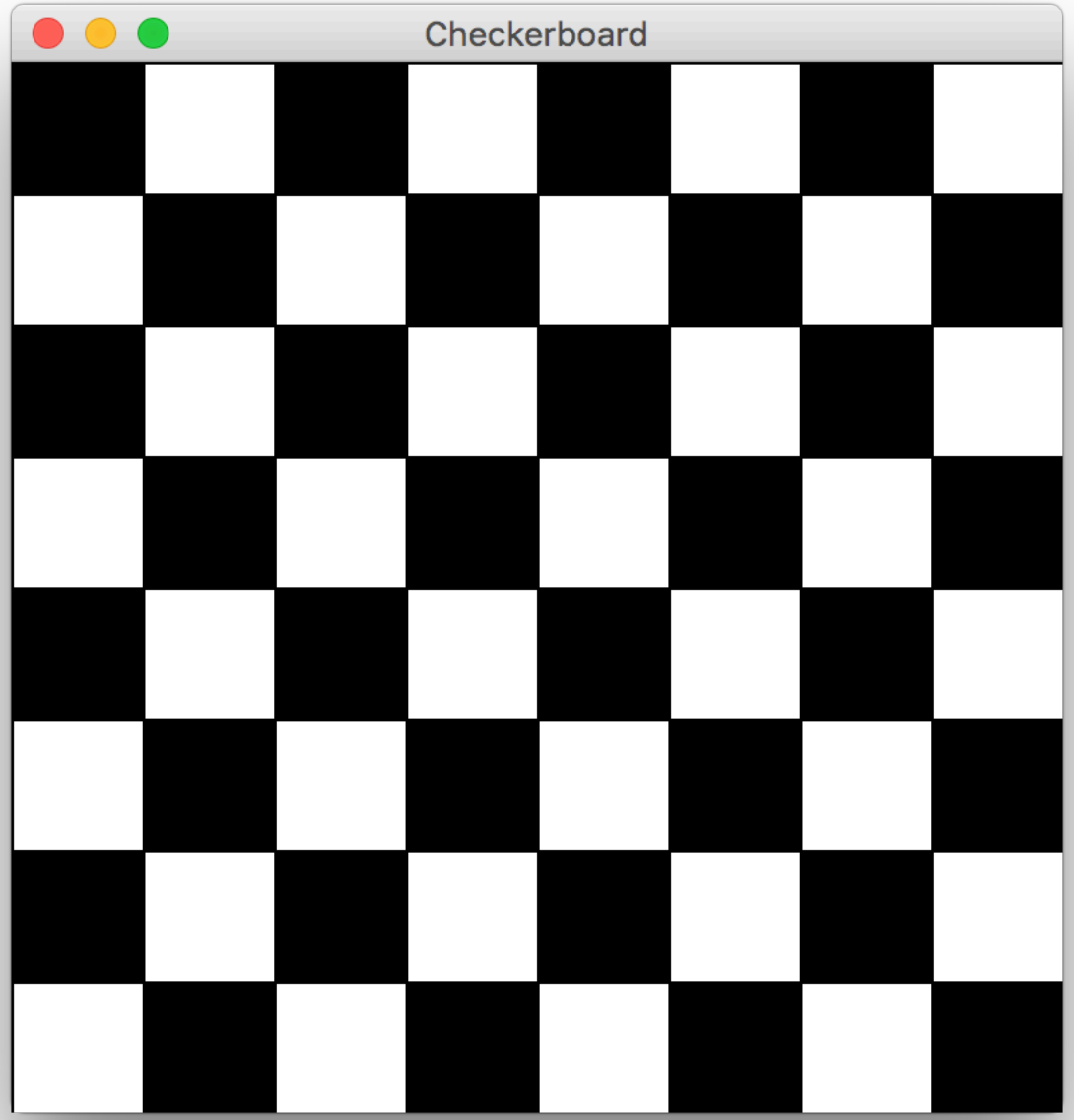
Simple Java

Chris Piech
CS106A, Stanford University


Checkerboard

Let’s fix an old program

Review: Operations

- Operations on numerical types
- Operations:

  `+` "addition"
  `-` "subtraction"
  `*` "multiplication"
  `/` "division" (different for **`int`** vs. **`double`**)
  `%` "remainder"

- Precedence (in order):

  `()` highest
  `*, /, %`
  `+, -` lowest

Operators in same precedence category evaluated left to right

Expressions Short Hand

```
int x = 3;

x = x + 1;   x += 1;   x++;

x = x + 5;   x += 5;

x = x - 1;   x -= 1;   x--;

x = x * 3;   x *= 3;

x = x / 2;   x /= 2;
```

Review: Boolean Expressions

- Boolean expression is just a ***test*** for a condition
  - Essentially, evaluates to **`true`** or **`false`**

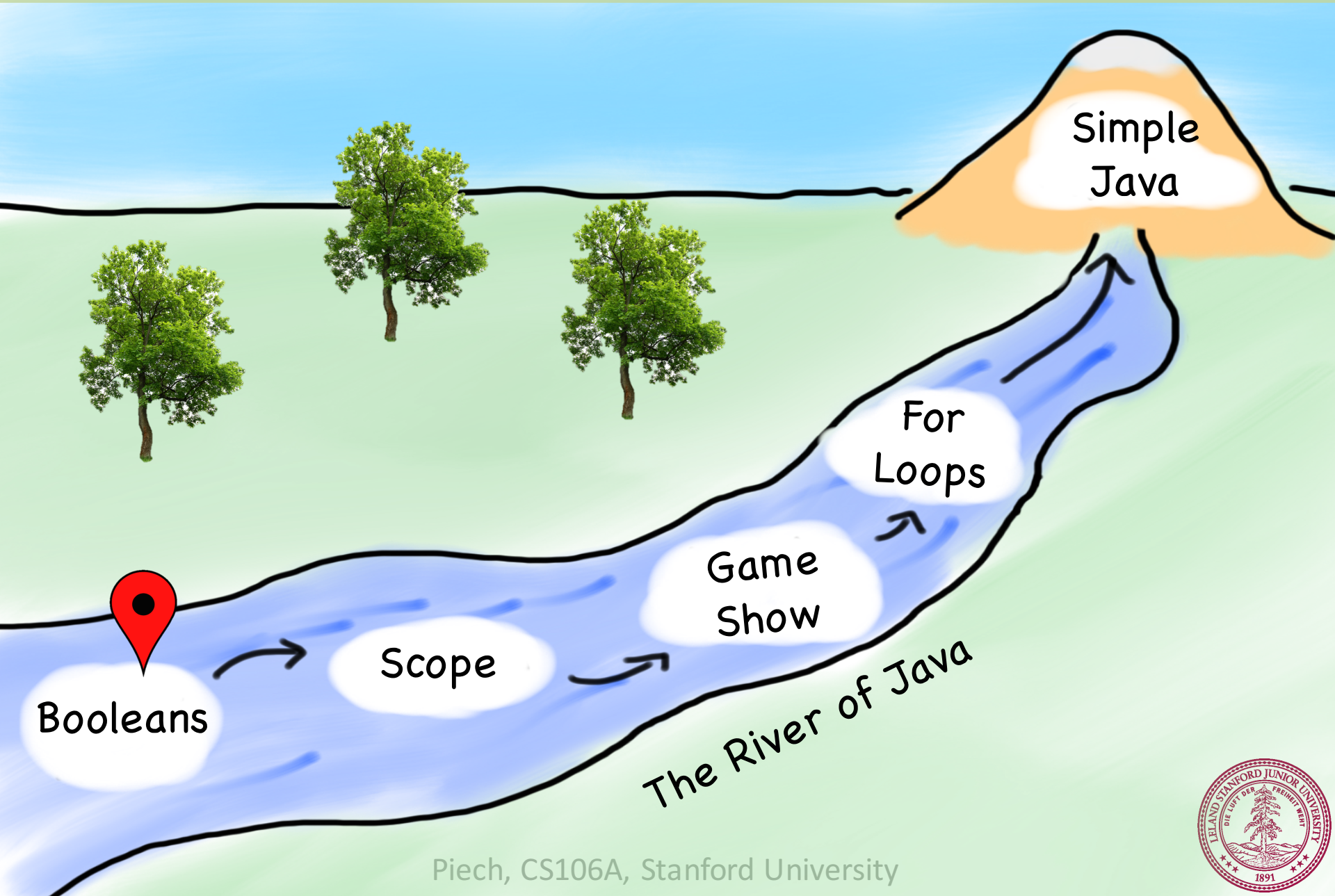
- Value comparisons:

