CS 106A, Lecture 5
Booleans and Control Flow

suggested reading:

Java Ch. 3.4-4.6

Plan For Today

- Announcements
- Recap: Java, Variables and Expressions
- Aside: Shorthand Operators + Constants
- Revisiting Control Flow
  - If and While
  - For

Plan For Today

- Announcements
- Recap: Java, Variables and Expressions
- Aside: Shorthand Operators + Constants
- Revisiting Control Flow
  - If and While
  - For

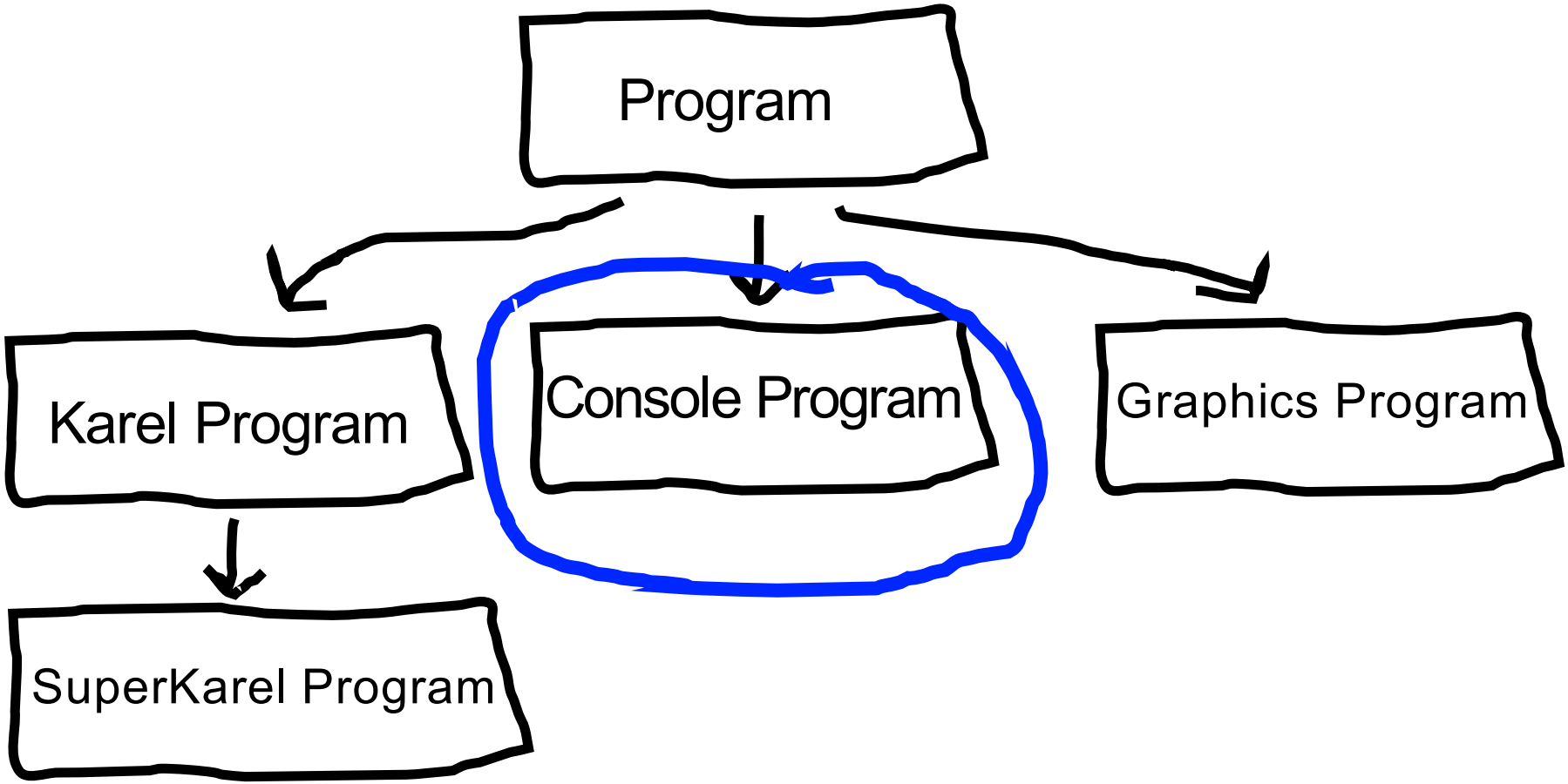
Announcements

- Everything canceled on Wednesday (7/4)
  - Lecture and LaIR are just not happening
  - Wednesday sections have been rescheduled
    - Go to your section leader's if you can; otherwise, go to a different *rescheduled* section
- Assignment 1 due *Thursday at 11AM*
- Debugger tutorial up on website

Plan For Today

- Announcements
- Recap: Java, Variables and Expressions
- Aside: Shorthand Operators + Constants
- Revisiting Control Flow
  - If and While
  - For

Java


Program
Karel Program
Console Program
Graphics Program
SuperKarel Program

Console I/O

- println allows out to output text to the user via the console
  - Output is the “O” in “I/O”
- We can also get input from the user via the console!
  - Use variables to store data collected via `readInt`, `readDouble`, etc.

Expressions

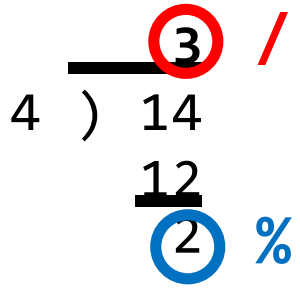
- You can combine literals or variables together into **expressions** using binary operators:

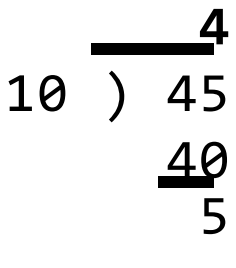
+	Addition	*	Multiplication
–	Subtraction	/	Division
		%	Remainder

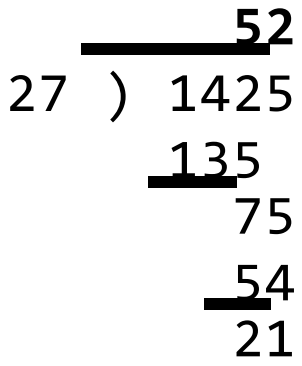
Integer division, remainder

- When we divide integers, the quotient is also an integer.

