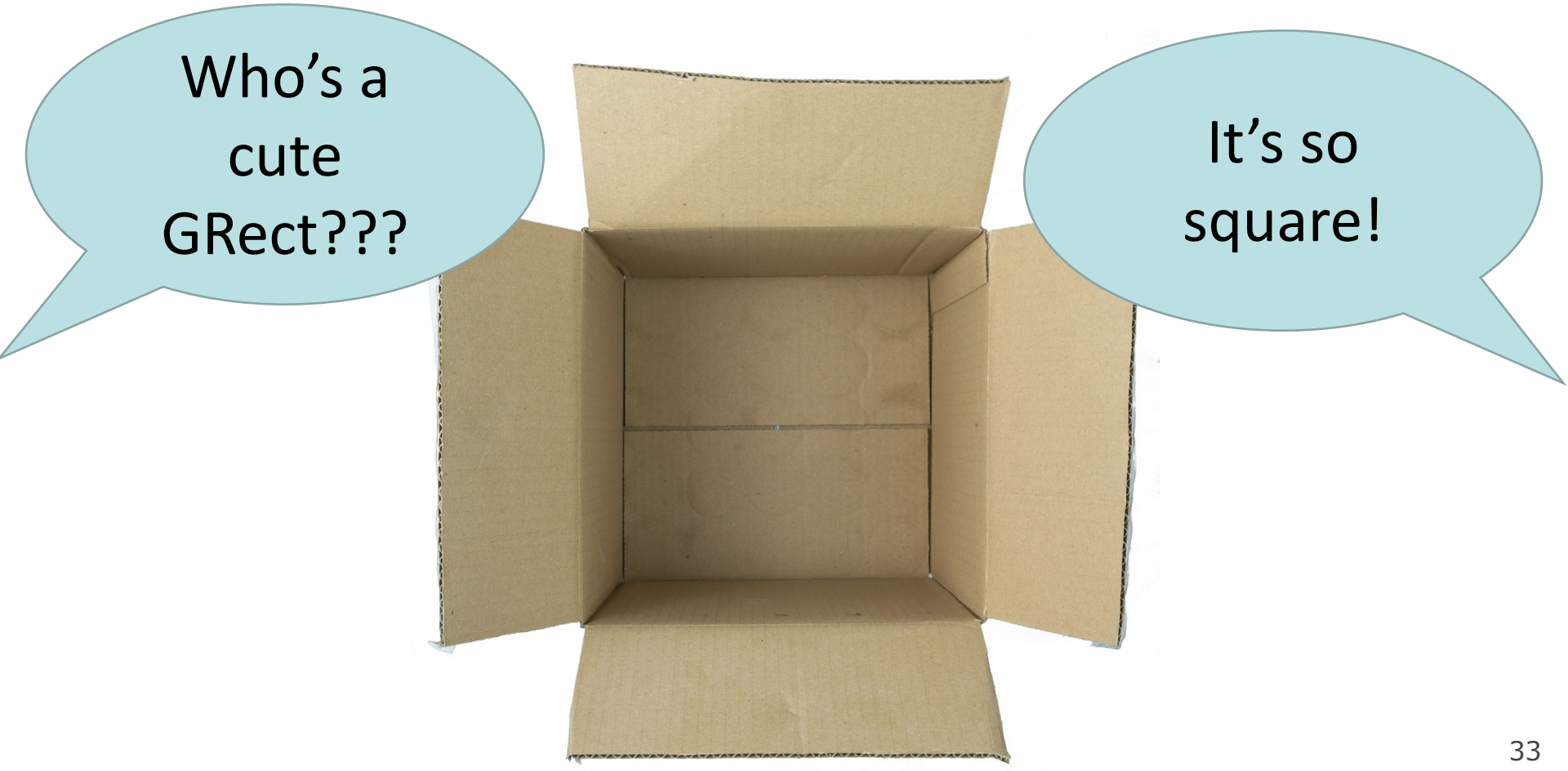
A few years later, the parents decided to have another variable.

...and a variable rect was born!

```
GRect rect;
```

rect was an object variable…

```
GRect rect;
```



...and its parents loved it very much.

```
GRect rect;
```

...and its parents loved it very much.

```
GRect rect = new Grect(0, 0, 50, 50);
```

