Extra Discussions (Lectures)

- Going over past topics — no new discussion
- Completely optional
- Thursday 6:00 — 7:45 pm
  - Physics203
  - (not Friday)
- Moving forward:
  - Selective Mondays 6:00 — 8:00 pm
  - LWSN 3102

Repetition Statements

CS 180
Prof. Sunil Prabhakar
Department of Computer Science
Purdue University

Problem

- *Write a game program that requires the user to guess a random integer.*
- *After each input from the user*
  - *Let the user know if the guess was correct*
  - *Otherwise, inform the user that the guess was either too high or too low.*
- *The game ends only when the user correctly guesses the value.*

Repetition

- To solve this problem, we need the ability to repeat a set of operations (get input, compare with secret and respond) an unknown number of times
- The number is determined by how many guesses the user takes to get it right.
- This week we will learn how to repeatedly execute portions of code using **`while`**, **`do-while`**, and **`for`** loops.

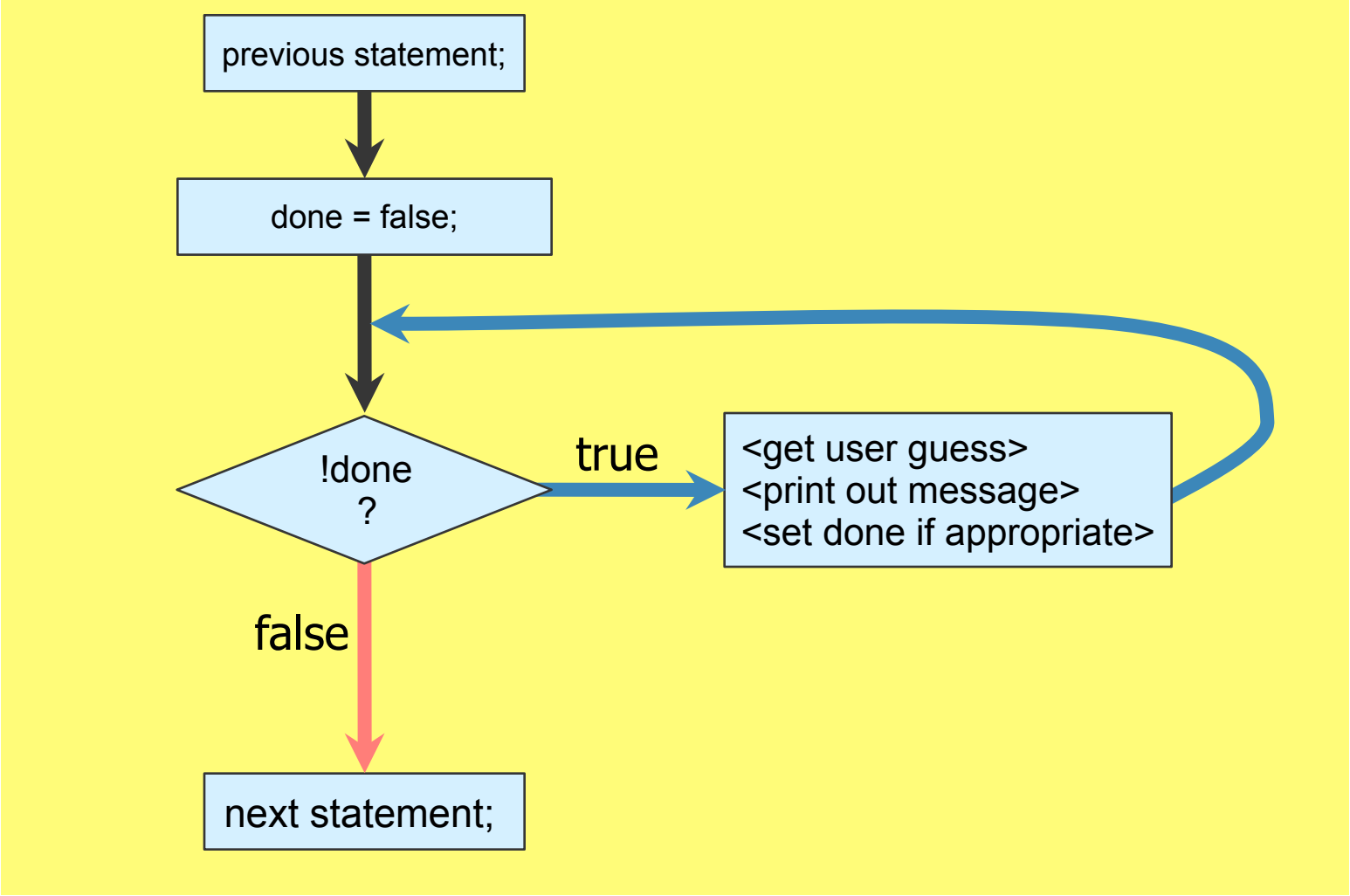
Guess

```
public class Guess {
   public static void main(String[] args){
        int secret, guess;
        boolean done;
        Random random = new Random();
        secret = random.nextInt();

        done = false;
        while(!done){
            guess = Integer.parseInt(JOptionPane.showInputDialog(
                    null, "Enter your guess."));
            if(guess == secret){
                done = true;
                System.out.println("You guessed correctly!");
            } else if (guess < secret)
                System.out.println("Your guess was too low");
            else
                System.out.println("Your guess was too high");
        }
    }
}
```