  `14 / 4` is `3`, not `3.5`. *(Java ALWAYS rounds down.)*







- More examples:
  - `32 / 5` is `6` `32 % 5` is `2`
  - `84 / 10` is `8` `84 % 10` is `4`
  - `156 / 100` is `1` `156 % 100` is `56`

  - Dividing by `0` using `/` or `%` causes an error when your program runs.

Type Interactions

`int` and `int` results in an `int`

`double` and `double` results in a `double`

`int` and `double` results in a `double`

`String` and `int` results in a `String`

etc.

* The general rule is: operations always return the most expressive type

Precedence

- **precedence**: Order in which operators are evaluated.
  - Generally operators evaluate left-to-right.

    `1 - 2 - 3` is `(1 - 2) - 3` which is `-4`

  - But `* / %` have a higher level of precedence than `+ -`

    `1 + 3 * 4` is 13

    ```
    6 + 8 / 2 * 3
    6 +   4   * 3
    6 +     12      is 18
    ```

  - Parentheses can alter order of evaluation, but spacing does not:

    `(1 + 3) * 4` is 16

    `1+3 * 4-2` is 11

Practice

- 1 / 2 **0**
- 1.0 / 2 **0.5**
- 1 + 2 / 3 **1**
- "abc" + (4 + 2) **"abc6"**
- "abc" + 4 + 2 **"abc42"**

Variable Types

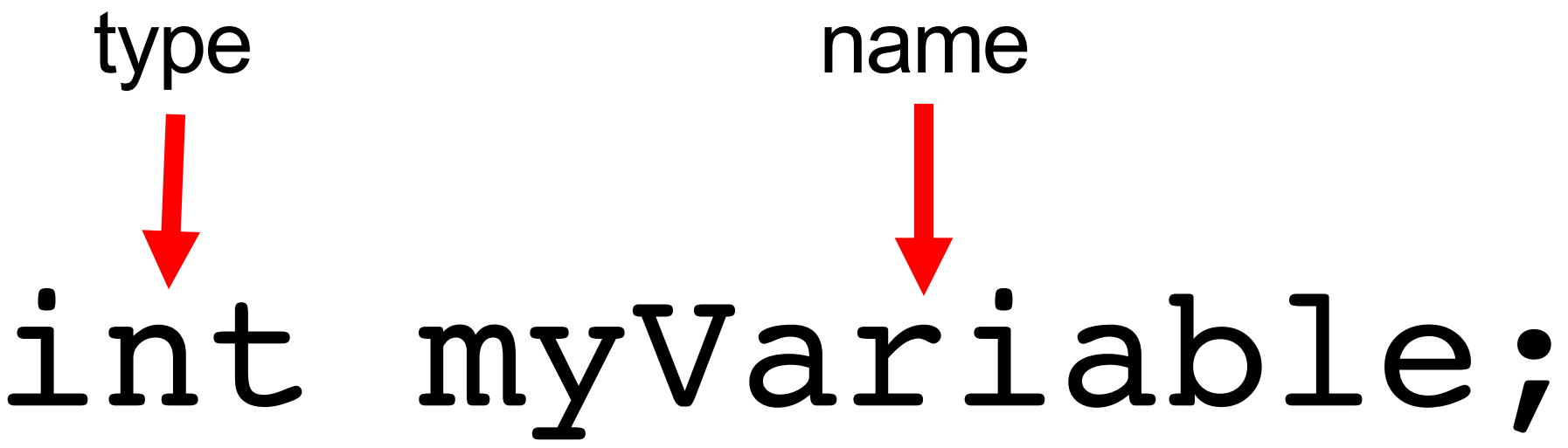
`int` – an integer number

`double` – a decimal number

`char` – a single character

`boolean` – true or false

Making a new Variable



Assignment

Existing variable name

value

```
myVariable = 2;
```

Declare / initialize

- A variable can be declared/initialized in one statement.
  - This is probably the most commonly used declaration syntax.

- Syntax:

```
type name = expression;
```

```
double tempF = 98.6;
```


```
int x = (12 / 2) + 3;
```

Plan For Today

- Announcements
- Recap: Variables and Expressions
- Aside: Shorthand Operators + Constants
- Revisiting Control Flow
  - If and While
  - For

Shorthand Operators

Shorthand	Equivalent longer version
variable += value;	*variable = variable + value;*
variable -= value;	*variable = variable - value;*
*variable *= value;*	*variable = variable * value;*
variable /= value;	*variable = variable / value;*
variable %= value;	*variable = variable % value;*
variable++;	*variable = variable + 1;*
variable--;	*variable = variable – 1;*
`x -= 3;`	`// x = x - 3;`
`number *= 2;`	`// number = number * 2;`
`x++;`	`// x = x + 1;`

Constants

- **constant**: A variable that cannot be changed after it is initialized.

