Nested Loops

Chris Piech

CS106A, Stanford University
A Variable love story

By Chris
Once upon a time...
int x = 5;
if(lookingForLove()) {
    int y = 5;
}
println(x + y);

X was looking for love!
X was looking for love!

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

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

X was looking for love!

x was definitely looking for love

5

x
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**

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

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

And met y

```
5
\_\_\_\_\_
X

5
\_\_\_\_\_
Y
```

println together?
They got along

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

```
x 5

y 5
```

Piech, CS106A, Stanford University
It was a beautiful match...
But then tragedy struck.
Tragedy Struck

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

```plaintext
5
\[ x \]
5
\[ y \]
```
Tragedy Struck

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

![Diagram](image-url)
Tragedy Struck

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

Piech, CS106A, Stanford University
Noooooooooooooooooooooo!
You see...
When a program exits a code block...

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

When a program exits a code block...

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

Piech, CS106A, Stanford University
All variables declared inside that block..

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

5

RIP y

Piech, CS106A, Stanford University
```java
int x = 5;
if(lookingForLove()) {
    int y = 5;
}
println(x + y);
```

Get deleted from memory!
int x = 5;
if(lookingForLove()) {
    int y = 5;
}
println(x + y);

Since y was declared in the if-block

\[
\begin{align*}
\text{RIP}
\end{align*}
\]
It gets deleted from memory here

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

It gets deleted from memory here
```java
int x = 5;
if(lookingForLove()) {
    int y = 5;
}
println(x + y);
```
int x = 5;
if(lookingForLove()) {
    int y = 5;
}
println(x + y);

Error. Undefined variable y.
The End
Sad times 😞
public void run(){
    double v = 8;
    if (condition) {
        v = 4;
        ... some code
    }
    ... some other code
}
**Variables have a lifetime (called scope)**

```java
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
}

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)
Example 2

```java
public void run(){
    ... some code
    if(condition) {
        int w = 4;
        ... some code
    }
    ... some other code
}
```

This is the scope of w
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!
x was looking for love...

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

x was definitely looking for love

5

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

```
5
\_\_\_
\_\_\_
x

5
\_\_\_
\_\_\_
y
```
Since they were both “in scope”

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

![Diagrams of x and y](image.png)
The story had a happy ending!
Scope Formally

- 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

```
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
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

\[
\begin{array}{|c|}
\hline
i & 0 \\
\hline
\end{array}
\]

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

Piech, CS106A, Stanford University
For Loop Redux

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

i 0

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

Stanford rocks socks
For Loop Redux

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

Stanford rocks socks
For Loop Redux

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

Stanford rocks socks
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!");
}
```

Stanford rocks socks
Stanford rocks socks
For Loop Redux

```
for(int i = 0; i < 3; i++) {
    println("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 Loop Redux

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

```
For Loop Redux

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!");
}
You can use the for loop variable
How would you println 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

Piech, CS106A, Stanford University
Printing Even Numbers

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

```java
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);
}
```

Piech, CS106A, Stanford University
Printing Even Numbers

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

```
| i | 0 |
```

Piech, CS106A, Stanford University
Printing Even Numbers

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

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

For Loop Redux

0
Printing Even Numbers

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

For Loop Redux

0
Printing Even Numbers

\[
\begin{array}{c|c}
\text{i} & 1 \\
\end{array}
\]

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

0
2
Printing Even Numbers

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

For Loop Redux

0
2
Printing Even Numbers

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

For Loop Redux

```
0
2
```
Printing Even Numbers

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

<table>
<thead>
<tr>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
</tr>
</tbody>
</table>

0
2
4
Printing Even Numbers

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

i 3

0
2
4
Printing Even Numbers

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

For Loop Redux

0
2
4
Printing Even Numbers

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

For Loop Redux

<table>
<thead>
<tr>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

```javascript
0
2
4
```
for (int i = 0; i < 3; i++) {
    println(i * 2);
}
Types of Programs

- Program
  - Karel Program
    - SuperKarel Program
  - Console Program
  - Graphics Program
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
- 120,40
- 40,120

getHeight();

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:

```java
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);
```

Piech, CS106A, Stanford University
# 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
Milestone 3