Sentinel

while loop

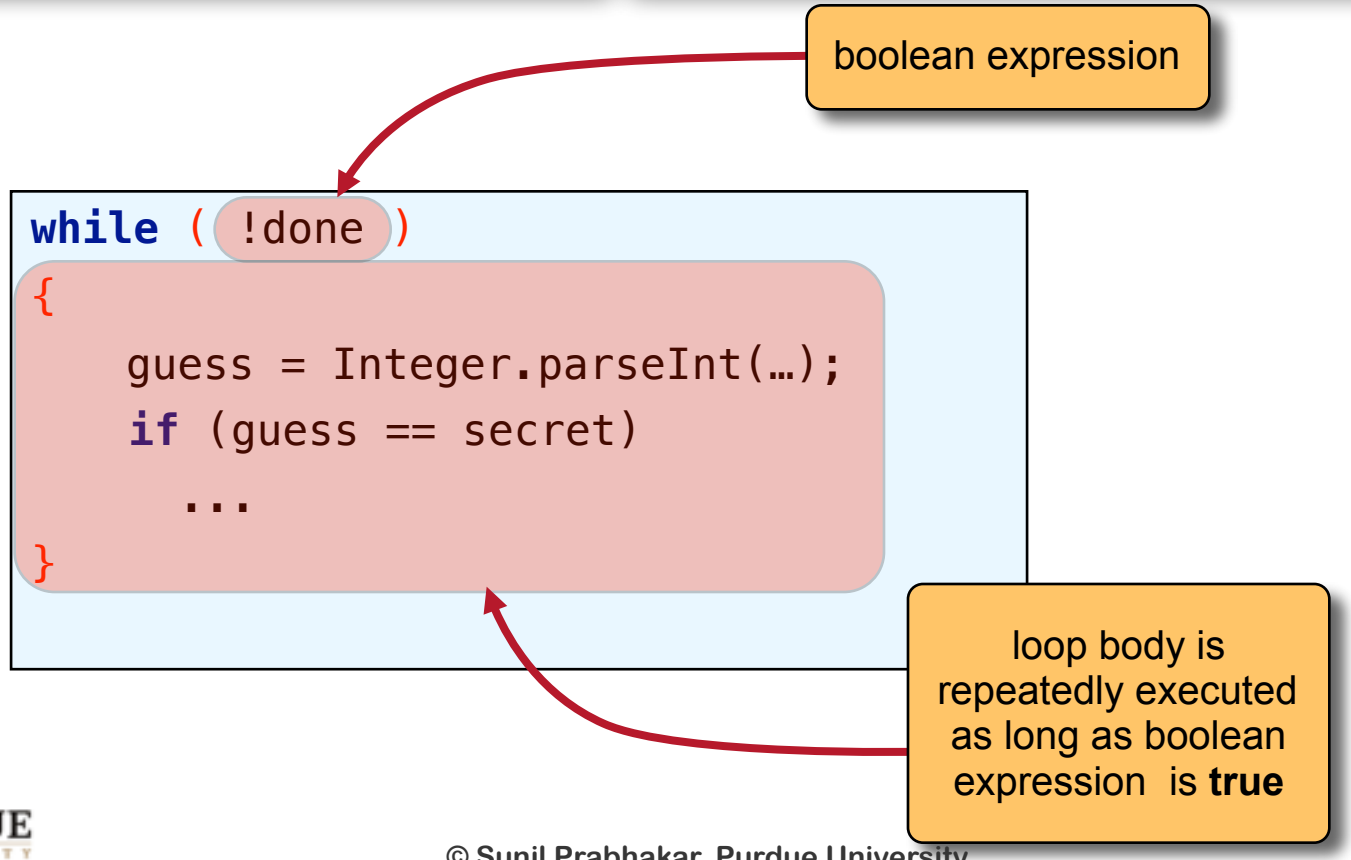
Control Flow of **while**



Syntax for the **while** Statement

```
while ( <boolean expression> )

        <statement>
```

```
while ( <boolean expression> ) {
        <statements>
}
```



Example: input check

```
char   grade;

grade = JOptionPane.showInputDialog(null, "Enter grade").charAt(0);

while (grade < 'A' || grade > 'E')
  grade = JOptionPane.showInputDialog(null,"Enter grade").charAt(0);
```

- Only accepts grades 'A' through 'E'
- Note: need for initial input before loop
  - better option **do-while** loop

The **do-while** Statement

```
char    grade;
do {
   grade = JOptionPane.showInputDialog(null,"Enter grade").charAt(0);
} while ( grade < 'A' || grade > 'E' )
```

