Simple Java YEAH Hours

Brahm Capoor and Vrinda Vasavada
What are YEAH hours?

Held soon after each assignment is released

Help you to get an early start on your assignments

Future dates TBA

Slides will be posted!
Roadmap

Review

Assignment overview and tips

Questions
Dropping the mic on Karel

---

Karel taught us a lot of things!
Dropping the mic on Karel

Karel taught us a lot of things!

Control Flow
Dropping the mic on Karel

Karel taught us a lot of things!

Control Flow

Decomposition & Top Down Design
Dropping the mic on Karel

Karel taught us a lot of things!

Control Flow

Decomposition & Top Down Design

Algorithmic Strategy
Control Flow in Karel

```java
for (int i = 0; i < 5; i++) {
    if (beepersPresent()) {
        move();
    } else {
        putBeeper();
    }
}
```

```java
while (frontIsClear()) {
    move();
    putBeeper();
}
```

// do whatever is in the loop 5 times
// what to do if a particular condition is true
// what to do if that condition is false
// do this until a particular condition is false
Control Flow outside Karel

```java
for (int i = 0; i < 100; i++) {
    if (i % 2 == 0) {
        println("Even: "+ i);
    } else {
        println("Odd: "+ i);
    }
}

while (true) {
    if (agentOfChaos()) {
        break;
    }
    println("Good prevails!");
}
```

// do whatever is in the loop 100 times
// what to do if a particular condition is true
// what to do if that condition is false
// loop indefinitely
// savagely immediately end while loop
Control Flow-ception

```java
for (int i = 0; i < 10; i++) {
    for (int j = 0; j < 10; j++) {
        if (i == j) {
            println("i and j are equal!");
        } else {
            int difference = i - j;
            if (difference > 0) {
                println("i is bigger than j by "+ difference + "!");
            } else {
                println("j is bigger than i by "+ difference + "!");
            }
        }
    }
}
```
for (int i = 0; i < 10; i++) {
    for (int j = 0; j < 10; j++) {
        if (i == j) {
            println("i and j are equal!");
        } else {
            int difference = i - j;
            if (difference > 0) {
                println("i is bigger than j by " + difference + "!");
            } else {
                println("j is bigger than i by " + difference + "!");
            }
        }
    }
}
Control Flow-ception

```java
for (int i = 0; i < 10; i++) {
    for (int j = 0; j < 10; j++) {
        if (i == j) {
            println("i and j are equal!");
        } else {
            int difference = i - j;
            if (difference > 0) {
                println("i is bigger than j by "+ difference + "!");
            } else {
                println("j is bigger than i by "+ difference + "!");
            }
        }
    }
}
```
Graphics

GRect rect = new GRect(50, 50, 200, 200);
rect.setFilled(true);
rect.setColor(Color.BLUE);

GOval oval = new GOval(0, 0, getWidth(), getHeight());
oval.setFilled(false);
oval.setColor(Color.GREEN);

GLabel text = new GLabel("banter", 200, 10);
add(text);
add(rect);
add(oval);
Graphics

```java
GRect rect = new GRect(50, 50, 200, 200);
rect.setFilled(true);
rect.setColor(Color.BLUE);

GOval oval = new GOval(0, 0, getWidth(), getHeight());
oval.setFilled(false);
oval.setColor(Color.GREEN);

GLabel text = new GLabel("banter", 200, 10);
add(text);
add(rect);
add(oval);
```

