Nested Loops

Chris Piech

CS106A, Stanford University
A Variable love story

By Chris
Once upon a time...
X was looking for love!

```java
int x = 5;
if(lookingForLove()) {
    int y = 5;
}
println(x + y);
```

\[ X \] was looking for love!
```java
int x = 5;
if(lookingForLove()) {
    int y = 5;
}
println(x + y);
X was looking for love!
```
```java
int x = 5;
if(lookingForLove()) {
    int y = 5;
}
println(x + y);
```

X was looking for love!
And met y

```
int x = 5;
if(lookingForLove()) {
    int y = 5;
}
println(x + y);
```
int x = 5;
if(lookingForLove()) {
    int y = 5;
}
println(x + y);

Hi, I'm y
“Wow!”
```java
int x = 5;
if(lookingForLove()) {
    int y = 5;
}
println(x + y);
```

Wow

```plaintext
5
\[\text{x}\]
```

```plaintext
5
\[\text{y}\]
```
```java
int x = 5;
if(lookingForLove()) {
    int y = 5;
}
println(x + y);
```

We have so much in common
int x = 5;
if(lookingForLove()) {
    int y = 5;
}
println(x + y);

We both have value 5!
```java
int x = 5;
if(lookingForLove()) {
    int y = 5;
}
println(x + y);

Maybe one day
we can...
```
int x = 5;
if(lookingForLove()) {
    int y = 5;
}
println(x + y);

println together?
They got along

```
int x = 5;
if(lookingForLove()) {
    int y = 5;
}
println(x + y);
```
It was a beautiful match...
But then tragedy struck.
int x = 5;
if(lookingForLove()) {
    int y = 5;
}
println(x + y);

Tragedy Struck
Tragedy Struck

```java
int x = 5;
if(lookingForLove()) {
    int y = 5;
}
println(x + y);
```

5

5

x

y
int x = 5;
if(lookingForLove()) {
    int y = 5;
}
println(x + y);
Noooooooooooooooooo!
You see...
int x = 5;
if(lookingForLove()) {
    int y = 5;
}
println(x + y);
int x = 5;
if(lookingForLove()) {
    int y = 5;
}
println(x + y);

All variables declared inside that block..
int \ x = 5;
if(lookingForLove()) {
    int \ y = 5;
}
println(x + y);

Get deleted from memory!
Since y was declared in the if-block

```java
int x = 5;
if(lookingForLove()) {
    int y = 5;
}
println(x + y);
```

5

RIP y

Piech, CS106A, Stanford University
It gets deleted from memory here

```java
int x = 5;
if(lookingForLove()) {
    int y = 5;
}
println(x + y);
```

![Diagram of memory deletion](image)
int x = 5;
if(lookingForLove()) {
    int y = 5;
}
println(x + y);
```java
int x = 5;
if (lookingForLove()) {
    int y = 5;
}
println(x + y);
```

Error. Undefined variable y.

And doesn’t exist here
The End
Sad times 😞
public void run(){
    double v = 8;
    if(condition) {
        v = 4;
        ... some code
    }
    ... some other code
}
public void run(){
    double v = 8;
    if (condition) {
        v = 4;
        ... some code
    }
    ... some other code
}
public void run(){
    double v = 8;
    if(condition){
        v = 4;
        ... some code
    }
    ... some other code
}
public void run(){
    double v = 8;
    if(condition){
        v = 4;
        ... some code
    }
    ... some other code
}
public void run(){
    double v = 8;
    if(condition) {
        v = 4;
        ... some code
    }
    ... some other code
}
public void run(){
    double v = 8;
    if (condition) {
        v = 4;
        ... some code
    }
    ... some other code
}

Live until end of their code block

It dies here (at the end of its code block)
public void run() {
    double v = 8;
    if (condition) {
        v = 4;
        ... some code
    }
    ... some other code
}

It dies here (at the end of its code block)
public void run(){
    ... some code
    if(condition) {
        int w = 4;
        ... some code
    }
    ... some other code
}
public void run(){
    ... some code
    if(condition) {
        int w = 4;
        ... some code
    }
    ... some other code
}
A Variable love story

Chapter 2
The programmer fixed her bug
int x = 5;
if(lookingForLove()) {
    int y = 5;
    println(x + y);
}

x was looking for love!
```java
int x = 5;
if (lookingForLove()) {
    int y = 5;
    println(x + y);
}
```

x was definitely looking for love...
int x = 5;
if(lookingForLove()) {
    int y = 5;
    println(x + y);
}

x met y
int x = 5;
if(lookingForLove()) {
    int y = 5;
    println(x + y);
}
The story had a happy ending!
The **scope** of a variable refers to the section of code where a variable can be accessed.

- **Scope starts** where the variable is declared.
- **Scope ends** at the termination of the inner-most code block in which the variable was defined.

- A **code block** is a chunk of code between `{ }` brackets.
Back to our regularly scheduled program...
How would you println “Stanford rocks socks” 100 times
public void run() {
    for(int i = 0; i < 100; i++) {
        println(“Stanford rocks socks!”);
    }
}
For Loop Redux

```java
for (int i = 0; i < 100; i++) {
    println("Stanford rocks socks!");
}
```

- This line is run once, just before the for loop starts.
- Enters the loop if this condition passes.
- This line is run each time the code gets to the end of the 'body'.