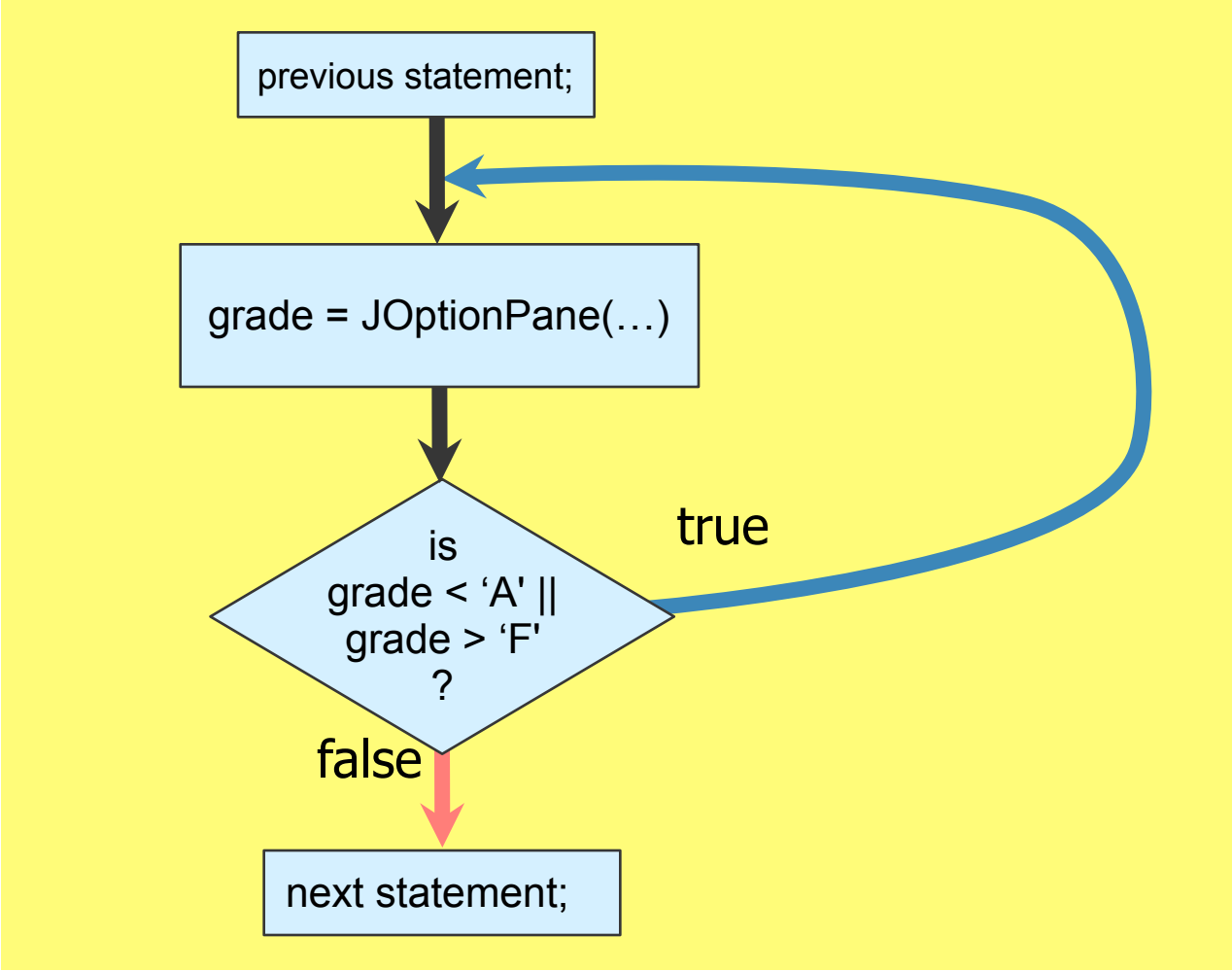
- Loop body executed before test (at least once).
- No need for initial input before loop

Common Errors

- Infinite loop
  - if the loop condition never becomes false the loop body will be executed endlessly
  - unless this is desired, ensure that the loop condition will change to false at some point

```
while(!done){
    guess = . . . ;
    if(guess == secret){
        done = true;
        System.out.println("You guessed correctly!");
    }else if . . .

}
```

Control Flow of **do while**



Problem

- *Write a program that prints out a multiplication table for a given number input by the user.*
  - *We will limit our tables to multiples up to 12.*
  - *The user input should be between 2 and 12.*