==	"equals"	(note: not single **=**)
!=	"not equals"	(cannot say **<>**)
>	"greater than"	
<	"less than"	
>=	"greater than or equal to"	
<=	"less than or equal to"	

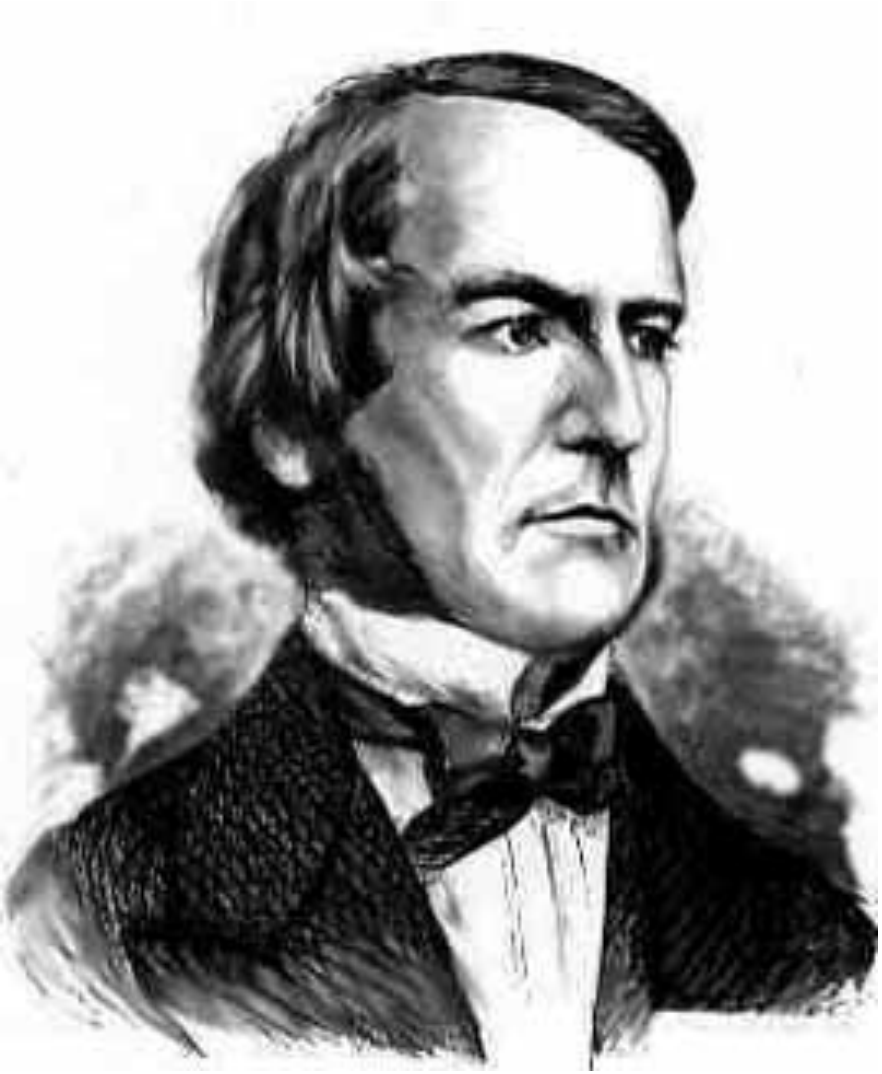
Today's Goal

1. How to use constants
2. Basics of boolean variables
3. Understand For loops
4. Know variable scope

Today's Route



George Boole



Boolean variable type

Boolean Expressions

Value comparisons (in order of precidence):

`!` "not"

`!p`

If p is true then !p is false (and vice versa)

`&&` "and"

`p && q`

Evaluates to true if both sides are true

`||` "or"

`p || q`

Evaluates to true if either p or q (or both) are true

```
boolean p = (x != 1) || (x != 2);
boolean p = (x != 1) && (X != 2);
```

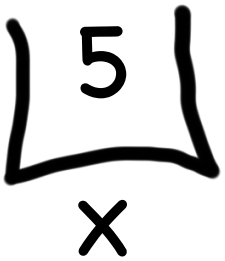
A Variable love story

By Chris

Once upon a time…

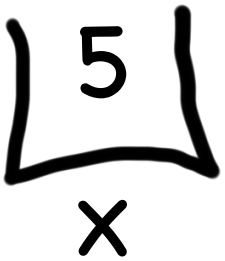
X was looking for love!

```
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

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

And met y

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

And met y

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

Hi, I'm y

“Wow!”

And met y

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

Wow

And met y

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

We have so much in common

And met y

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

We both have value 5!

And met y

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

Maybe one day we can…

And met y

```
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.

Tragedy Struck

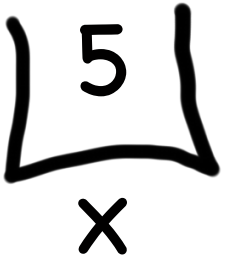
```
int x = 5;
if(lookingForLove()) {
    int y = 5;
}
println(x + y);
```

Tragedy Struck

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

Tragedy Struck

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

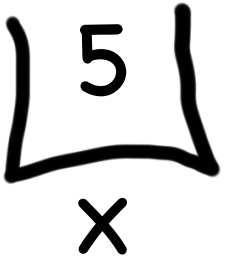


Nooooooooooooooooo!

You see…

When a program exits a code block…

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

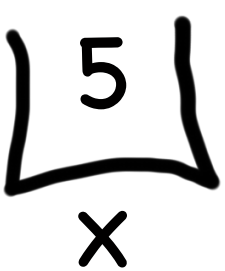


All variables declared inside that block..

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

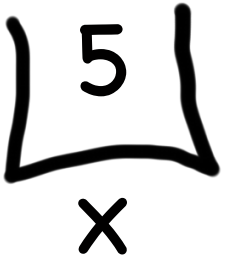
Get deleted from memory!

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



Since y was declared in the if-block

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

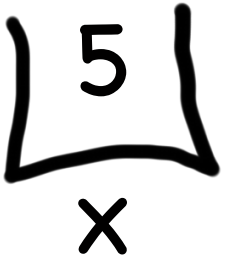


It gets deleted from memory here

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

And doesn't exist here

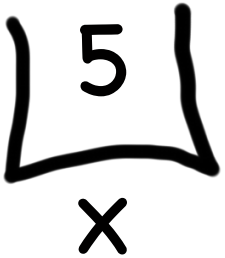
```
int x = 5;
if(lookingForLove()) {
    int y = 5;
}
println(x + y);
```



And doesn't exist here

```
int
if(l
    i
}
println(x + y);
```

Error. Undefined variable y.