Things to remember

- Coordinates are doubles
- Coordinates are measured from the top left of the screen
- Coordinates of a shape are coordinates of its top left corner
- Coordinates of a label are coordinates of its bottom left corner
- Remember to add objects to the screen!
- Use the [online documentation](#)!
- These are class variables!
**Primitive variables**

```java
int x = 7; // declare and initialize a variable
x = 9;    // change the value of x
x = x + 1; // increment (add 1 to) x. A.K.A. x++
x = x + 2; // add 2 to x. A.K.A. x += 2
x /= 2    // divide x by 2, and truncate result

double d = 3.5;

boolean isThisTrue = true;
isThisTrue = !isThisTrue; // flip isThisTrue
```
Primitive variables

```java
int x = 7;  // declare and initialize a variable
x = 9;     // change the value of x
x = x + 1; // increment (add 1 to) x.  A.K.A. x++
x = x + 2; // add 2 to x.              A.K.A. x += 2
x /= 2     // divide x by 2, and truncate result

double d = 3.5;

boolean isThisTrue = true;
isThisTrue = !isThisTrue;  // flip isThisTrue
```

Things to remember

- The expressive hierarchy:
  boolean < char < int < double

- Compare variables using ==
  if (x == 7) {...}

- Conditional operators: && and ||
  if (x == 7 && y == 6.3)
  if (x == 7 || x == 6)
  Avoid this:
  if (x == 7 || 6)

- Use constants!
  private static final int MY_NUM = 10;
Methods

```java
private returnType methodName(type param1, type param2, ...) {
    // sick code here
}
```

- A method header provides some **guarantees** about the method (what it returns, how many parameters it takes)
- Parameters and return values generalize the methods we saw in Karel to allow the use of variables
- If a method returns something, that something needs to be stored in a variable

```java
returnType storedValue = methodName(/* params */);
```

- Primitive variables passed into a method are **passed by value**
Methods, parameters and variables
Parameters and a return value are both optional!

```java
private returnType methodName(type parameter1, type parameter2,...)
private int returnsInt() {...}
private void drawsRect(int width, int length) {...} //void is no type
public boolean frontIsClear() {...} //look familiar?
```
Example: Methods and Parameters

```java
public void run() {
    println("Choose 2 numbers!");
    int n1 = readInt("Enter n1"); //5
    int n2 = readInt("Enter n2"); //7

    int total = addNumbers(n1, n2);
    println ("The total is " + total);
}

private int addNumbers(int num1, int num2) {
    int sum = num1 + num2; //12
    return sum;
}
```

Example:

```java
addNumbers(n1, n2)
```

```
num1 = 5, num2 = 7
sum = 12
```

run()

GET n1 AND n2

addNumbers(n1, n2)

total = 12

PRINT RESULT
```
Variable scope

Variables live inside the block in which they’re declared

```java
for (int i = 0; i < 5; i++) {
    int y = i * 4;
}
```

```java
i = 3; // Error!
```

```java
y = 2; // Error!
```

```java
... // in some code far, far away
int y = 0;
for (int i = 0; i < 5; i++) {
    y = i * 4;
}
```

```java
y = 2;
```
private int multipleReturns(int x) {
    if (x == 5) {
        return 0;
    }
    return 1; // this only happens if x != 5
    return 5; // never gets to this line
}

// note: every path through the method ends with a single return statement

// note: a function ends immediately after it returns

---
Assignment 2!
High level overview

- Due Monday 23/4/2017
- 10 Problems
- 2 warmups
- 3 Graphics Programs
- 4 Console Programs
- 1 Debug Practice
Problem 1: Draw a blue, filled rectangle in the center of the screen with dimensions 350 x 270

Questions to ask yourself:

1. How do I find the center of the screen?
2. Given the location of the center of the screen, where should I put the rectangle?

Useful ideas from lecture

- Coordinates are measured from the top left of shapes and the window

Useful methods:

- getWidth() tells you the width of the canvas
- getHeight() tells you the height of the canvas
- rect.getWidth() tells you the width of rect
- rect.getHeight() tells you the height of rect
- See lecture/video and GRect documentation for more!
Problem 2 Print out a countdown down from 10 to 1 and then print “Liftoff!”

Questions to ask yourself:

1. What sort of control flow structure best suits this problem?
2. What’s a nice way to represent what the current number is?

Useful ideas from lecture

- You can use the variables inside for loops!
Problem 3 Pythagorean Theorem

---

Questions to ask yourself:

1. What data type should I store numbers as?
2. How many variables do I need?

Useful ideas from lecture

- Primitive data types
- The expressive hierarchy

Useful methods

- `math.sqrt(n)` tells you the square root of `n`
- Look at the [lecture](#) for more!

Enter values to compute the Pythagorean theorem.