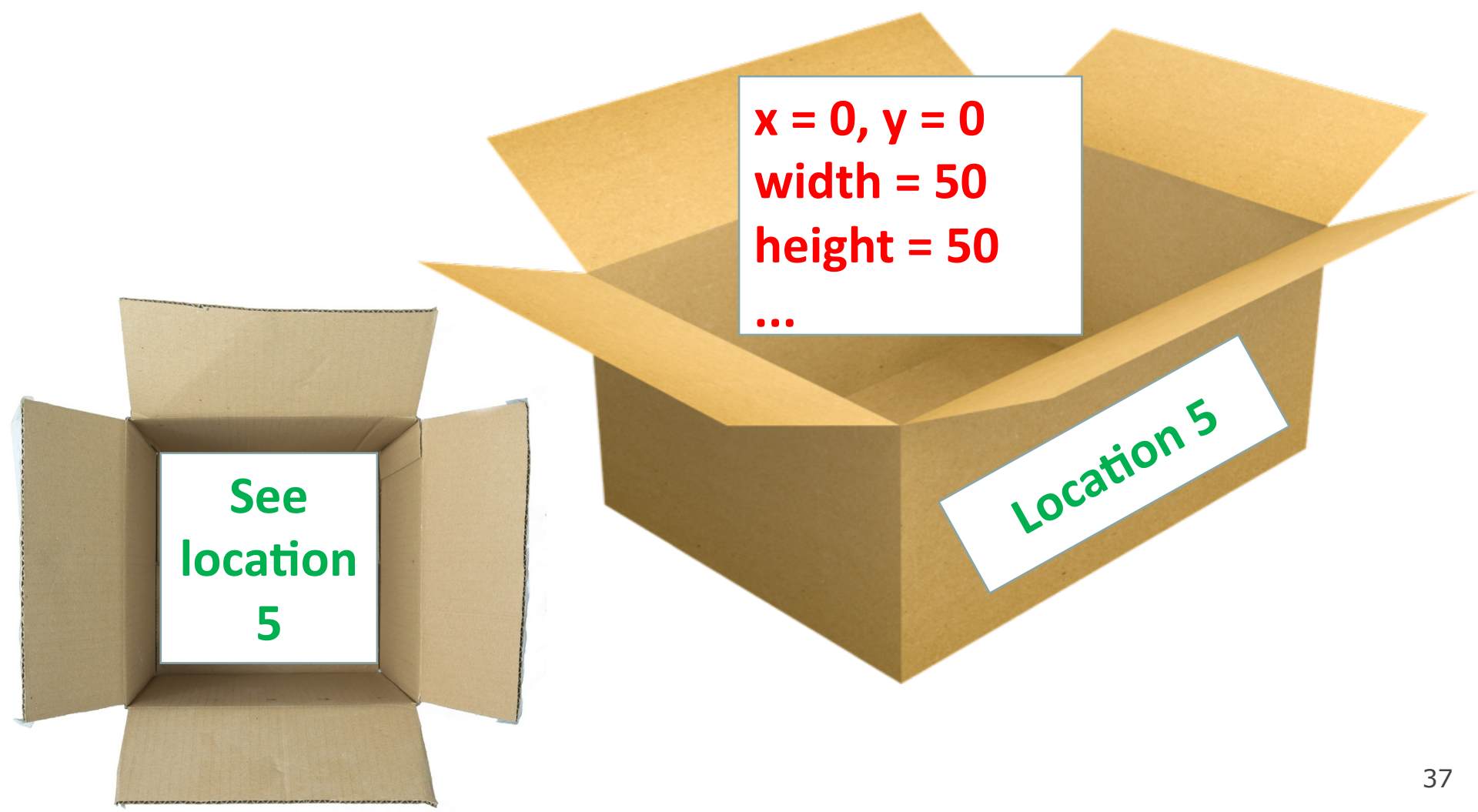
…but rect's box was not big enough for an object!

```
GRect rect = new Grect(0, 0, 50, 50);
```

That box isn't big enough to store everything about a GRect!

...so they stored the information in a bigger box somewhere else.

```
GRect rect = new Grect(0, 0, 50, 50);
```



Chapter 2: Coming soon

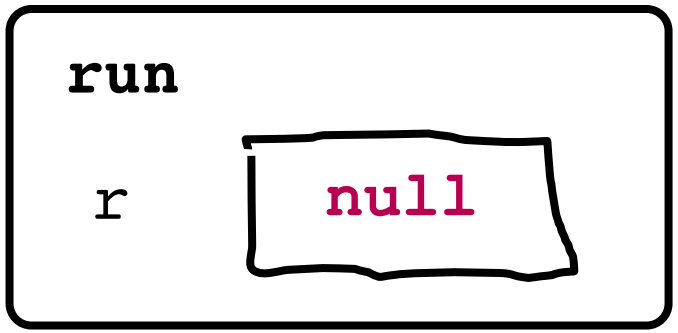
```
public void run() {
    GRect r = null;
}
```