The End

Sad times ☹

Variables have a lifetime (called scope)

```
public void run(){
    double v = 8;
    if(condition){
        v = 4;
        … some code
    }
   … some other code
}
```

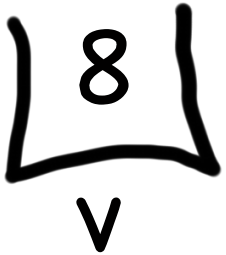
Variables have a lifetime (called scope)

```
public void run(){
    double v = 8;
    if(condition){
        v = 4;
        … some code
    }
    … some other code
}
```

Vars come to existence when declared

```
public void run(){
    double v = 8;
    if(condition){
        v = 4;
        … some code
    }
    … some other code
}
```

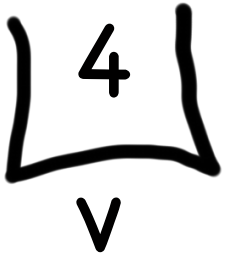
Comes to life here



Live until end of their code block

```
public void run(){
    double v = 8;
    if(condition){
        v = 4;
        ... some code
    }
    ... some other code
}
```

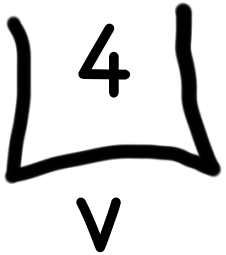
This is the **inner most** code block in which it was declared….



Live until end of their code block

```
public void run(){
    double v = 8;
    if(condition){
        v = 4;
        ... some code
    }
   ... some other code
}
```

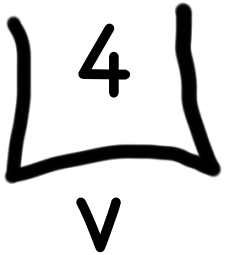
Still alive here…



Live until end of their 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)



Live until end of their 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

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

This is the scope of w

Example 2

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

w comes to life here

w dies here (at the end of its code block)

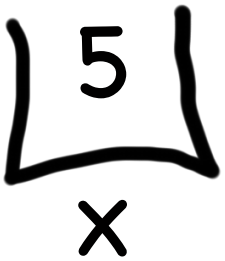
A Variable love story

Chapter 2

The programmer fixed her bug

x was looking for love!

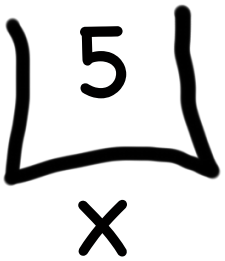
```
int x = 5;
if(lookingForLove()) {
    int y = 5;
    println(x + y);
}
```



x was looking for love…

x was definitely looking for love

```
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);
}
```

Since they were both "in scope"

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

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

Game Show

GameShow

```
Welcome to the CS106A game show!
Choose a door and win a prize
Door: 2
You chose door 2
You win $
```

Choose a Door

```
int door = readInt("Door: ");
// while the input is invalid
while(door < 1 || door > 3) {
   // tell the user the input was invalid
   println("Invalid door!");
   // ask for a new input
   door = readInt("Door: ");
}
```

`||`	or
`&&`	and

The Door Logic

```
int prize = 3;
if(door == 1) {
   prize = 2 + 9 / 10 * 100;
} else if(door == 2) {
   boolean locked = prize % 2 != 1;
   if(!locked) {
      prize += 7;
   }
} else if(door == 3) {
   prize++;
}
```

How would you println “Nick rocks socks” 100 times

```
println("Nick rocks socks!");
println("Nick rocks socks!");
println("Nick rocks socks!");
println("Nick rocks socks!");
println("Nick rocks socks!");
println("Nick rocks socks!");
println("Nick rocks socks!");
println("Nick rocks socks!");
println("Nick rocks socks!");
println("Nick rocks socks!");
println("Nick rocks socks!");
println("Nick rocks socks!");
println("Nick rocks socks!");
println("Nick rocks socks!");
println("Nick rocks socks!");
println("Nick rocks socks!");
```

For Loop Redux

This line is run once, just before the for loop starts

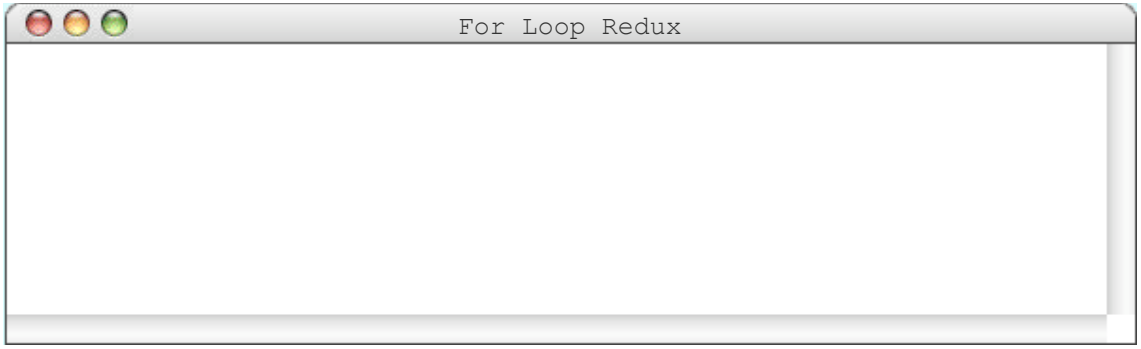
Enters the loop if this condition passes

This line is run each time the code gets to the end of the 'body'

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

For Loop Redux

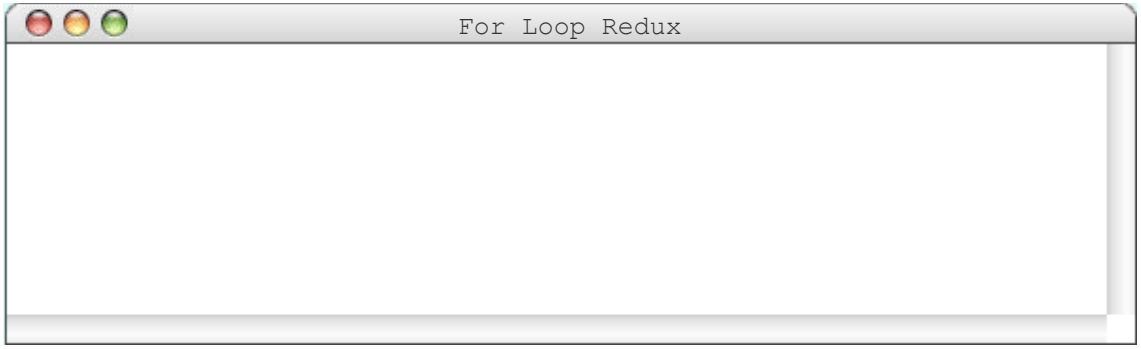
```
for(int i = 0; i < 3; i++) {
   println("Nick rocks socks!");
}
```



