

Days and Pixels

Name: _____

Date: _____

Nested data

```
record DayTemps(String day, double[] temp) {
    double high() {
        double best = temp[0];
        for(double t: temp) {
            if(t > best) { best = t; }
        }
        return best;
    }
}

record TempLog(DayTemps[] days) {
    String hottestDay() {
        DayTemps hottest = days[0];
        for(DayTemps d: days) {
            if(d.high() > hottest.high()) {
                hottest = d;
            }
        }
        return hottest.day;
    }
}

double[] monT = { 58.5, 66.0, 61.0 };
double[] tueT = { 61.5, 72.5, 70.0 };
double[] wedT = { 64.0, 69.5, 63.0 };
DayTemps mon = new DayTemps("Mon", monT);
DayTemps tue = new DayTemps("Tue", tueT);
DayTemps wed = new DayTemps("Wed", wedT);
DayTemps[] week = { mon, tue, wed };
TempLog log = new TempLog(week);
String hot = log.hottestDay();
```

Q1. What is hot? _____ How many times does high() run? _____

Q2. Fill in memory just before log.hottestDay() is called.

NAMES (THE STACK)

HEAP

top level
monT = @1001
tueT = @1002
wedT = @1003
mon = _____
tue = _____
wed = _____
week = _____
log = _____

@1001: [58.5, 66.0, 61.0]
@1002: [61.5, 72.5, 70.0]
@1003: [64.0, 69.5, 63.0]
@1004: _____
@1005: _____
@1006: _____
@1007: _____
@1008: _____

Calls over time

Q3. Draw the stack for the whole run of log.hottestDay(). Each box is one call, above the call that made it, in the order they ran from left to right. Fill in each call as written and the address it runs on, its names (cross out old values), and what it returns. The heap is the one from Q2.

NAMES (THE STACK), OVER TIME →

_____ on _____ best = _____ t = _____ returns _____					
---	--	--	--	--	--

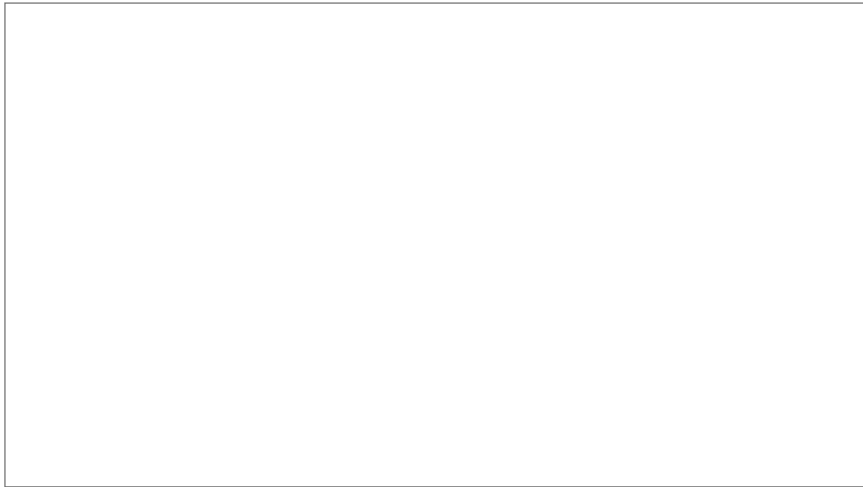
_____ on _____ hottest = _____ d = _____ returns _____

top level (monT = @1001, tueT = @1002, wedT = 2003, ... as above in Q2) hot = _____

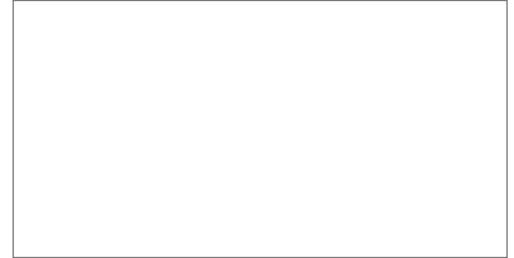
One row, many times

```
Color sky = new Color(120, 180, 255);
Color[] row = { sky, sky, sky, sky, sky };
Color[][] pix = { row, row, row };
// ★
pix[0][0] = Color.RED;    // top left
pix[0][4] = Color.RED;    // top right
pix[2][0] = Color.RED;    // bottom left
pix[2][4] = Color.RED;    // bottom right
Raster framed = new Raster(pix);
```

Q4. Draw memory at ★.



Q5. The four `Color.RED` lines run next. What changes in your diagram? Mark or list just the changes.



Q6. Shade the red pixels of framed.

	0	1	2	3	4
0					
1					
2					

Width and height

Q7. Fill in solid so it builds a width-wide, height-tall grid of one color, with every row its own array. Then fill in the corners.

```
Color[][] solid(int width, int height, Color c) {
    Color[][] pix = new Color[____][____];
    for(int row = 0; row < ____; row += 1) {
        for(int col = 0; col < ____; col += 1) {
            pix[row][col] = c;
        }
    }
    return pix;
}
int width = 4;
int height = 6;
Color[][] pix = solid(width, height, sky);
pix[0][0] = Color.RED;
pix[____][____] = Color.RED;    // top right
pix[____][____] = Color.RED;    // bottom left
pix[____][____] = Color.RED;    // bottom right
Raster corners = new Raster(pix);
```

Q8. Shade corners.

	0	1	2	3
0				
1				
2				
3				
4				
5				

Q9. A classmate's `solid` starts with `new Color[width][height]` and is otherwise the same as yours. What does `solid(4, 6, sky)` do? And `solid(4, 4, sky)`?