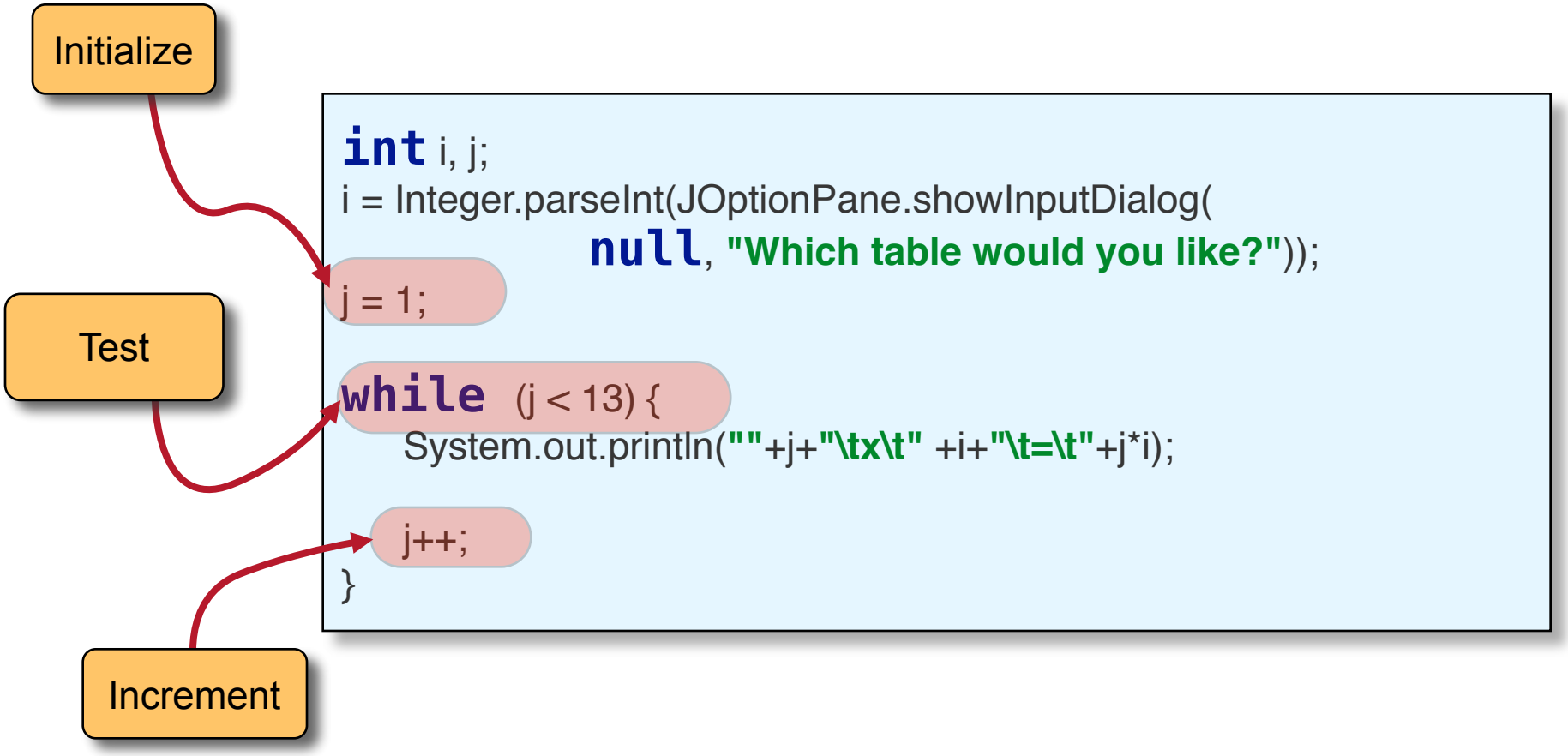
PrintOneTable

```
public class PrintOneTable {
   public static void main(String[] args){

      int i;

      i = Integer.parseInt(JOptionPane.showInputDialog(
                  null, "Which table would you like?"));
      System.out.println("1\tx\t" + i + "\t=\t" + 1*i);
      System.out.println("2\tx\t" + i + "\t=\t" + 2*i);
      System.out.println("3\tx\t" + i + "\t=\t" + 3*i);
      . . .
      System.out.println("12\tx\t" + i + "\t=\t" + 12*i);

   }
}
```

Issues

- This is not very convenient.
- What if we wanted to print the table up to multiples of 1000?
  - we would have to add 1000 print statements to our code!
- What if we wanted to change the range of multiples?
- Could use a **while** loop instead.