For Loop Redux

i | 0

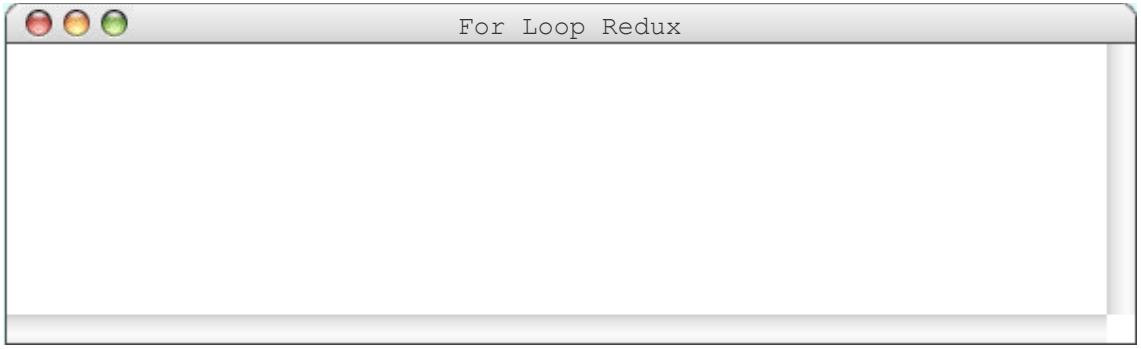
```
for(int i = 0; i < 3; i++) {
   println("Nick rocks socks!");
}
```



For Loop Redux

i | 0 |

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



For Loop Redux

i | 0

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

For Loop Redux

```
Nick rocks socks
```

For Loop Redux

i | 0 |

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

For Loop Redux

```
Nick rocks socks
```

For Loop Redux

i 1

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

For Loop Redux

```
Nick rocks socks
```

For Loop Redux

i | 1 |

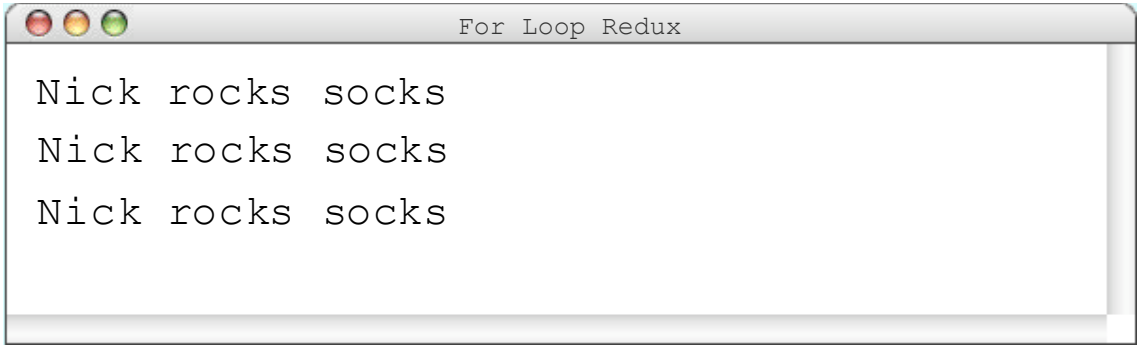
```
for(int i = 0; i < 3; i++) {
   println("Nick rocks socks!");
}
```

For Loop Redux

```
Nick rocks socks
```

For Loop Redux

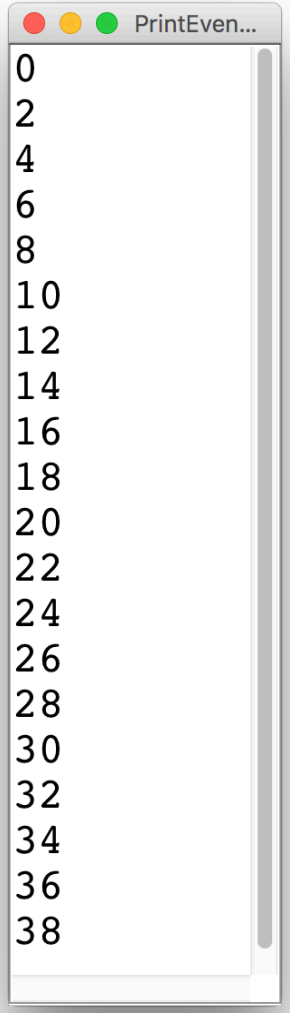
```
for(int i = 0; i < 3; i++) {
   println("Nick rocks socks!");
}
```



You can use the for loop variable

How would you println the first 100 even numbers?

Printing Even Numbers

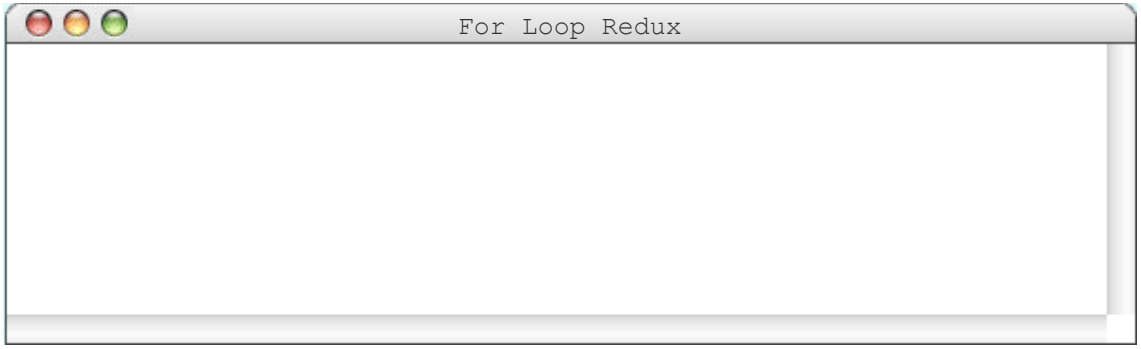


Printing Even Numbers

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

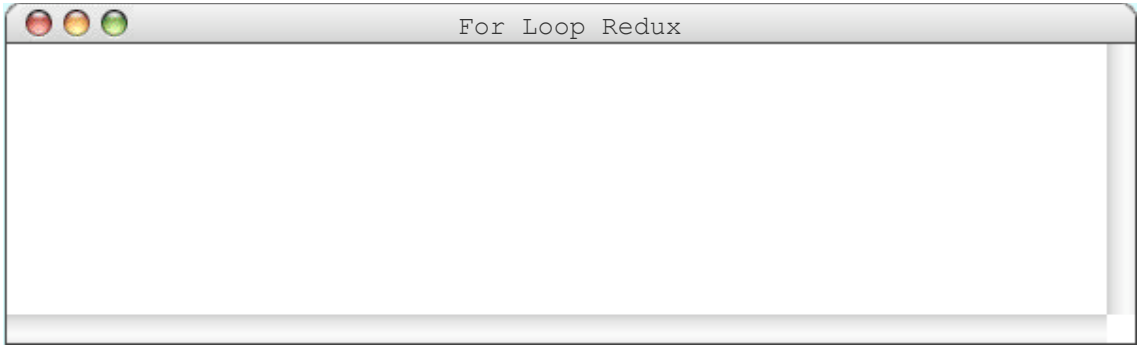
Printing Even Numbers

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



Printing Even Numbers

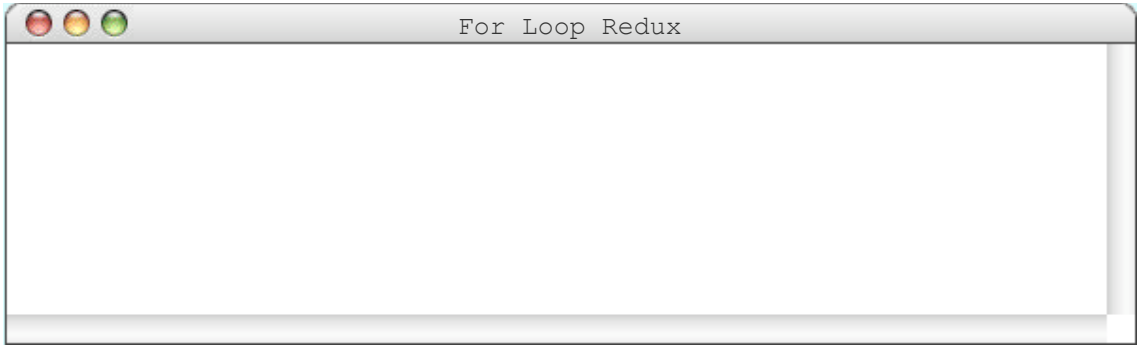
```
for(int i = 0; i < 3; i++) {
   println(i * 2);
}
```