Method memory

Object memory

```
public void run() {
   GRect r = null;
}
```

Method memory

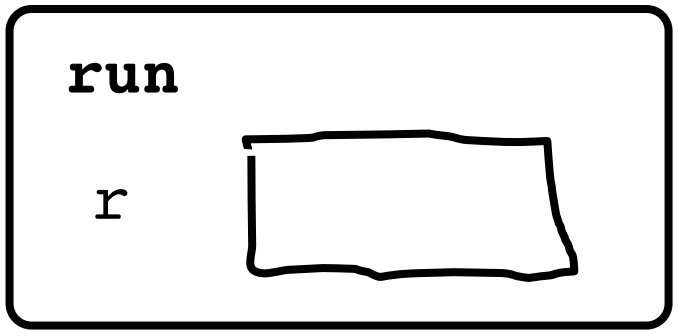


Object memory

Wahoo!

```
public void run() {
    GRect r = new GRect(50, 50);
}
```

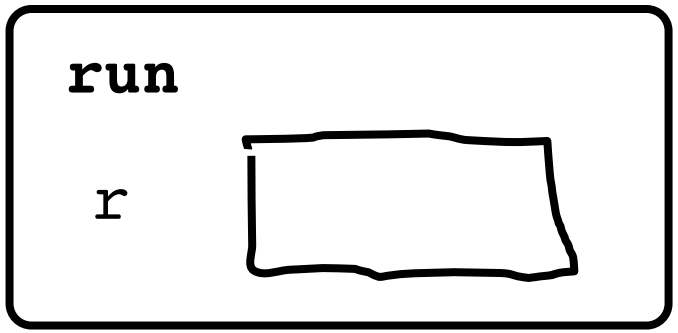
Method memory



Object memory

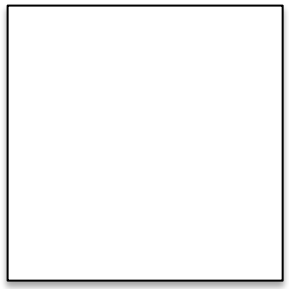
```
public void run() {
    GRect r = new GRect(50, 50);
}
```

Method memory



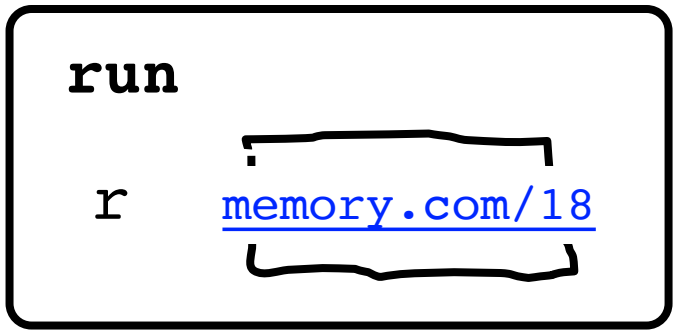
Object memory

memory.com/18

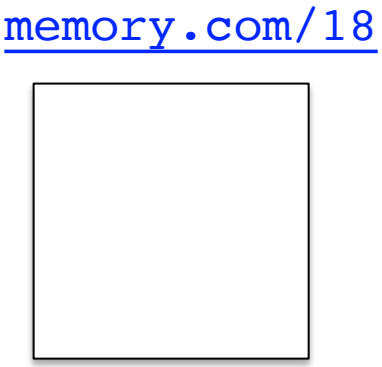


```
public void run() {
   GRect r = new GRect(50, 50);
}
```

Method memory

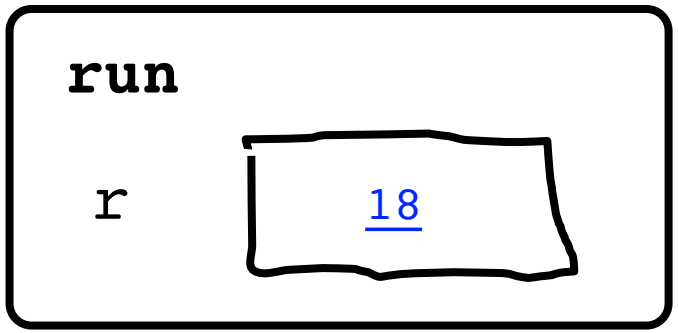


Object memory

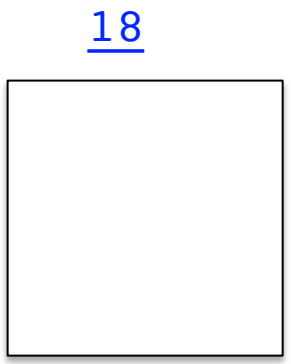


```
public void run() {
   GRect r = new GRect(50, 50);
}
```