```
a: 3.5
b: 4.2
c = 5.4671747731346585
```
Problem 4: Keeping track of the largest and smallest

Questions to ask yourself:

1. What sorts of things do you need to store?
2. How do you initialize variables?

Useful ideas from lecture

- Loop structures
- Variable scope
- Edge cases
- Sentinel values

This program finds the largest and smallest numbers.

? 11
? 17
? 42
? 9
? -3
? 35
? 0
smallest: -3
largest: 42
Problem 5  Hailstone sequence

Questions to ask yourself:

1. What sorts of things do you need to store?
2. How do you initialize variables?

Useful ideas from lecture

- Loop structures
- Variable scope
- Edge cases
- Sentinel values

Enter a number: 17
17 is odd, so I make $3n + 1$: 52
52 is even so I take half: 26
26 is even so I take half: 13
13 is odd, so I make $3n + 1$: 40
40 is even so I take half: 20
20 is even so I take half: 10
10 is even so I take half: 5
5 is odd, so I make $3n + 1$: 16
16 is even so I take half: 8
8 is even so I take half: 4
4 is even so I take half: 2
2 is even so I take half: 1
The process took 12 to reach 1
Problem 6 Draw a pyramid!

Questions to ask yourself:

1. What sort of control flow structure best suits this problem?
2. How do I decompose this problem?
3. What information do I need to draw a row and the bricks inside a row?

Useful ideas from lecture

- You can use the variables inside for loops!
- You can nest for loops!
- This [checkerboard example](#) from lecture

Useful methods

- `getWidth()` tells you the width of the canvas
- `getHeight()` tells you the height of the canvas
- `rect.getWidth()` tells you the width of `rect`
- `rect.getHeight()` tells you the height of `rect`
- See [lecture](#) and `GRect documentation` for more!

** remember that coordinates should be `doubles`
Problem 7 Bullseye!

Questions to ask yourself:

1. Can this problem be decomposed?
2. What information is needed to draw each circle?

Useful ideas from lecture

- How methods can be used to **encapsulate repeated functionality**

Useful methods

- See [lecture](#) and [GOval documentation](#) for more!
Questions to ask yourself:

1. Can this problem be decomposed?
2. What information is needed to draw each rectangle?

Useful ideas from lecture

- How methods can be used to **encapsulate repeated functionality**
- Remember that a label’s coordinate is its **bottom left corner**

Useful methods

- `label.getAscent()` tells you the **distance** between the **baseline** of the label and the top of the label. This is useful for centering!
- See [lecture](#) and GRect [documentation](#) and GLabel [documentation](#) for more!
Problem 9 isDivisibleBy

---

Questions to ask yourself:

1. Which of the various cases is the most conclusive? When should I check them?
2. How can I effectively test a method?

Useful ideas from lecture

- How can we use the remainder operator
- Testing for edge cases! (Look at the run method)
Problem 10 Debugging (You can do this after Wednesday’s lecture)

Questions to ask yourself:

1. Where should I be placing breakpoints?
2. Once I’ve identified a misbehaving variable, how should I correct it?

Useful ideas from lecture

● How to use the debugger (step into/out of methods)
A last few tips and tricks

● “Write a GraphicsProgram SubClass”: Don’t worry about what this means! (You’ll learn a lot about this in a few weeks)
● **Draw things on paper for Graphics Programs**
● **Use Top Down Decomposition wherever you can**
● Go to the LaIR (6:50-10:50 PM, First floor of Tresidder)!
● Incorporate your IG feedback!
● Use the debugger!
● Work on extensions
Questions?