Printing Even Numbers

i | 0

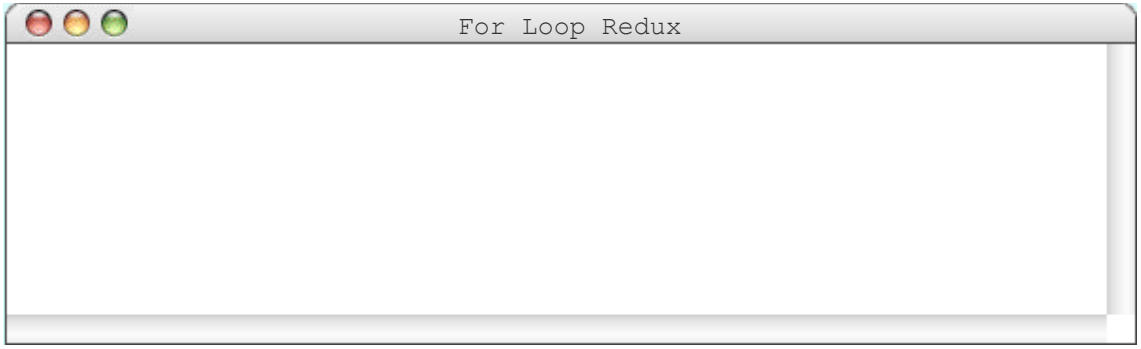
```
for(int i = 0; i < 3; i++) {
   println(i * 2);
}
```



Printing Even Numbers

i | 0 |

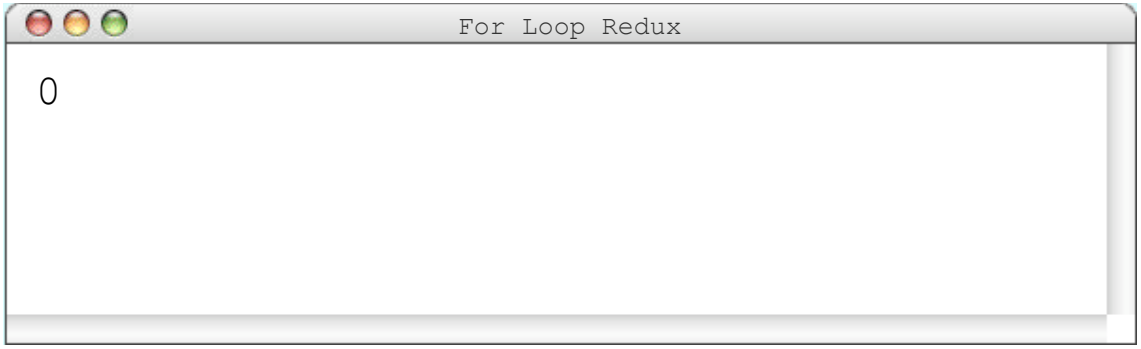
```
for(int i = 0; i < 3; i++) {
   println(i * 2);
}
```



Printing Even Numbers

i | 0

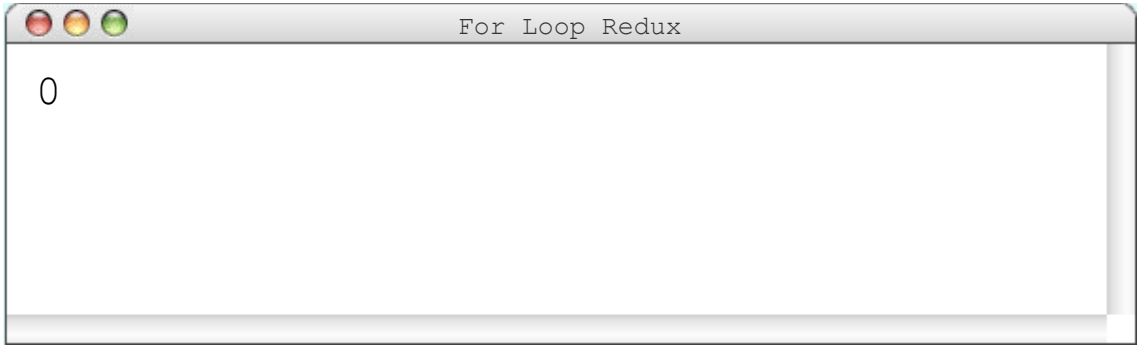
```
for(int i = 0; i < 3; i++) {
   println(i * 2);
}
```



Printing Even Numbers

i 1

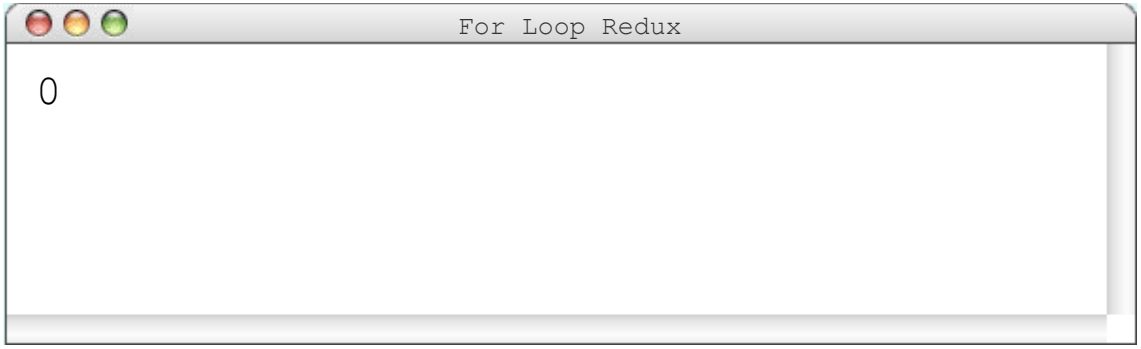
```
for(int i = 0; i < 3; i++) {
   println(i * 2);
}
```