  Declared at the top of your class, *outside of the run() method* but inside `public class Name`. Can be used anywhere in that class.
- Better style – can easily change their values in your code, and they are easier to read in your code.
- Syntax:

```
private static final type name = value;
```

  - name is usually in ALL_UPPER_CASE

  - Examples:

```
private static final int DAYS_IN_WEEK = 7;
private static final double INTEREST_RATE = 3.5;
```

Receipt Program - Before

```
public class Receipt extends ConsoleProgram {
  public void run() {
    double subtotal = readDouble("Meal cost? $");
    double tax = subtotal * 0.08;
    double tip = subtotal * 0.20;
    double total = subtotal + tax + tip;

    println("Tax : $" + tax);
    println("Tip: $" + tip);
    println("Total: $" + total);
  }
}
```

Receipt Program – After

```
public class Receipt extends ConsoleProgram {
  private static final double TAX_RATE = 0.08;
  private static final double TIP_RATE = 0.2;

  public void run() {
    double subtotal = readDouble("Meal cost? $");
    double tax = subtotal * TAX_RATE;
    double tip = subtotal * TIP_RATE;
    double total = subtotal + tax + tip;

    println("Tax : $" + tax);
    println("Tip: $" + tip);
    println("Total: $" + total);
  }
}
```

Plan For Today

- Announcements
- Recap: Variables and Expressions
- Aside: Shorthand Operators + Constants
- Revisiting Control Flow
  - If and While
  - For

If/Else in Karel

```
if (condition) {
    statement;
    statement;
    ...
} else {
    statement;
    statement;
    ...
}
```

Runs the first group of statements if ***condition*** is true; otherwise, runs the second group of statements.

While Loops in Karel

```
while (condition) {
    statement;
    statement;
    ...
}
```

Repeats the statements in the body until ***condition*** is no longer true.

Each time, Karel executes *all statements*, and **then** checks the condition.

Conditions in Karel

```
while (frontIsClear()) {
    body
}
```

```
if (beepersPresent()) {
    body
}
```

Conditions in Java

```
while (condition) {
    body
}
```

```
if (condition) {
    body
}
```

The condition should be a “boolean” which is either **`true`** or **`false`**

Booleans

$1 < 2$

Booleans

$1 < 2$

`true`

Relational Operators

Operator	Meaning	Example	Value
==	equals	`1 + 1 == 2`	`true`
!=	does not equal	`3.2 != 2.5`	`true`
<	less than	`10 < 5`	`false`
>	greater than	`10 > 5`	`true`
<=	less than or equal to	`126 <= 100`	`false`
>=	greater than or equal to	`5.0 >= 5.0`	`true`

* All have equal precedence

Relational Operators

Operator	Meaning	Example	Value
==	equals	`1 + 1 == 2`	`true`
!=	does not equal	`3.2 != 2.5`	`true`
<	less than	`10 < 5`	`false`
>	greater than	`10 > 5`	`true`
<=	less than or equal to	`126 <= 100`	`false`
>=	greater than or equal to	`5.0 >= 5.0`	`true`

* All have equal precedence

Relational Operators

```
if (1 < 2) {
    println("1 is less than 2!");
}
```

```
int num = readInt("Enter a number: ");
if (num == 0) {
    println("That number is 0!");
} else {
    println("That number is not 0.");
}
```

Practice: Sentinel Loops

- **sentinel**: A value that signals the end of user input.
  - **sentinel loop**: Repeats until a sentinel value is seen.

- Example: Write a program that prompts the user for numbers until the user types -1, then output the sum of the numbers.
  - In this case, -1 is the sentinel value.

```
Type a number: 10
Type a number: 20
Type a number: 30
Type a number: -1
Sum is 60
```

Practice: Sentinel Loops

```
// fencepost problem!
// ask for number - post
// add number to sum - fence



int sum = 0;
int num = readInt("Enter a number: ");
while (num != -1) {
      sum += num;
      num = readInt("Enter a number: ");
}
println("Sum is " + sum);
```

Practice: Sentinel Loops

```
// fencepost problem!
// ask for number - post
// add number to sum - fence
private static final int SENTINEL = -1;
```

(outside of `run()`)

```
int sum = 0;
int num = readInt("Enter a number: ");
while (num != SENTINEL) {
      sum += num;
      num = readInt("Enter a number: ");
}
println("Sum is " + sum);
```

Practice: Sentinel Loops

```
// Solution #2: "break" out of the loop
// ONLY appropriate to use in fencepost cases

int sum = 0;
while (true) {
      int num = readInt("Enter a number: ");
      if (num == -1) {
            break;     // immediately exits loop
      }
      sum += num;
}
println("Sum is " + sum);
```

Colin prefers this solution, but the debate of how to solve the "loop-and-a-half" problem has been raging for >50 years!

Logical Operators

In order of precedence:

Operator	Description	Example	Result
!	not	!(2 == 3)	true
&&	and	(2 == 3) && (-1 < 5)	false
||	or	(2 == 3) || (-1 < 5)	true

Cannot "chain" tests as in algebra; use && or || instead

```
// assume x is 15
2 <= x <= 10
true   <= 10
Error!
```

```
// correct version
2 <= x && x <= 10
true   && false
false
```

Precedence Madness

Precedence: arithmetic > relational > logical

```
5 * 7 >= 3 + 5 * (7 - 1) && 7 <= 11
5 * 7 >= 3 + 5 * 6 && 7 <= 11
35    >= 3 + 30 && 7 <= 11
35    >= 33 && 7 <= 11
true && true
true
```

Boolean Variables

```
// Store expressions that evaluate to true/false
boolean x = 1 < 2;          // true
boolean y = 5.0 == 4.0;     // false
```

Boolean Variables

```
// Store expressions that evaluate to true/false
boolean x = 1 < 2;          // true
boolean y = 5.0 == 4.0;     // false

// Directly set to true/false
boolean isFamilyVisiting = true;
boolean isRaining = false;
```