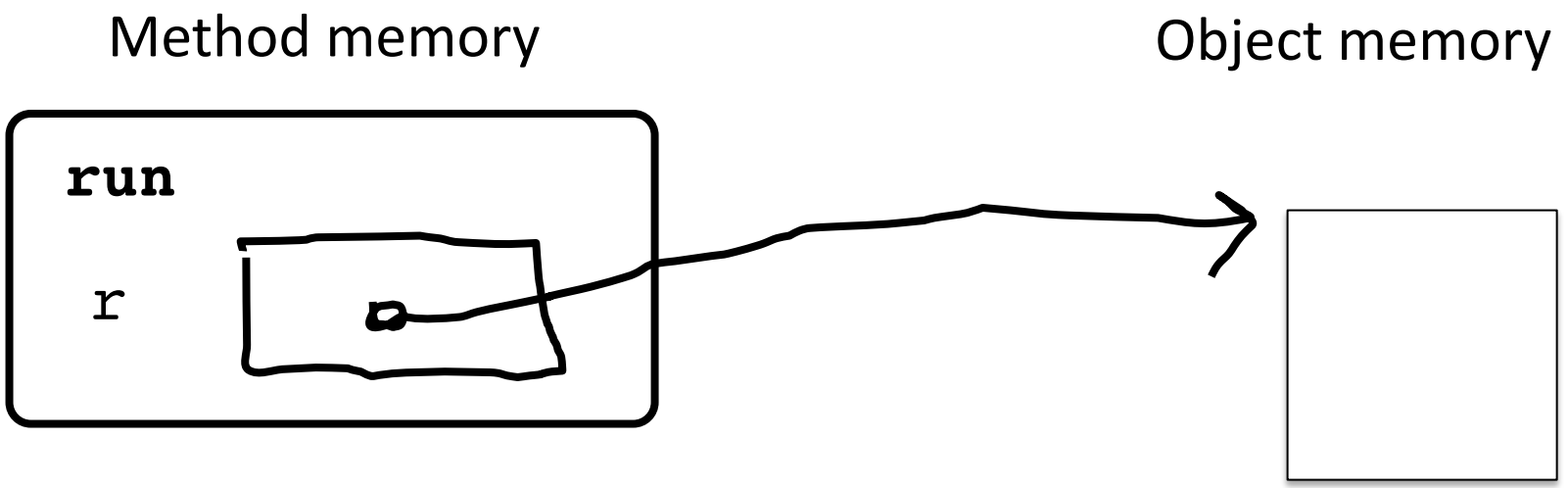
Method memory



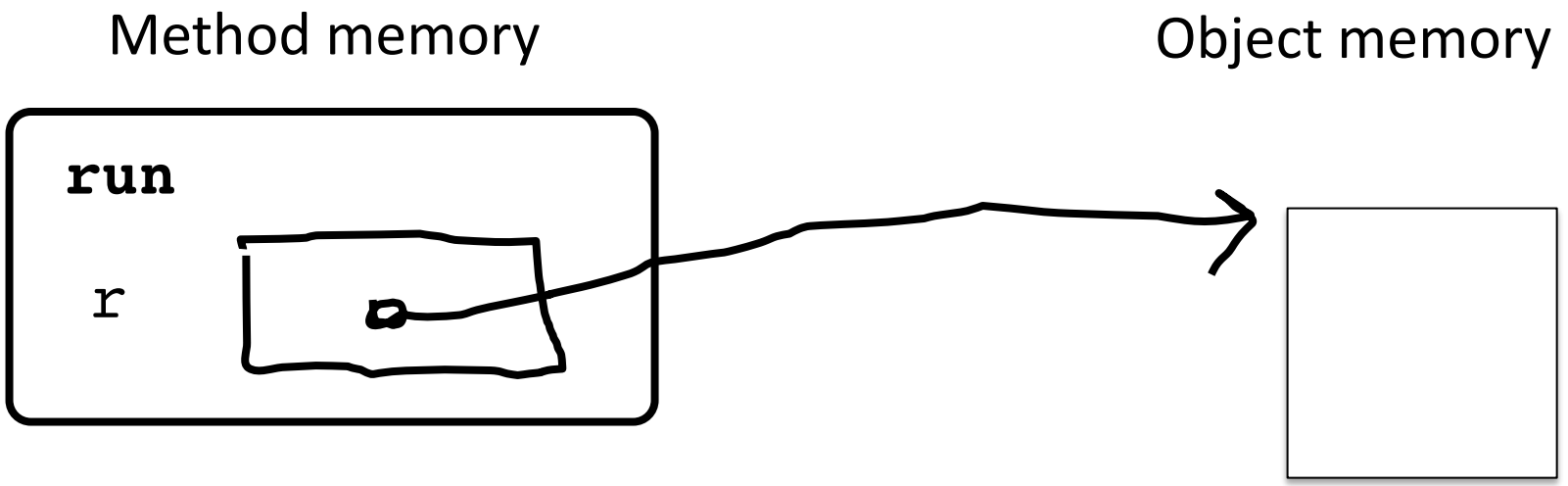
Object memory



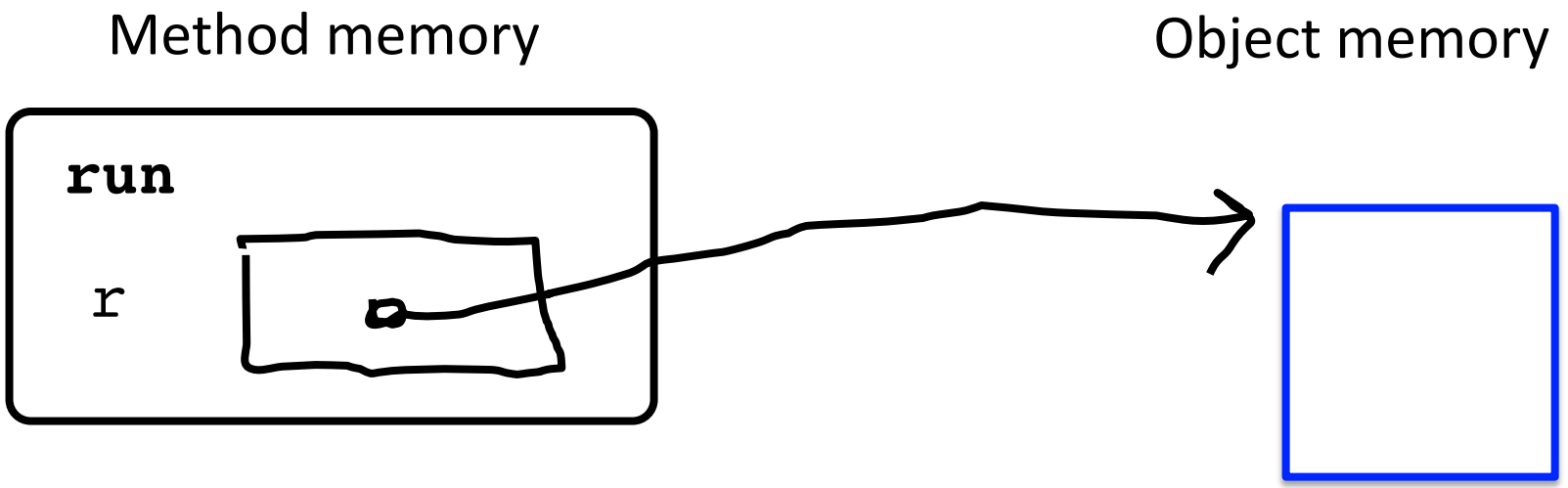
```
public void run() {
   GRect r = new GRect(50, 50);
}
```



```
public void run() {
   GRect r = new GRect(50, 50);
   r.setColor(Color.BLUE);
   r.setFilled(true);
}
```



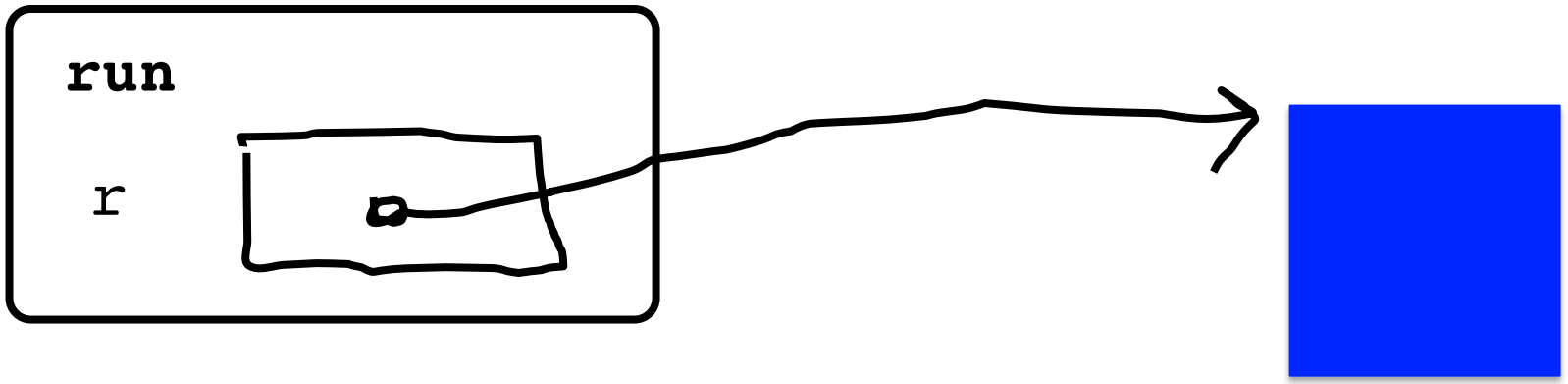
```
public void run() {
    GRect r = new GRect(50, 50);
    r.setColor(Color.BLUE);
    r.setFilled(true);
}
```



```
public void run() {
    GRect r = new GRect(50, 50);
    r.setColor(Color.BLUE);
    r.setFilled(true);
}
```

Method memory

Object memory



#1: `new` allocates memory for objects

* The data for an object can't always fit inside a fixed size bucket

#2: object variables store addresses

#ultimatekey

```
public class SimpleRect extends GraphicsProgram {

    public void run() {
        GRect r = null;
        r = new GRect(300, 300);
        r.setColor(Color.MAGENTA);
        add(r, 0, 0);
        addMouseListeners();
    }

    public void mousePressed(MouseEvent e) {
        GObject obj = getElementAt(1, 1);
        remove(obj);
    }

}
```


What does an object store?