Printing Even Numbers

i | 1 |

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



Printing Even Numbers

i | 1 |

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

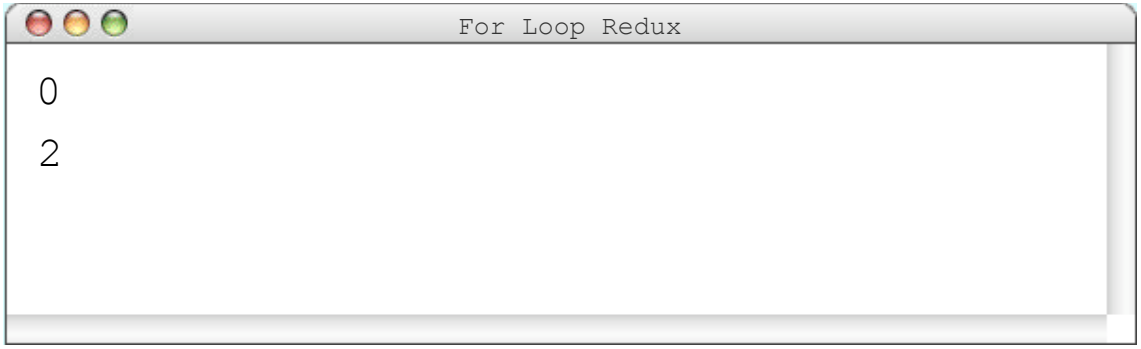
For Loop Redux

```
0
2
```

Printing Even Numbers

i | 2 |

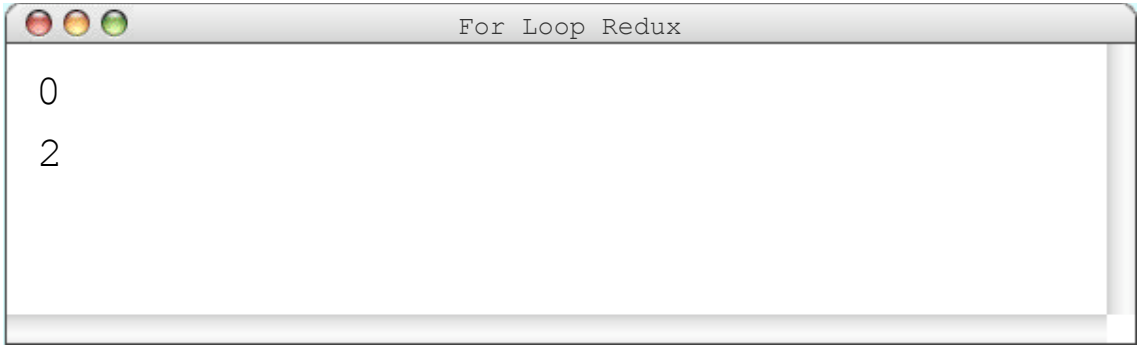
```
for(int i = 0; i < 3; i++) {
   println(i * 2);
}
```



Printing Even Numbers

i | 2 |

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



Printing Even Numbers

i | 2 |

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

For Loop Redux

```
0
2
4
```

Printing Even Numbers

i | 3

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

For Loop Redux

```
0
2
4
```

Printing Even Numbers

i | 3 |

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

For Loop Redux

```
0
2
4
```

Printing Even Numbers

i | 3 |

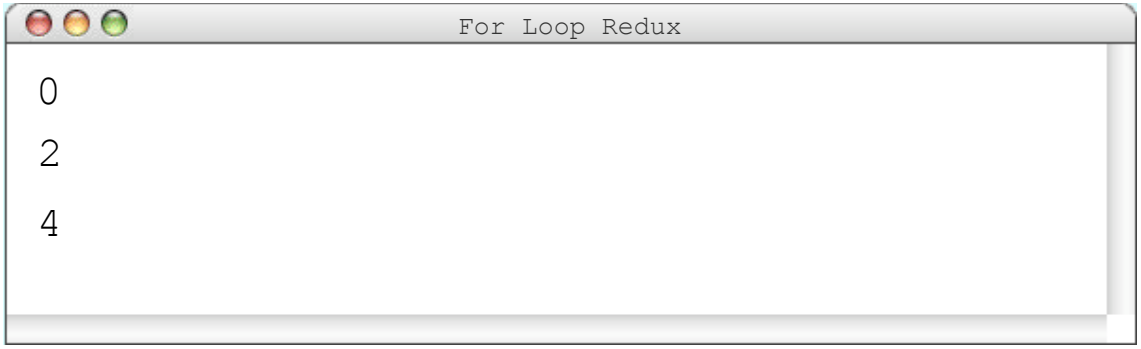
```
for(int i = 0; i < 3; i++) {
   println(i * 2);
}
```