Boolean Variables

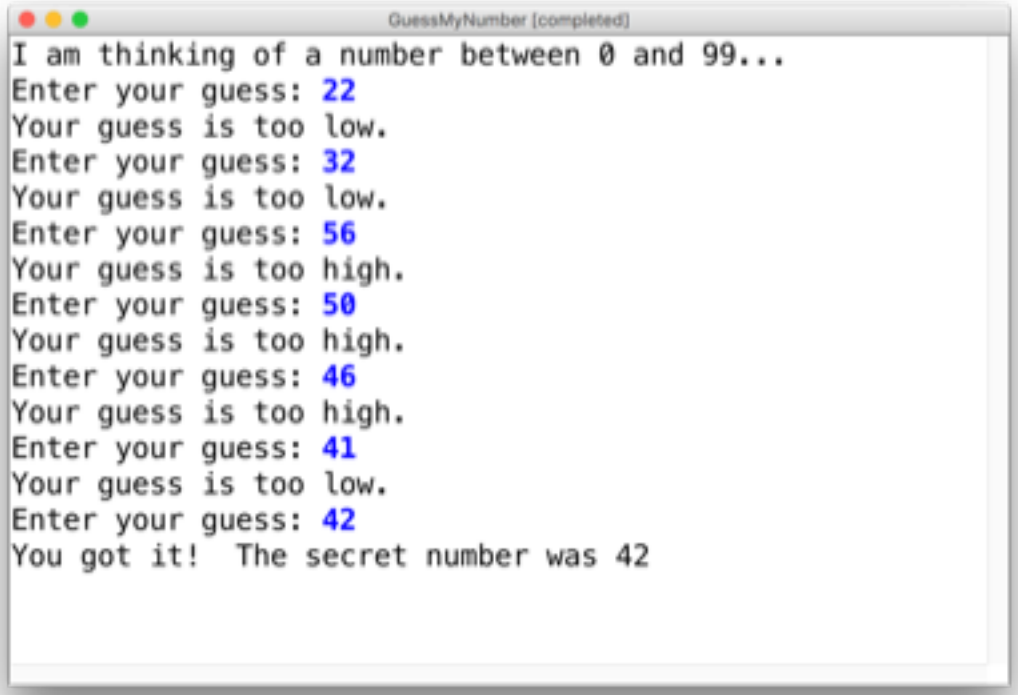
```
// Store expressions that evaluate to true/false
boolean x = 1 < 2;          // true
boolean y = 5.0 == 4.0;     // false

// Directly set to true/false
boolean isFamilyVisiting = true;
boolean isRaining = false;

// Ask the user a true/false (yes/no) question
boolean playAgain = readBoolean("Play again?", "y", "n");
if (playAgain) {
...
```

Practice: GuessMyNumber

- Let's write a program called *GuessMyNumber* that prompts the user for a number until they guess our secret number.
- If a guess is incorrect, the program should provide a hint; specifically, whether the guess is too high or too low.



Summary: Conditions

```
while (condition) {
    body
}
```

```
if (condition) {
    body
}
```

The condition should be a **boolean** which is either `true` or `false`

Plan For Today

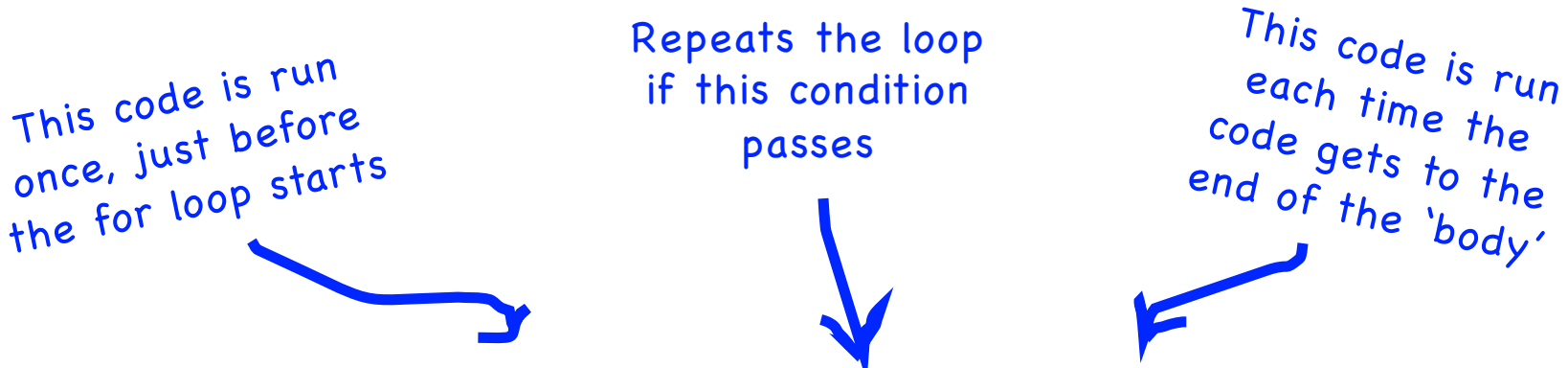
- Announcements
- Recap: Variables and Expressions
- Aside: Shorthand Operators + Constants
- Revisiting Control Flow
  - If and While
  - For

For Loops in Karel

```
for (int i = 0; i < max; i++) {
    statement;
    statement;
    ...
}
```

Repeats the statements in the body ***max*** times.