PrintOneTable

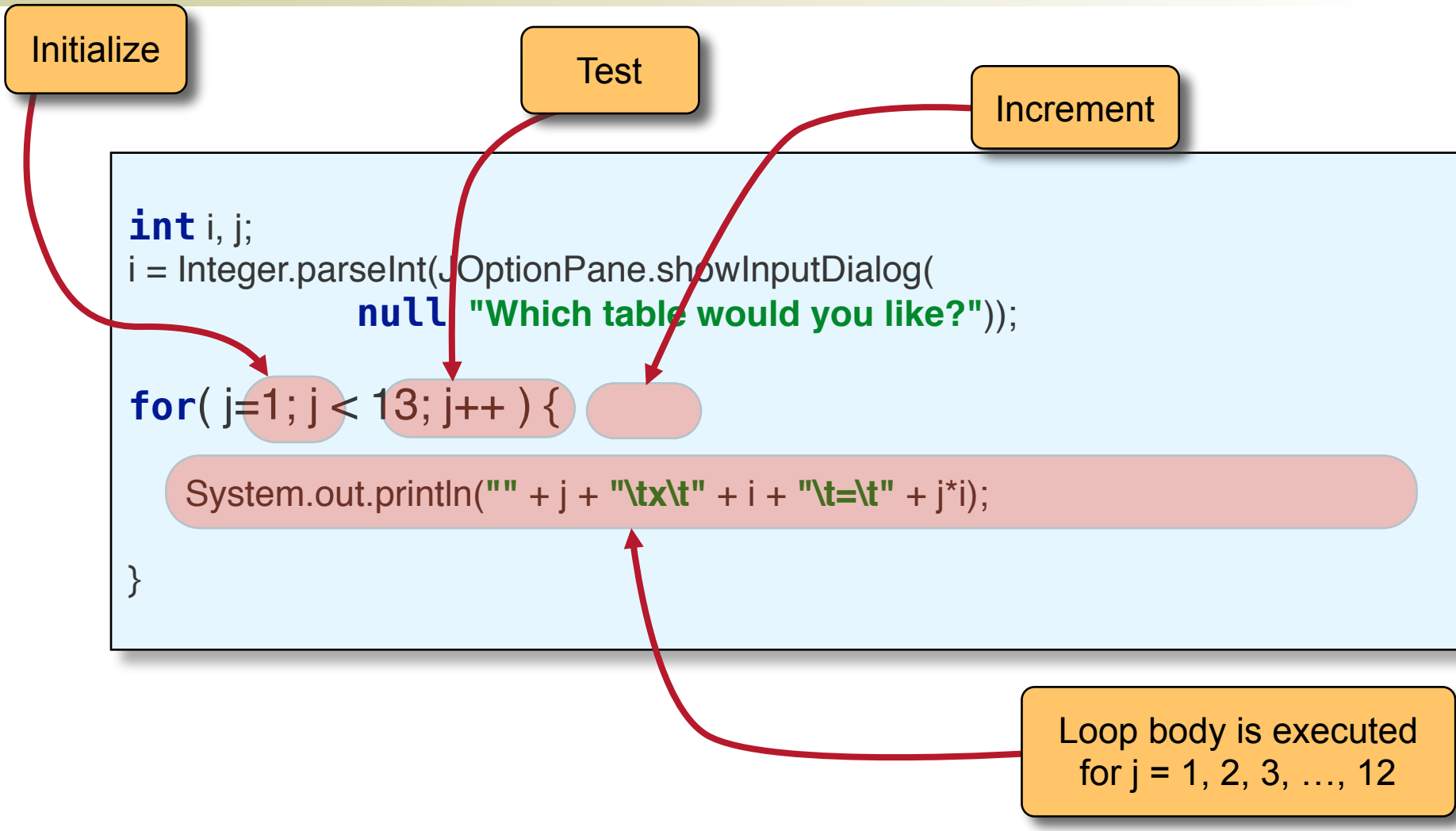


Initialize-Test-Increment

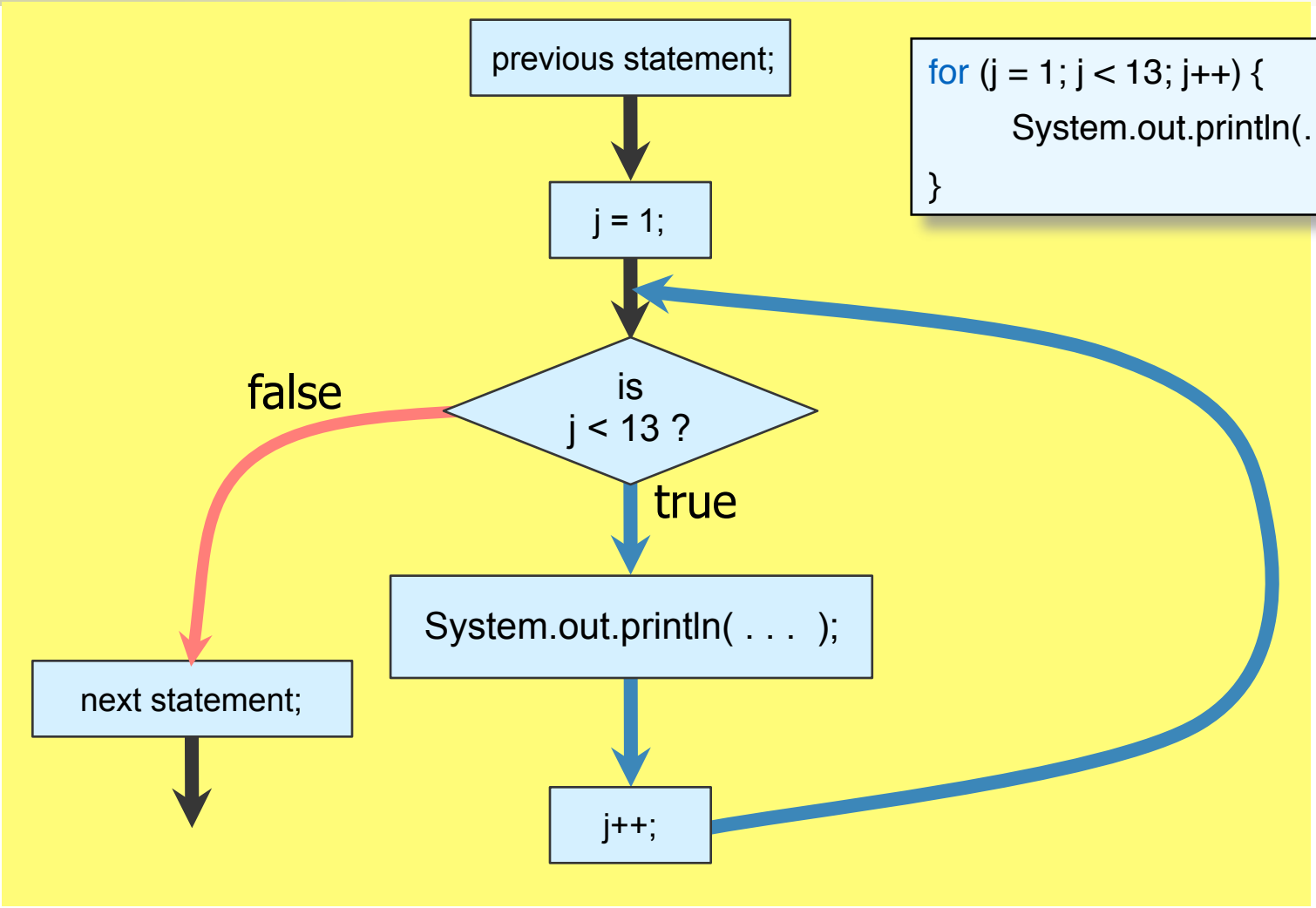
- This is a very common situation:
  - initialize a variable
  - repeat a loop body until some condition is true
  - update variable in each loop
- A **`for`** loop can be used in this situation.
  - makes the three steps explicit
  - separate from the loop body

```
for ( <initialization>; <boolean expression>; <increment> )
                                        <statement>
```