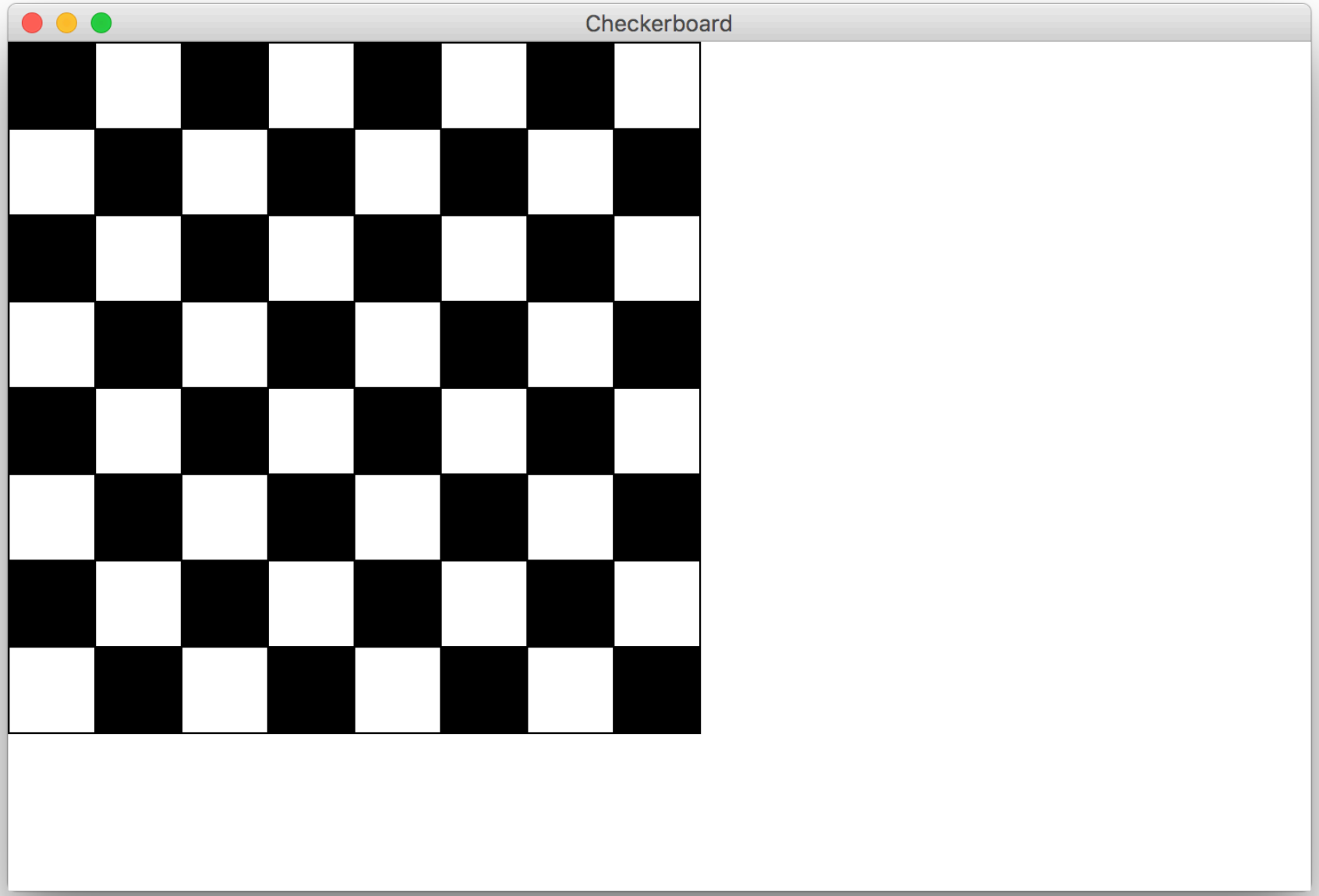
For Loop Redux

```
0
2
4
```

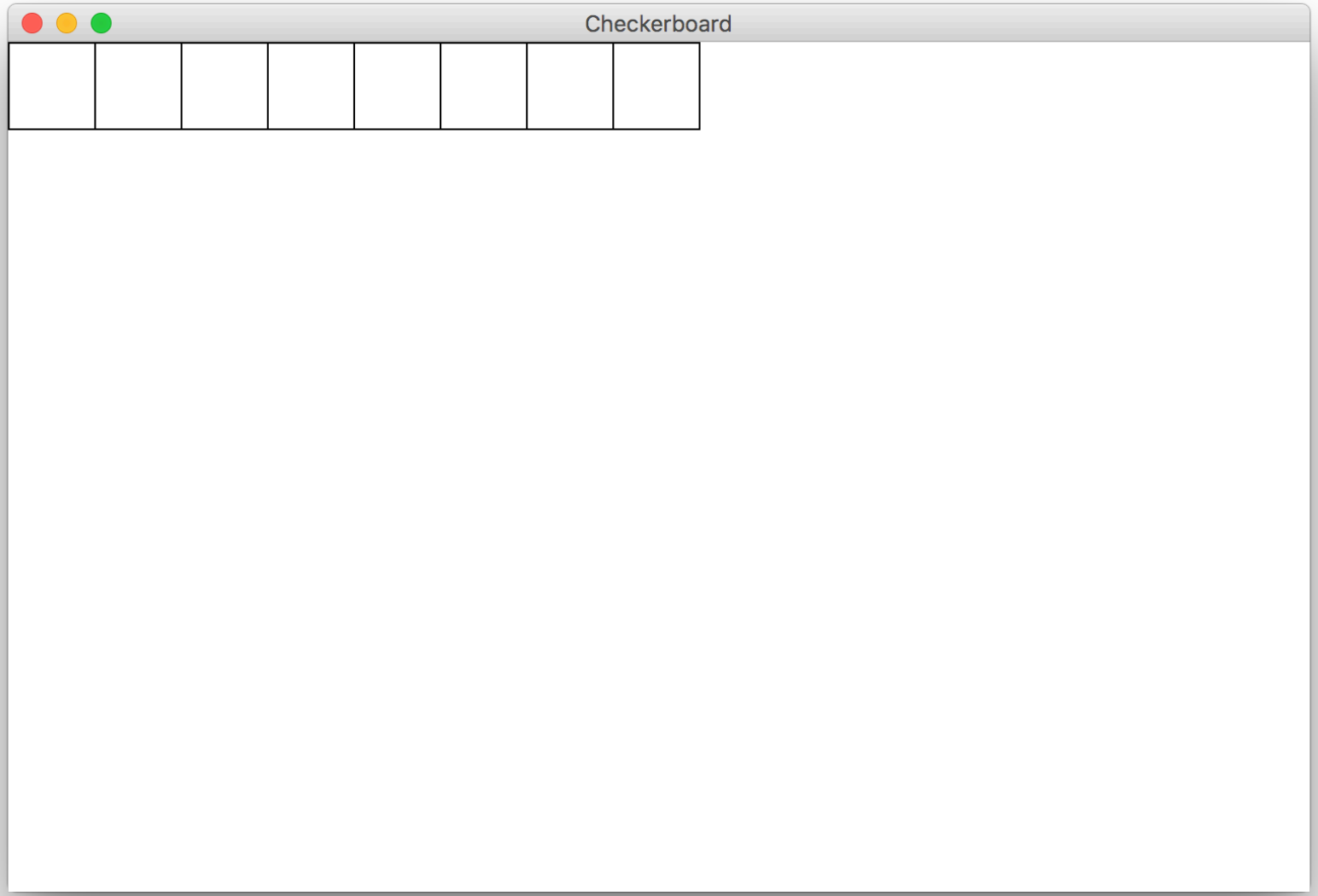
Printing Even Numbers

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

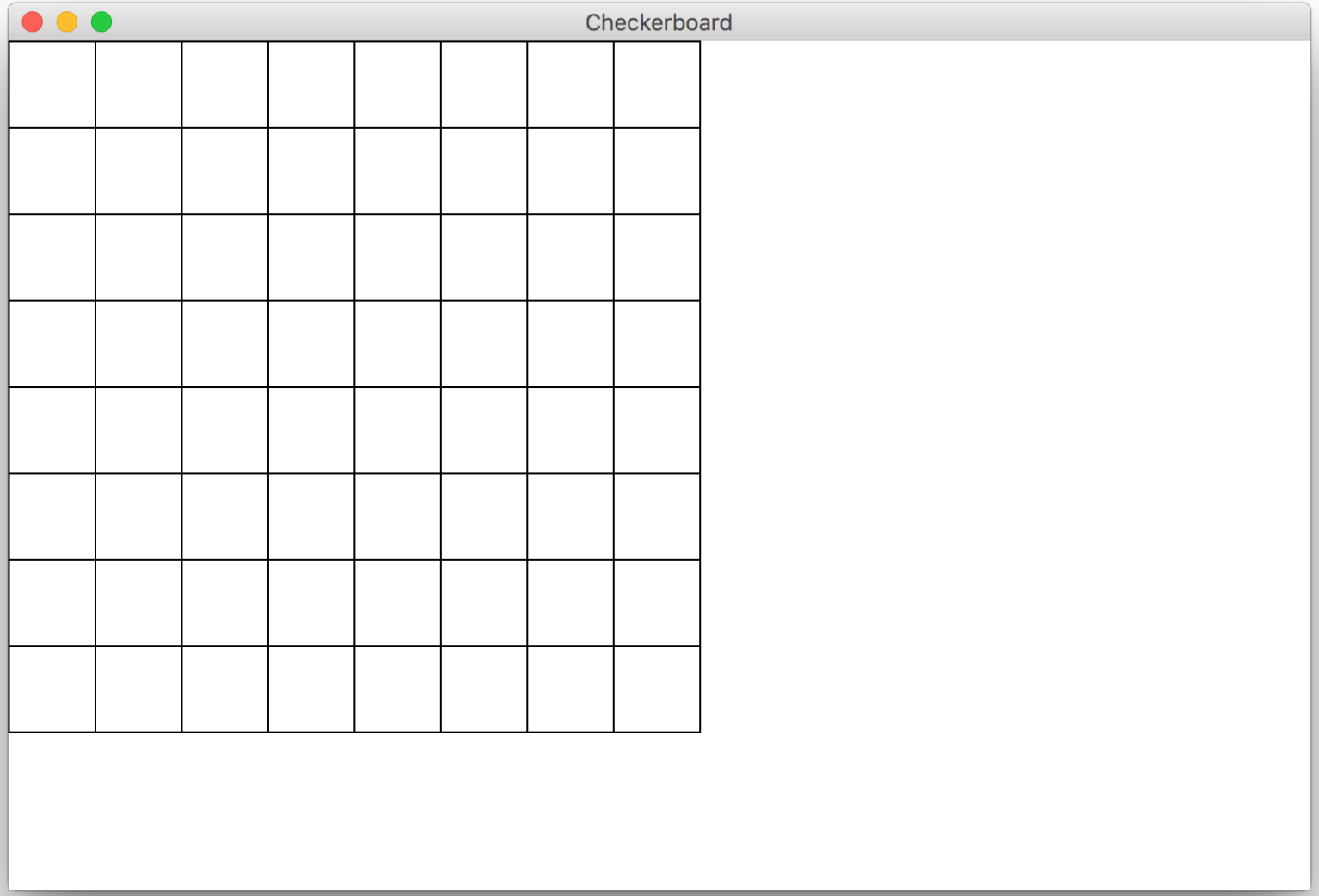



Checkerboard