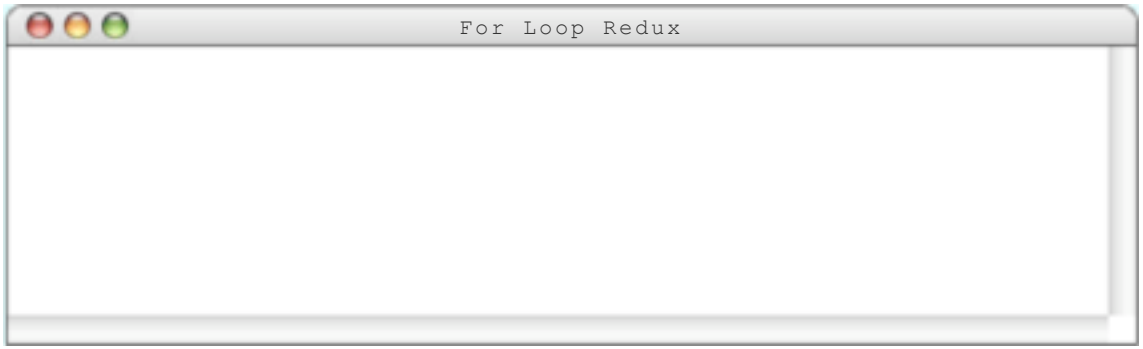
For Loops in Java



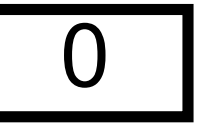

```
for (int i = 0; i < 3; i++) {
    println("I love CS 106A!");
}
```

For Loops in Java

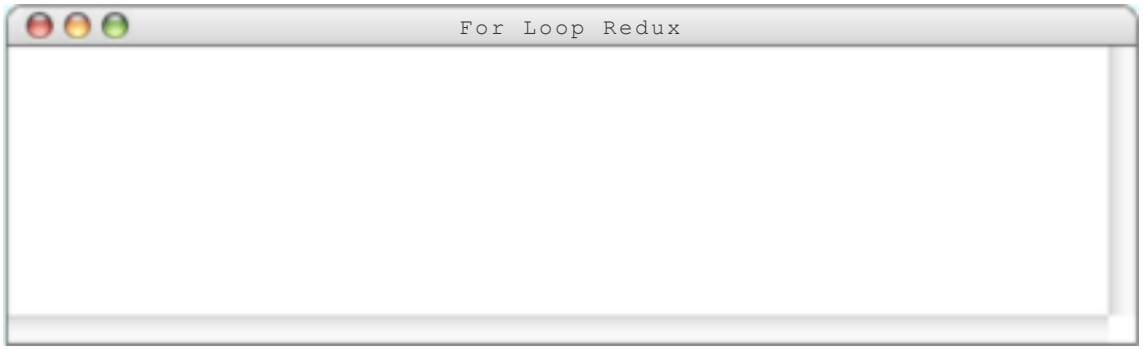
```
for (int i = 0; i < 3; i++) {
    println("I love CS 106A!");
}
```



For Loops in Java

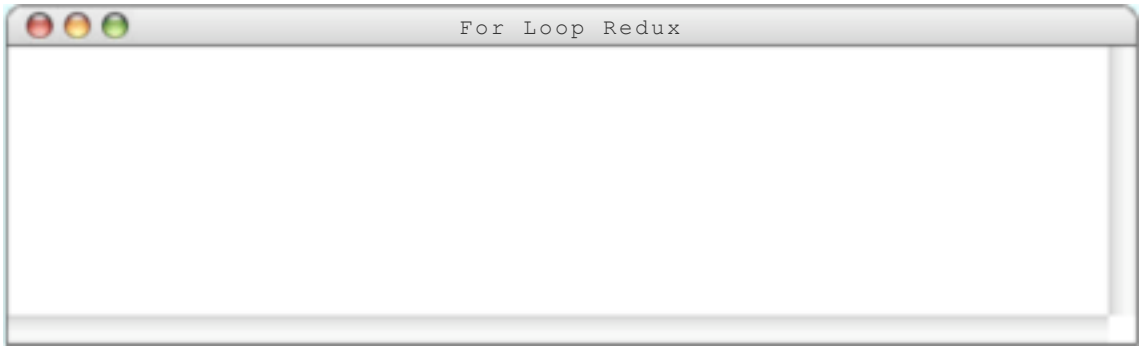



```
for (int i = 0; i < 3; i++) {
    println("I love CS 106A!");
}
```

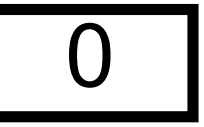


For Loops in Java

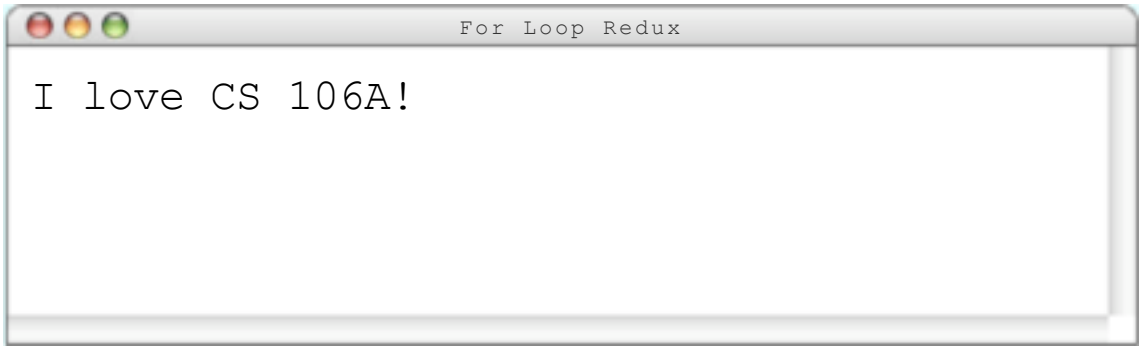

```
for (int i = 0; i < 3; i++) {
    println("I love CS 106A!");
}
```