Piech, CS106A, Stanford University
```java
for (int i = 0; i < 3; i++) {
    println("Stanford rocks socks!");
}
```
for(int i = 0; i < 3; i++) {
    println("Stanford rocks socks!");
}
for (int i = 0; i < 3; i++) {
    println("Stanford rocks socks!");
}
for (int i = 0; i < 3; i++) {
    println("Stanford rocks socks!");
}

i  0
for(int i = 0; i < 3; i++) {
    println("Stanford rocks socks!");
}
For Loop Redux

```
for(int i = 0; i < 3; i++) {
    println("Stanford rocks socks!");
}
```

![Output screenshot showing "Stanford rocks socks"]
For Loop Redux

```
for (int i = 0; i < 3; i++) {
    println("Stanford rocks socks!");
}
```

```
for (int i = 0; i < 3; i++) {
    println("Stanford rocks socks!");
}
```
for (int i = 0; i < 3; i++) {
    println("Stanford rocks socks!");
}
For Loop Redux

```java
for(int i = 0; i < 3; i++) {
    println("Stanford rocks socks!");
}
```

Output:

```
Stanford rocks socks
Stanford rocks socks
```
for loop

```java
for(int i = 0; i < 3; i++) {
    println("Stanford rocks socks!");
}
```
For Loop Redux

```java
for (int i = 0; i < 3; i++) {
    println("Stanford rocks socks!");
}
```

Output:
Stanford rocks socks
Stanford rocks socks
Stanford rocks socks
For Loop Redux

```
for(int i = 0; i < 3; i++) {
    println("Stanford rocks socks!");
}
```

Output:
Stanford rocks socks
Stanford rocks socks
Stanford rocks socks
for (int i = 0; i < 3; i++) {
    println("Stanford rocks socks!");
}
for(int i = 0; i < 3; i++) {
    println("Stanford rocks socks!");
}
for(int i = 0; i < 3; i++) {
    println("Stanford rocks socks!");
}
You can use the for loop variable
How would you print the first 100 even numbers?
Printing Even Numbers

0
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
34
36
38
for(int i = 0; i < NUM_NUMS; i++) {
    println(i * 2);
}
for(int i = 0; i < 3; i++) {
    println(i * 2);
}
Printing Even Numbers

```java
for(int i = 0; i < 3; i++) {
    println(i * 2);
}
```
Printing Even Numbers

```
for(int i = 0; i < 3; i++) {
    println(i * 2);
}
```
Printing Even Numbers

```
for(int i = 0; i < 3; i++) {
    println(i * 2);
}
```

For Loop Redux
Printing Even Numbers

for(int i = 0; i < 3; i++) {
    println(i * 2);
}

Piech, CS106A, Stanford University
Printing Even Numbers

```
for(int i = 0; i < 3; i++) {
    println(i * 2);
}
```
For Loop Redux

```java
for(int i = 0; i < 3; i++) {
    println(i * 2);
}
```
Printing Even Numbers

```
for(int i = 0; i < 3; i++) {
    println(i * 2);
}
```

<table>
<thead>
<tr>
<th>i</th>
<th>1</th>
</tr>
</thead>
</table>

```
0
2
```
Printing Even Numbers

for(int i = 0; i < 3; i++) {
    println(i * 2);
}

i  2

0
2
Printing Even Numbers

for(int i = 0; i < 3; i++) {
    println(i * 2);
}

0
2
Printing Even Numbers

```
for(int i = 0; i < 3; i++) {
    println(i * 2);
}
```

For Loop Redux

| i | 2 |

0
2
4
Printing Even Numbers

\[
\text{for (int } i = 0; i < 3; i++) \{ \\
\quad \text{println}(i * 2);
\}
\]
Printing Even Numbers

\[
\text{for} (\text{int } i = 0; i < 3; i++) \{
    \text{println}(i \times 2);
\}
\]
Printing Even Numbers

\[
\begin{array}{c|c}
\text{i} & 3 \\
\end{array}
\]

```java
for(int i = 0; i < 3; i++) {
    println(i * 2);
}
```

0
2
4
for(int i = 0; i < 3; i++) {
    println(i * 2);
}

0
2
4
Types of Programs

Program

- Karel Program
  - SuperKarel Program

- Console Program

- Graphics Program

Piech, CS106A, Stanford University
Graphics Programs
GRect is a variable type that stores a rectangle.

As an example, the following `run` method displays a blue square:

```java
public void run() {
    GRect rect = new GRect(50, 50, 200, 200);
    rect.setFilled(true);
    rect.setColor(Color.BLUE);
    add(rect);
}
```
Graphics Coordinates

0,0

- 40,20
- 40,120
- 120,40

getHeight();

get getWidth();
The `GOval` class represents an elliptical shape defined by the boundaries of its enclosing rectangle.

As an example, the following `run` method creates the largest oval that fits within the canvas:

```
public void run() {
    GOval oval = new GOval(0, 0, getWidth(), getHeight());
    oval.setFilled(true);
    oval.setColor(Color.GREEN);
    add(oval);
}
```
Graphics Variable Types

```
GRect myRect = new GRect(350, 270);
```
## Primitives vs Classes

<table>
<thead>
<tr>
<th>Primitive Variable Types</th>
<th>Class Variable Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>GRect</td>
</tr>
<tr>
<td>double</td>
<td>GOval</td>
</tr>
<tr>
<td>char</td>
<td>GLine</td>
</tr>
<tr>
<td>boolean</td>
<td>...</td>
</tr>
</tbody>
</table>

### Class variables:
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
Goal
Milestone 1
Milestone 2