PrintOneTable



Control flow of `for`




```
for (j = 1; j < 13; j++) {
        System.out.println(…);
}
```

More for Loop Examples

```
int sum = 0;
for (i = 1; i <= 1000; i++)
    sum += i;
```

Sum of the first 1000 integers

```
int product = 1;
for (i = 2; i <= 100;i += 2)
    product *= i;
```

Product of the even number below 100

```
for (j = 2; j < 40; j *= 2)
```

```
for ( int k = 100; k > 0; k--) )
```

Quiz

```
int x = 0;
int n = 1;
while ( n > 4 ) {
      x += 2;
      n++;
}
```

- What is the value of x after the above code executes?
  A. 0
  B. 2
  C. 6
  D. 8
  E. This is an infinite loop

Practice Makes Perfect

- Trying to write snippets of code to solve various problem is a very effective learning technique
- Coding Bat is a great website for doing this.
- http://www.codingbat.com/java

Quiz

```
int x = 0;
int n = 1;
while ( n < 4 ) {
      x += 2;
      n++;
}
```

- What is the value of x after the above code executes?
  A. 0
  B. 1
  C. 4
  D. 6
  E. There is an infinite loop in this code

Problem

- Write a program to print out a multiplication tables from 1 through 12.

	1	2	3	4	5	6	7	8	9	10	11	12
1	1	2	3	4	5	6	7	8	9	10	11	12
2	2	4	6	8	10	12	14	16	18	20	22	24
3	3	6	9	12	15	18	21	24	27	30	33	36
4	4	8	12	16	20	24	28	32	36	40	44	48
5	5	10	15	20	25	30	35	40	45	50	55	60
6	6	12	18	24	30	36	42	48	54	60	66	72
7	7	14	21	28	35	42	49	56	63	70	77	84
8	8	16	24	32	40	48	56	64	72	80	88	96
9	9	18	27	36	45	54	63	72	81	90	99	108
10	10	20	30	40	50	60	70	80	90	100	110	120
11	11	22	33	44	55	66	77	88	99	110	121	132
12	12	24	36	48	60	72	84	96	108	120	132	144

Generating the Table

Outer loop

Inner loop

```
for(col = 1; col < 13; col++)
    System.out.print(" " +col);
System.out.print("\n");

for(row = 1; row < 13; row++){
    System.out.print(""+row);
    for(col = 1; col < 13; col++)
        System.out.print(" " + row * col);

    System.out.print("\n");
}
```

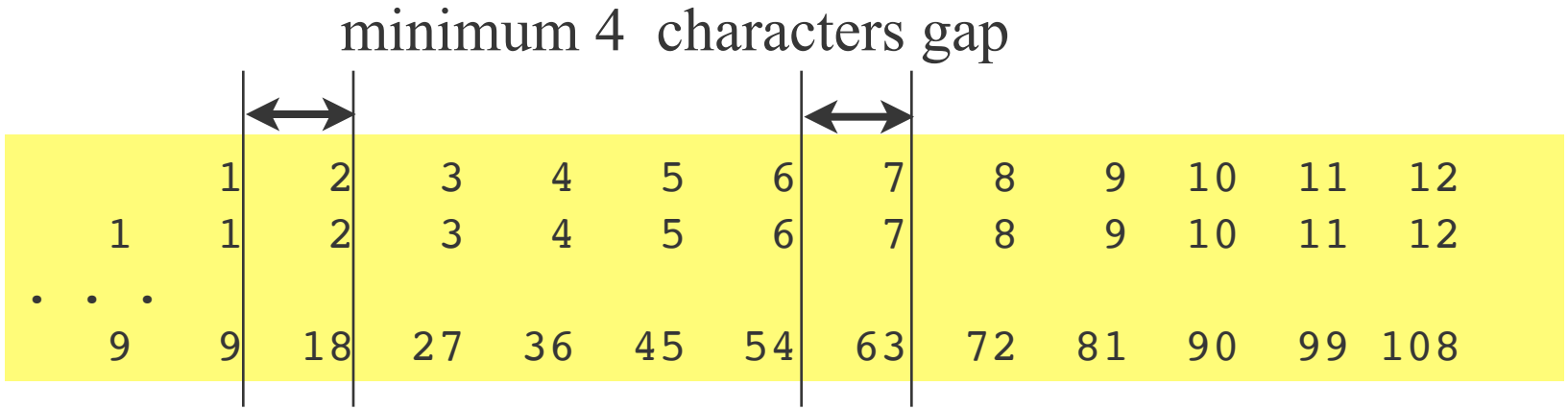
Output

```
   1  2  3  4  5  6  7  8  9  10  11  12
1  1  2  3  4  5  6  7  8  9  10  11  12
2  2  4  6  8  10  12  14  16  18  20  22  24
3  3  6  9  12  15  18  21  24  27  30  33  36
4  4  8  12  16  20  24  28  32  36  40  44  48
5  5  10  15  20  25  30  35  40  45  50  55  60
6  6  12  18  24  30  36  42  48  54  60  66  72
7  7  14  21  28  35  42  49  56  63  70  77  84
8  8  16  24  32  40  48  56  64  72  80  88  96
9  9  18  27  36  45  54  63  72  81  90  99  108
10  10  20  30  40  50  60  70  80  90  100  110  120
11  11  22  33  44  55  66  77  88  99  110  121  132
12  12  24  36  48  60  72  84  96  108  120  132  144
```