For Loops in Java

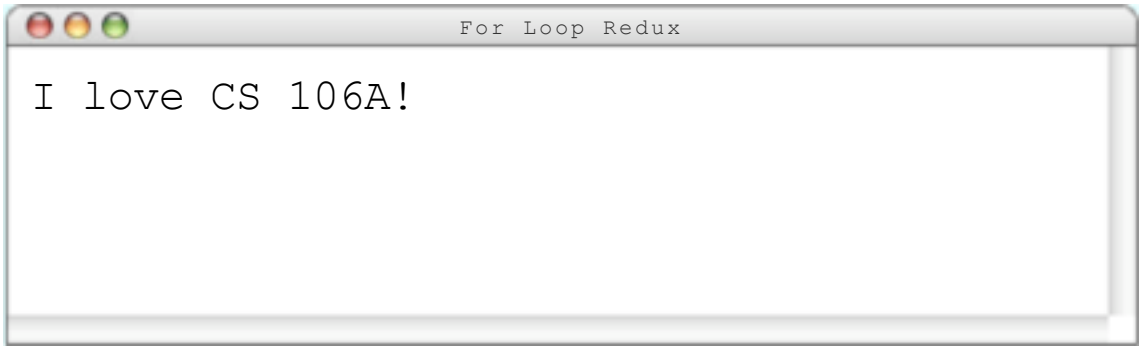



```
for (int i = 0; i < 3; i++) {
    println("I love CS 106A!");
}
```



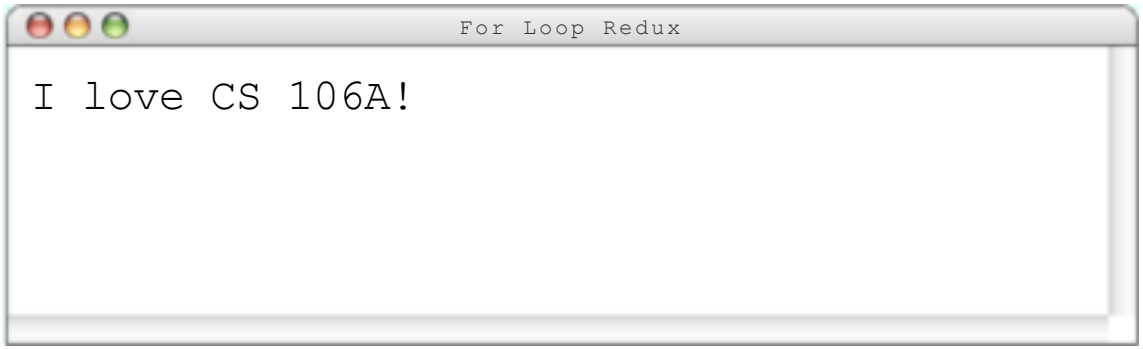
For Loops in Java


```
for (int i = 0; i < 3; i++) {
    println("I love CS 106A!");
}
```

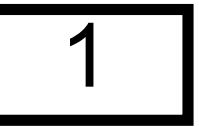


For Loops in Java

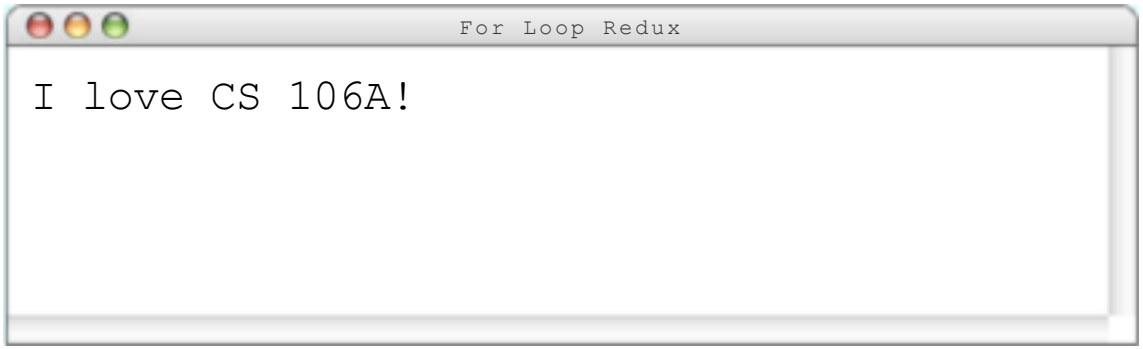

```
for (int i = 0; i < 3; i++) {
    println("I love CS 106A!");
}
```



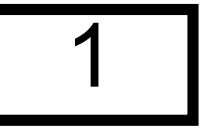
For Loops in Java



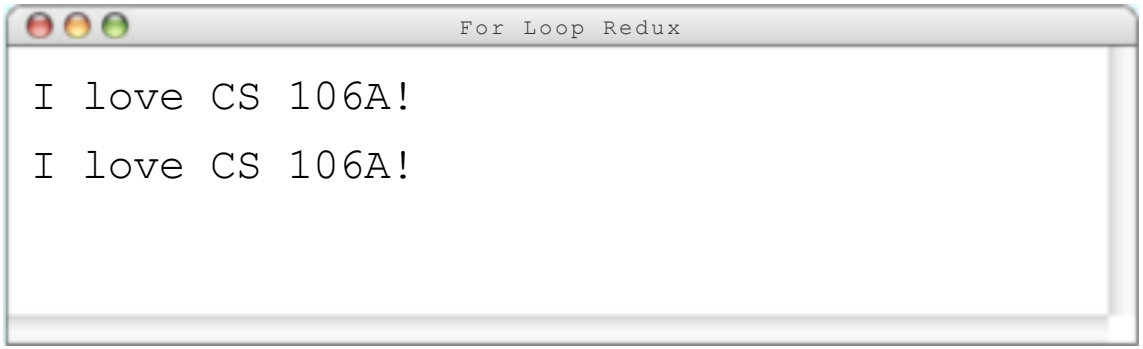

```
for (int i = 0; i < 3; i++) {
    println("I love CS 106A!");
}
```