Objects store addresses
(which are like URLs)

```
public class SimpleRect extends GraphicsProgram {

    public void run() {
        GRect r = null;
        r = new GRect(300, 300);
        r.setColor(Color.MAGENTA);
        add(r, 0, 0);
        addMouseListeners();
    }

    public void mousePressed(MouseEvent e) {
        GObject obj = getElementAt(1, 1);
        remove(obj);
    }

}
```


```
public class SimpleRect extends GraphicsProgram {

    public void run() {
        GRect r = null;
        r = new GRect(300, 300);
        r.setColor(Color.MAGENTA);
        add(r, 0, 0);
        addMouseListeners();
    }

    public void mousePressed(MouseEvent e) {
        GObject obj = getElementAt(1, 1);
        remove(obj);
    }

}
```


```
public class SimpleRect extends GraphicsProgram {

    public void run() {
        GRect r = null;
        r = new GRect(300, 300);
        r.setColor(Color.MAGENTA);
        add(r, 0, 0);
        addMouseListeners();
    }

    public void mousePressed(MouseEvent e) {
        GObject obj = getElementAt(1, 1);
        remove(obj);
    }

}
```


```
public class SimpleRect extends GraphicsProgram {

    public void run() {
        GRect r = null;
        r = new GRect(300, 300);
        r.setColor(Color.MAGENTA);
        add(r, 0, 0);
        addMouseListeners();
    }

    public void mousePressed(MouseEvent e) {
        GObject obj = getElementAt(1, 1);
        remove(obj);
    }

}
```


```
public class SimpleRect extends GraphicsProgram {

    public void run() {
        GRect r = null;
        r = new GRect(300, 300);
        r.setColor(Color.MAGENTA);
        add(r, 0, 0);
        addMouseListeners();
    }

    public void mousePressed(MouseEvent e) {
        GObject obj = getElementAt(1, 1);
        remove(obj);
    }

}
```