Formatting Output using printf

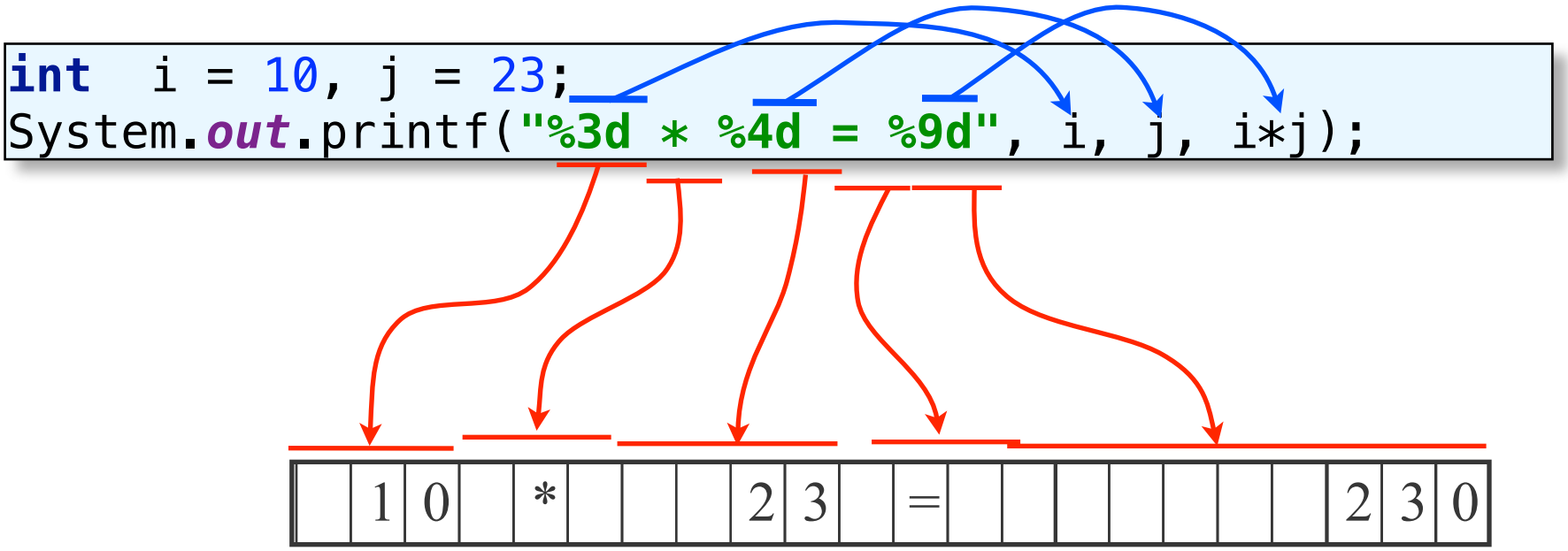
In order to control output, we can use the printf() function.

```
for(row=1; row<13; row++){
    System.out.printf("%4d", row);
    for(col=1;col<13;col++)
        System.out.print("%4d", row*col);
    System.out.print("\n");
}
```



Formatting Output

- The first argument to printf() is the string to be displayed.
- Each occurrence of a format specifier (e.g., %4d) is replaced by a matching argument.



Format Specifiers

- Integer data
  - %<min width>d e.g., %5d
- Real numbers (float and double)
  - %<min width>.<decimal places>f
  - e.g., %3.5f -- use 5 decimal places
- String
  - %s or %10s or %10.3s %-10s
- Character
  - %c or %3c

The format Method of PrintStream

- Instead of using the printf() method of PrintStream, we can achieve the same result by using the format method of PrintStream or a Formatter object.

```
System.out.printf ("%6d", 498);
```

is equivalent to

```
Formatter formatter = new Formatter(System.out);
formatter.format("%6d", 498);
```

and equivalent to

```
System.out.format("%6d", 498);
```

See API for details.

breaking out of a loop

- In some cases, it is necessary to get out of a loop.
- This is achieved using a **break** statement.

```
for(row = 1; row < 13; row++){
    System.out.print("" + row);
    for(col = 1; col < 13; col++){
        System.out.print(" " + row*col);
        if(col >= row)
            break;
    }
    System.out.print("\n");
}
```

```
     1   2   3   4   5   6   7
 1   1
 2   2   4
 3   3   6   9
 4   4   8  12  16
 5   5  10  15  20  25
 6   6  12  18  24  30  36
 7   7  14  21  28  35  42  49
 8   8  16  24  32  40  48  56
 9   9  18  27  36  45  54  63
10  10  20  30  40  50  60  70
11  11  22  33  44  55  66  77
12  12  24  36  48  60  72  84
```

breaking Out of Outer Loop

```
for(row = 1; row < 13; row++){
    System.out.print("" + row);
    for(col = 1; col < 13; col++)
        System.out.print(" " + row*col);
    System.out.print("\n");
    if(col*row > 30)
        break;
}
```

```
    1   2   3   4   5   6   7   8   9  10  11  12
1   1   2   3   4   5   6   7   8   9  10  11  12
2   2   4   6   8  10  12  14  16  18  20  22  24
3   3   6   9  12  15  18  21  24  27  30  33  36
```