For Loops in Java

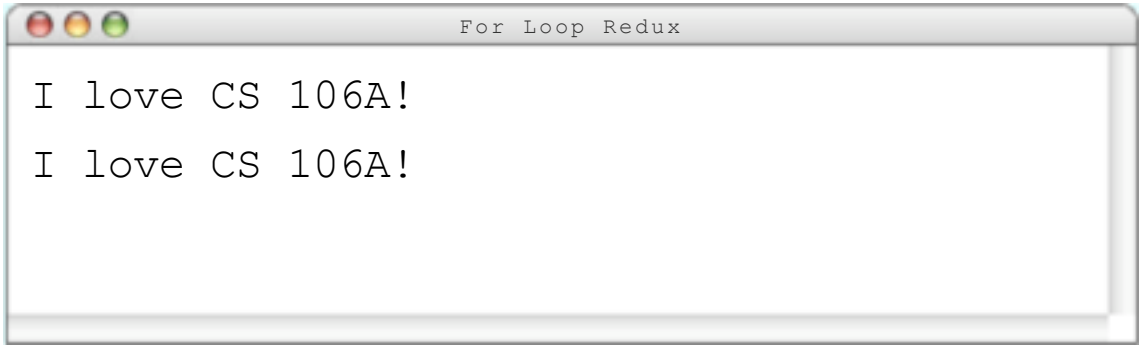



```
for (int i = 0; i < 3; i++) {
    println("I love CS 106A!");
}
```

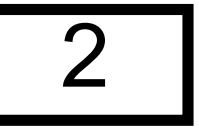


For Loops in Java

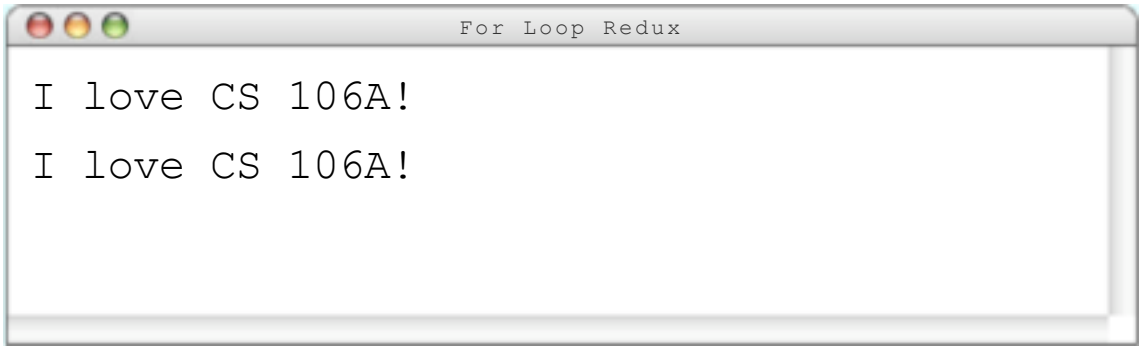

```
for (int i = 0; i < 3; i++) {
    println("I love CS 106A!");
}
```



For Loops in Java



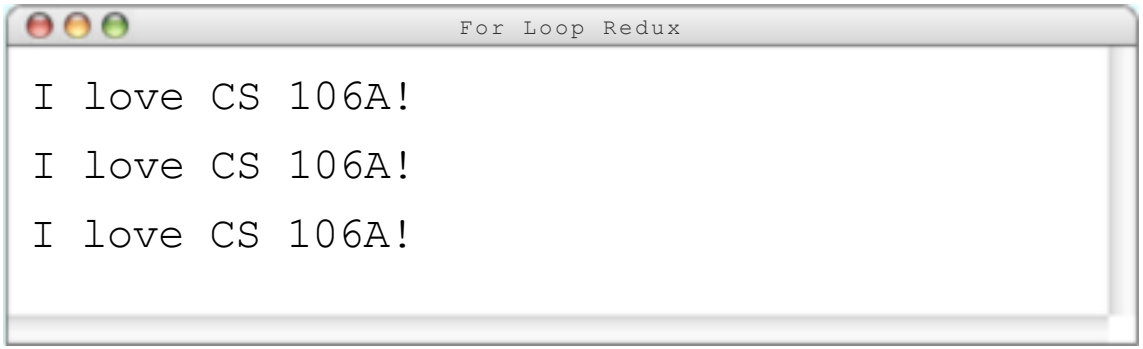

```
for (int i = 0; i < 3; i++) {
    println("I love CS 106A!");
}
```



For Loops in Java

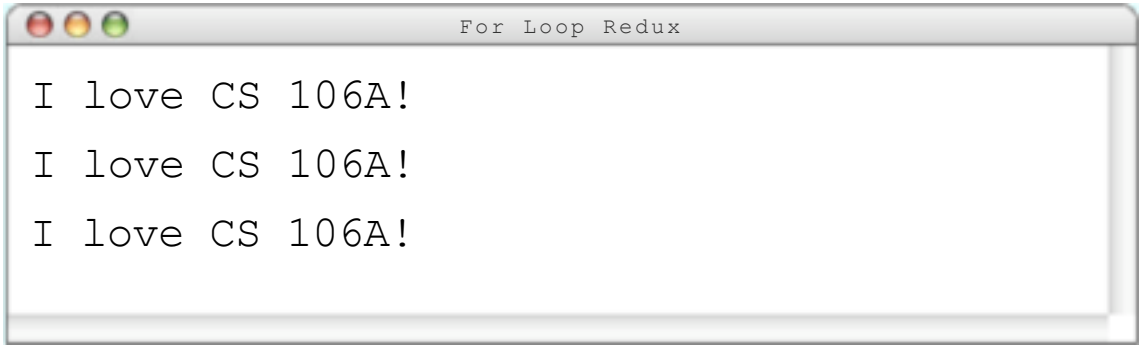
i 2

```
for (int i = 0; i < 3; i++) {
    println("I love CS 106A!");
}
```