Memory

Instance Variables

canvas

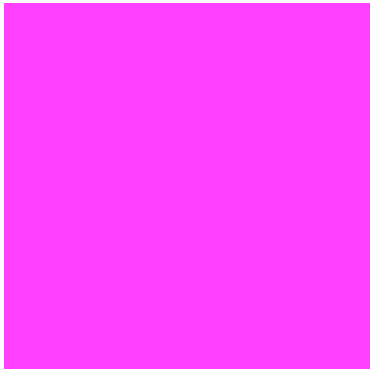
run

r

www.memory.com/12

Object Memory

www.memory.com/12



Memory

Instance Variables

canvas

memory.com/12

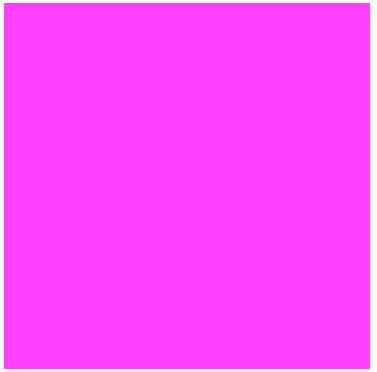
run

r

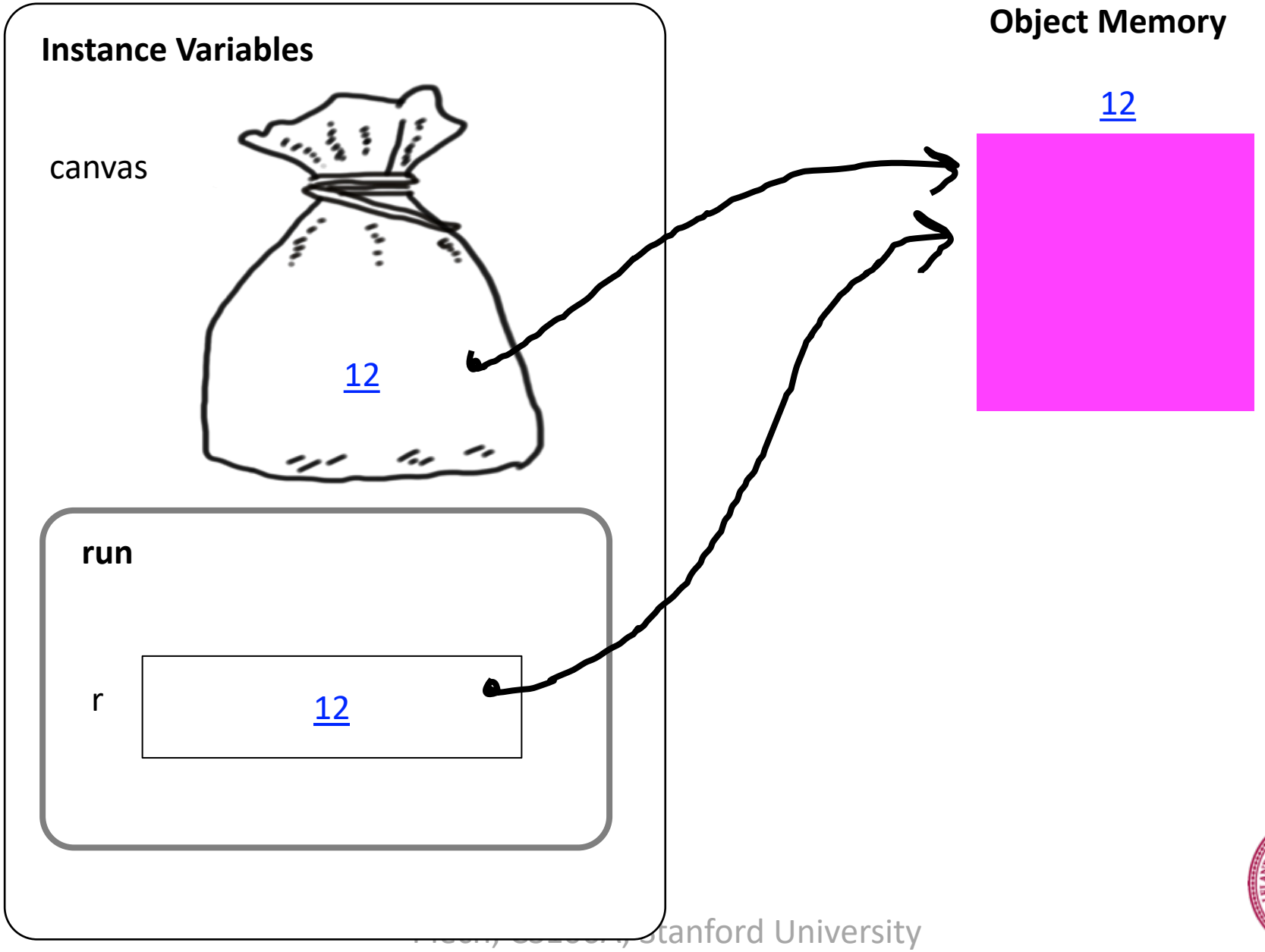
www.memory.com/12

Object Memory

www.memory.com/12



Memory

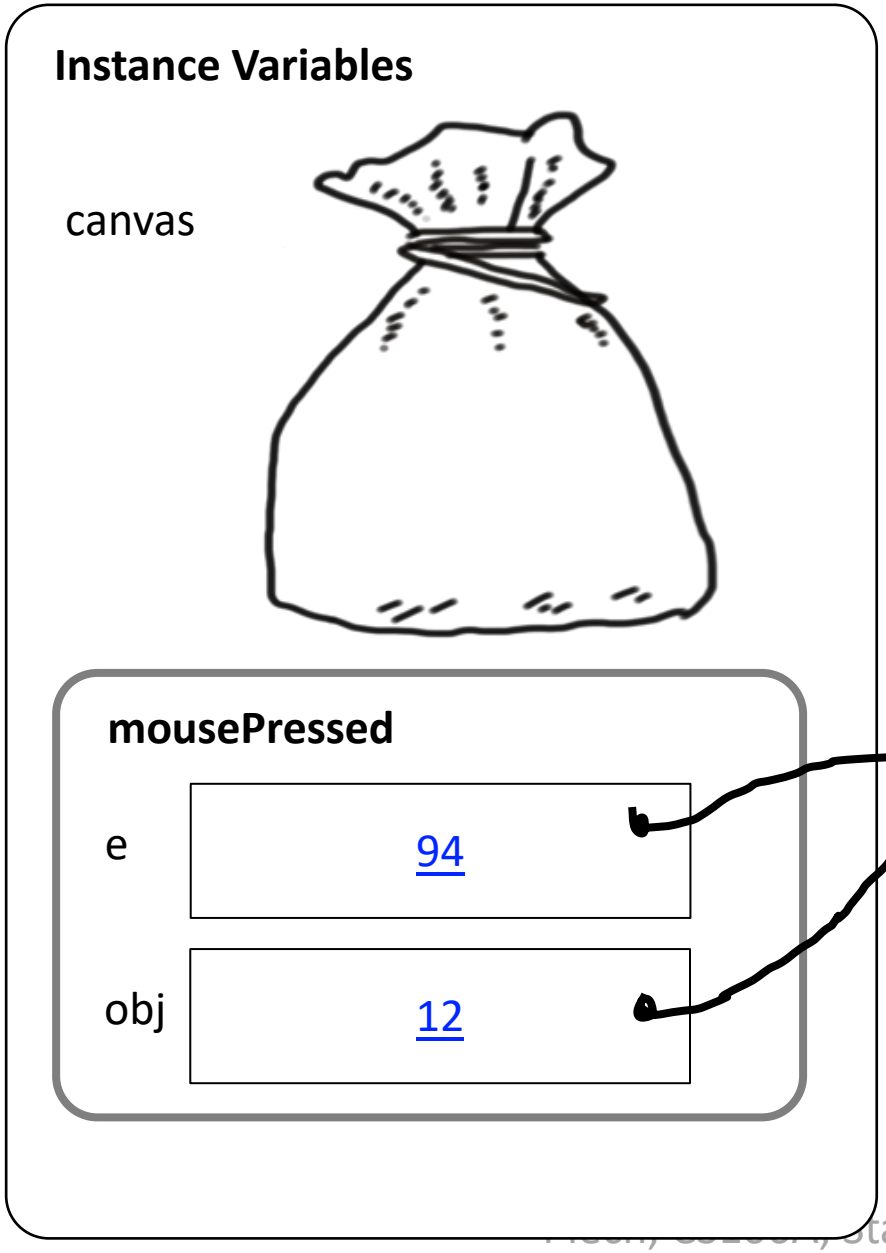


```
public class SimpleRect extends GraphicsProgram {

    public void run() {
        GRect r = null;
        r = new GRect(300, 300);
        r.setColor(Color.MAGENTA);
        add(r, 0, 0);
        addMouseListeners();
    }

    public void mousePressed(MouseEvent e) {
        GObject obj = getElementAt(1, 1);
        remove(obj);
    }

}
```


```
public class SimpleRect extends GraphicsProgram {

    public void run() {
        GRect r = null;
        r = new GRect(300, 300);
        r.setColor(Color.MAGENTA);
        add(r, 0, 0);
        addMouseListeners();
    }

    public void mousePressed(MouseEvent e) {
        GObject obj = getElementAt(1, 1);
        remove(obj);
    }

}
```


```
public class SimpleRect extends GraphicsProgram {

    public void run() {
        GRect r = null;
        r = new GRect(300, 300);
        r.setColor(Color.MAGENTA);
        add(r, 0, 0);
        addMouseListeners();
    }

    public void mousePressed(MouseEvent e) {
        GObject obj = getElementAt(1, 1);
        remove(obj);
    }

}
```