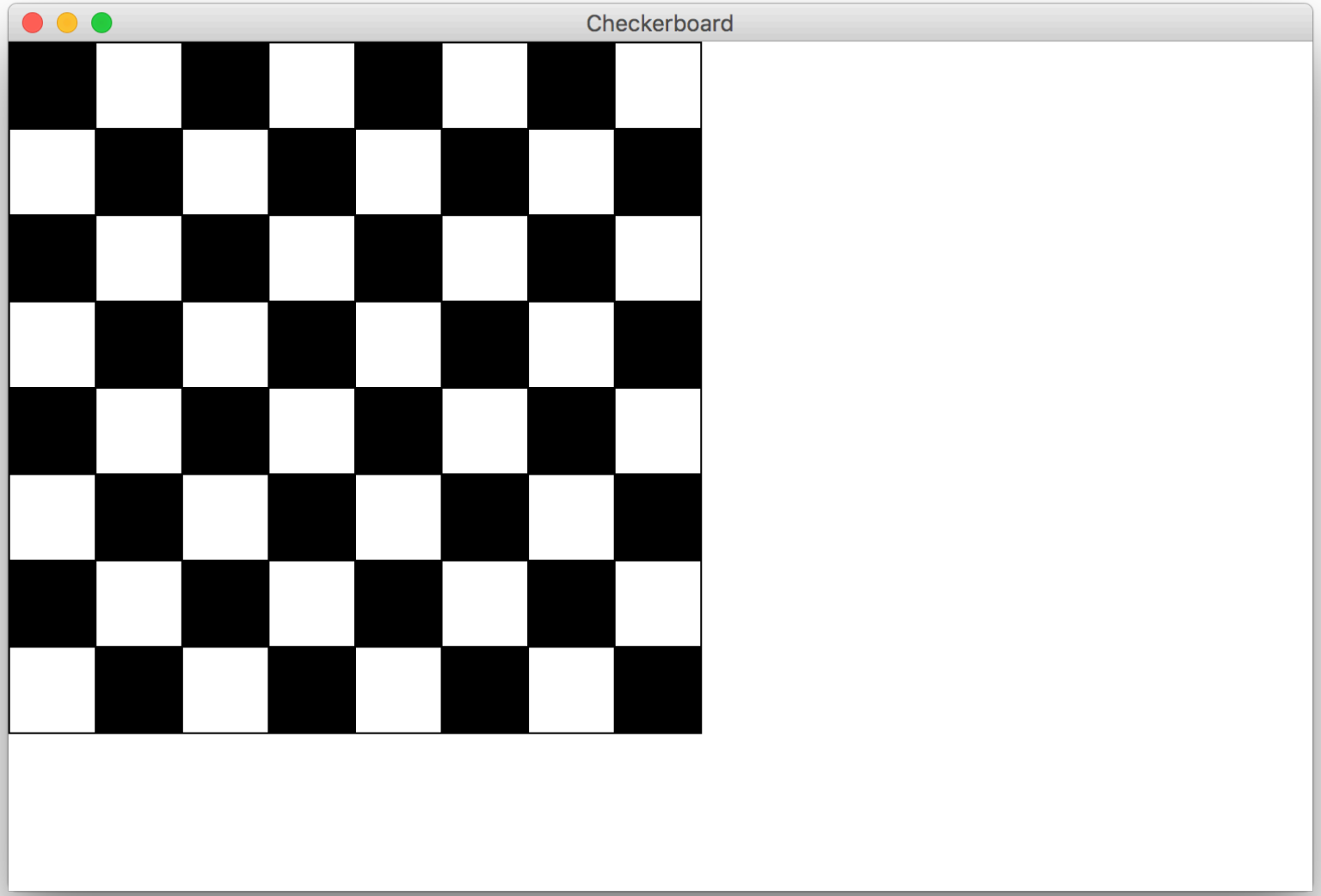
Milestone 1



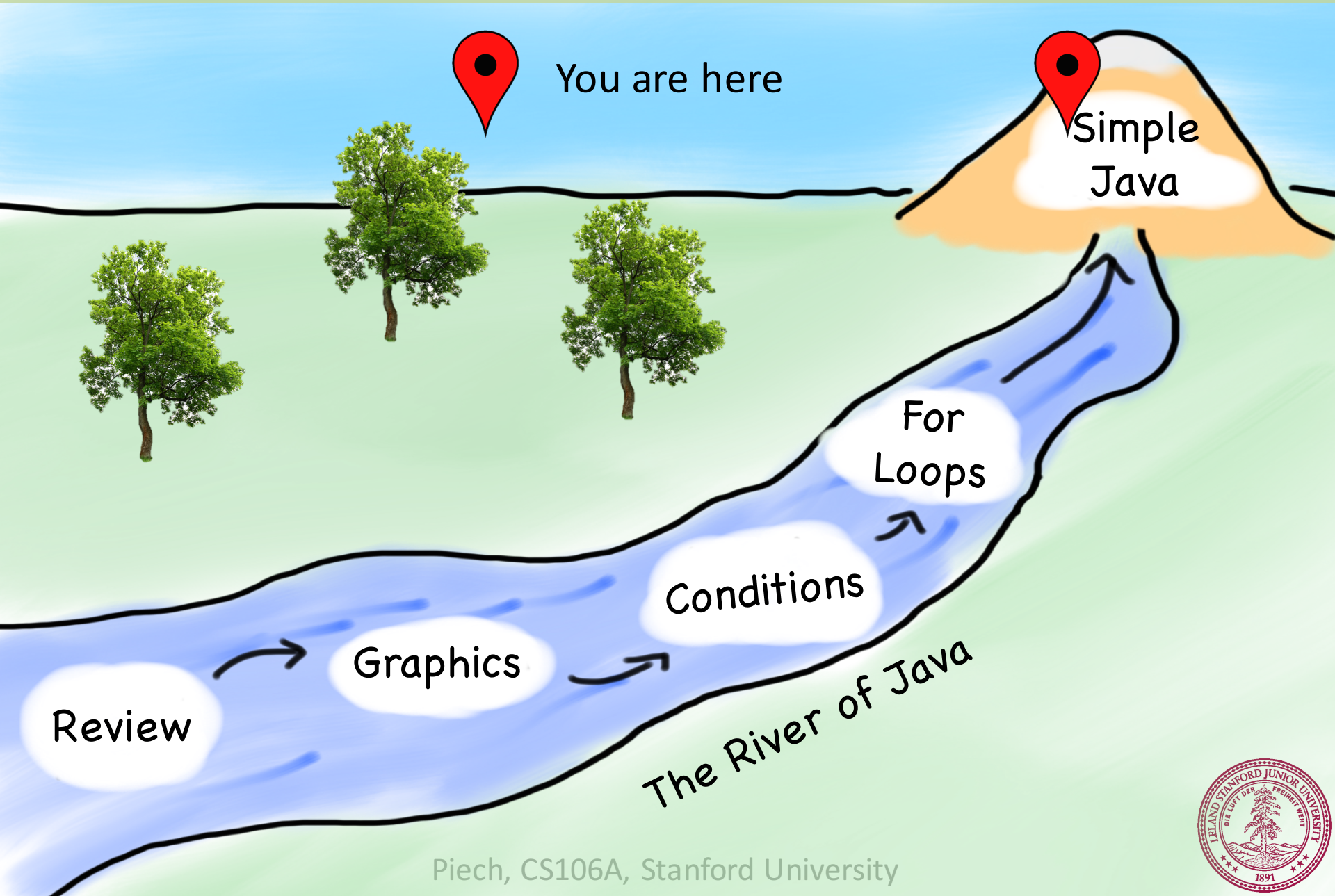
Milestone 2



Milestone 3



Today's Route



Today's Goal

1. How to use constants
2. Basics of boolean variables
3. Understand For loops
4. Know variable scope