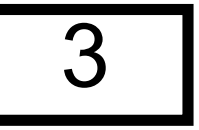


For Loops in Java

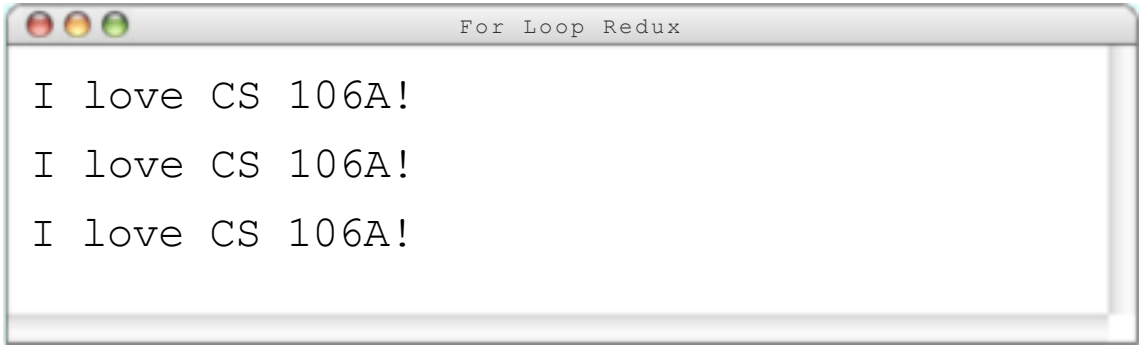

```
for (int i = 0; i < 3; i++) {
    println("I love CS 106A!");
}
```



For Loops in Java

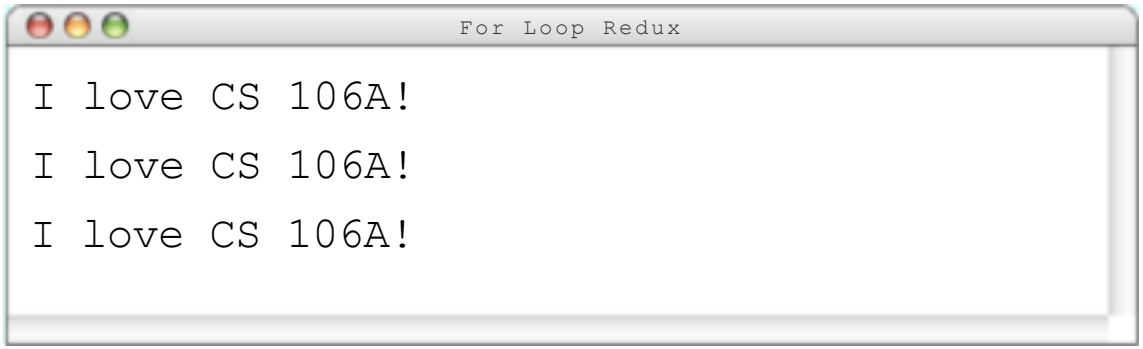



```
for (int i = 0; i < 3; i++) {
    println("I love CS 106A!");
}
```



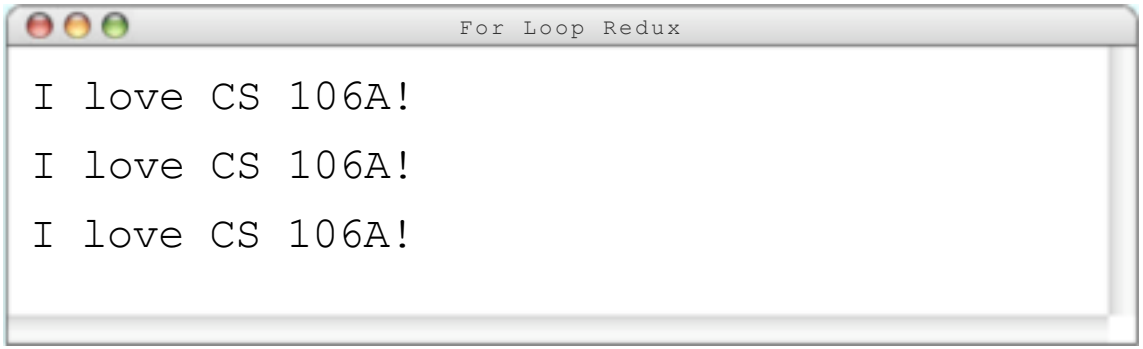
For Loops in Java

```
for (int i = 0; i < 3; i++) {
    println("I love CS 106A!");
}
```



For Loops in Java

```
for (int i = 0; i < 3; i++) {
    println("I love CS 106A!");
}
```



Using the For Loop Variable

```
// prints the first 100 even numbers
for (int i = 0; i < 100; i++) {
    println(i * 2);
}
```

Using the For Loop Variable

```
// Adds up the first 100 numbers
int sum = 0;
for (int i = 0; i < 100; i++) {
    sum += i;
}
println("The sum is " + sum);
```

Using the For Loop Variable

```
// Launch countdown
for (int i = 10; i >= 1; i--) {
    println(i);
}
println("Blast off!");
```

Output:

```
10
9
8
...
1
Blast off!
```

Recap

- Announcements
- Recap: Variables and Expressions
- Aside: Shorthand Operators + Constants
- Revisiting Control Flow
  - If and While
  - For

Next time: More control flow, methods in Java

[Extra] If/*Else If*/Else

```
if (condition1) {
    ...
} else if (condition2) {        // NEW
    ...
} else {
    ...
}
```

Runs the first group of statements if ***condition1*** is true; otherwise, runs the second group of statements if ***condition2*** is true; otherwise, runs the third group of statements.

You can have multiple else if clauses together.

[Extra] If/*Else If*/Else

```
int num = readInt("Enter a number: ");
if (num > 0) {
      println("Your number is positive");
} else if (num < 0) {
      println("Your number is negative");
} else {
      println("Your number is 0");
}
```