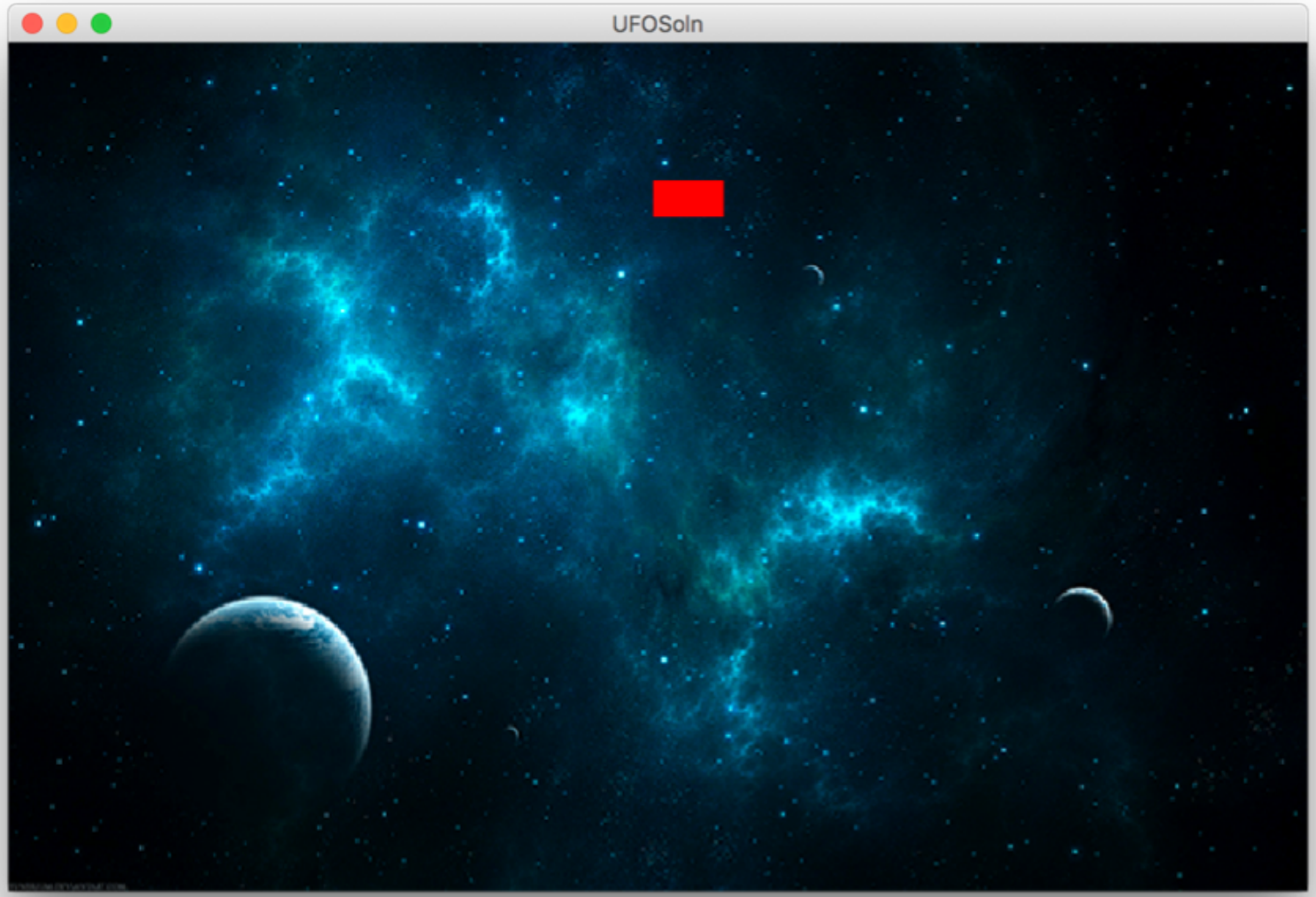

Memory


Instance Variables
canvas
mousePressed
e
94
obj
12
Heap
12
94
x = 72
y = 94
time = 192332123

What does an object store?

Objects store addresses
(which are like URLs)

Finish Up



Learning Goals

1. Be able to write a large program
2. Be able to trace memory with references