Skipping an iteration

- We can skip the current iteration of a loop using a **`continue`** statement.
- A continue transfers control to the test statement of the loop.

```
for(row = 1; row < 13; row++){
    if(row % 2==0)
        continue;
    System.out.printf("%4d", row);
    for(col = 1; col < 13; col++)
        System.out.printf("%4d", row*col);
    System.out.print("\n");
}
```

```
     1   2   3   4   5   6   7   8   9  10  11
   1   1   2   3   4   5   6   7   8   9  10  11
   3   3   6   9  12  15  18  21  24  27  30  33
   5   5  10  15  20  25  30  35  40  45  50  55
   7   7  14  21  28  35  42  49  56  63  70  77
   9   9  18  27  36  45  54  63  72  81  90  99
  11  11  22  33  44  55  66  77  88  99 110 121
```

Multiple statements in for loop

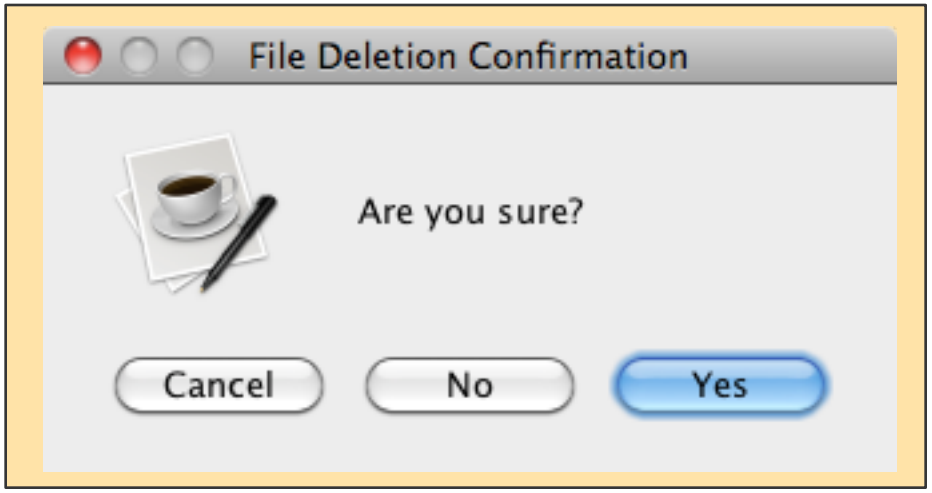
- The initialization and increment of a for loop can contain multiple statements separated by commas.

```
int i, sum;

for (i = 0,sum = 0; i <= 100;  sum += i, i++) ;

System.out.println("Sum from 1 to 100 is:" + sum);
```

Confirmation Dialog

- Used to give the user a choice between different buttons.

```
JOptionPane.showConfirmDialog(null,
        "Are you sure?",
        "File Deletion Confirmation",
        JOptionPane.YES_NO_CANCEL_OPTION);
```



Example: Confirmation Dialog

```
int choice;
choice = JOptionPane.showConfirmDialog(null,
        "Are you sure?",
        "File Deletion Confirmation",
        JOptionPane.YES_NO_CANCEL_OPTION);

if(choice == JOptionPane.YES_OPTION)
    System.out.println("You chose to delete the file”);
else if (choice == JOptionPane.NO_OPTION)
    System.out.println("You chose not to delete the file");
else
    System.out.println("You chose to cancel");
```

Caution: Reals and Equality

1

```
float count = 0.0f;

while ( count != 1.0f ) {
    count = count + 0.3333333f;
}          //seven 3s
```

2

```
float count = 0.0f;

while ( count != 1.0f ) {
    count = count + 0.33333333f;
}          //eight 3s
```

Using Real Numbers

Loop 2 terminates, but Loop 1 does not because only an approximation of a real number can be stored in a computer's memory.

Loop Pitfall – 2a

1

```
int result = 0; double cnt =
1.0;
while (cnt <= 10.0){
    cnt += 1.0;
    result++;
}
System.out.println ( result);
```

→ 10

2

```
int result = 0; double cnt = 0.111111111;
while ( cnt <= 1.11111111){
    cnt += 0.111111111;
    result++;
}
System.out.println ( result);
```

→ 11

Using Real Numbers

Loop 1 prints out 10, as expected, but Loop 2 prints out 11. The value 0.1 cannot be stored precisely in computer memory.

Problem

- Write a program to compute the average of integer inputs.
- The exact number of inputs is not fixed.

Problem

- Write a program to reverse a string.
- Write a program to test if a string is a palindrome.

Problem

- Write a program to convert an integer from decimal to binary.
  - How about conversion to a base other than binary?
  - How about negative numbers?
  - What about converting to bases other than 2?

Problem

- Write a program to test if an integer input by the user is a prime number.
  - Start with a simple test that checks for each potential factor.
- How can we improve on this?
  - Do we have to check every factor?
  - Only odd numbers other than 2
  - Nothing larger than the square root of the number