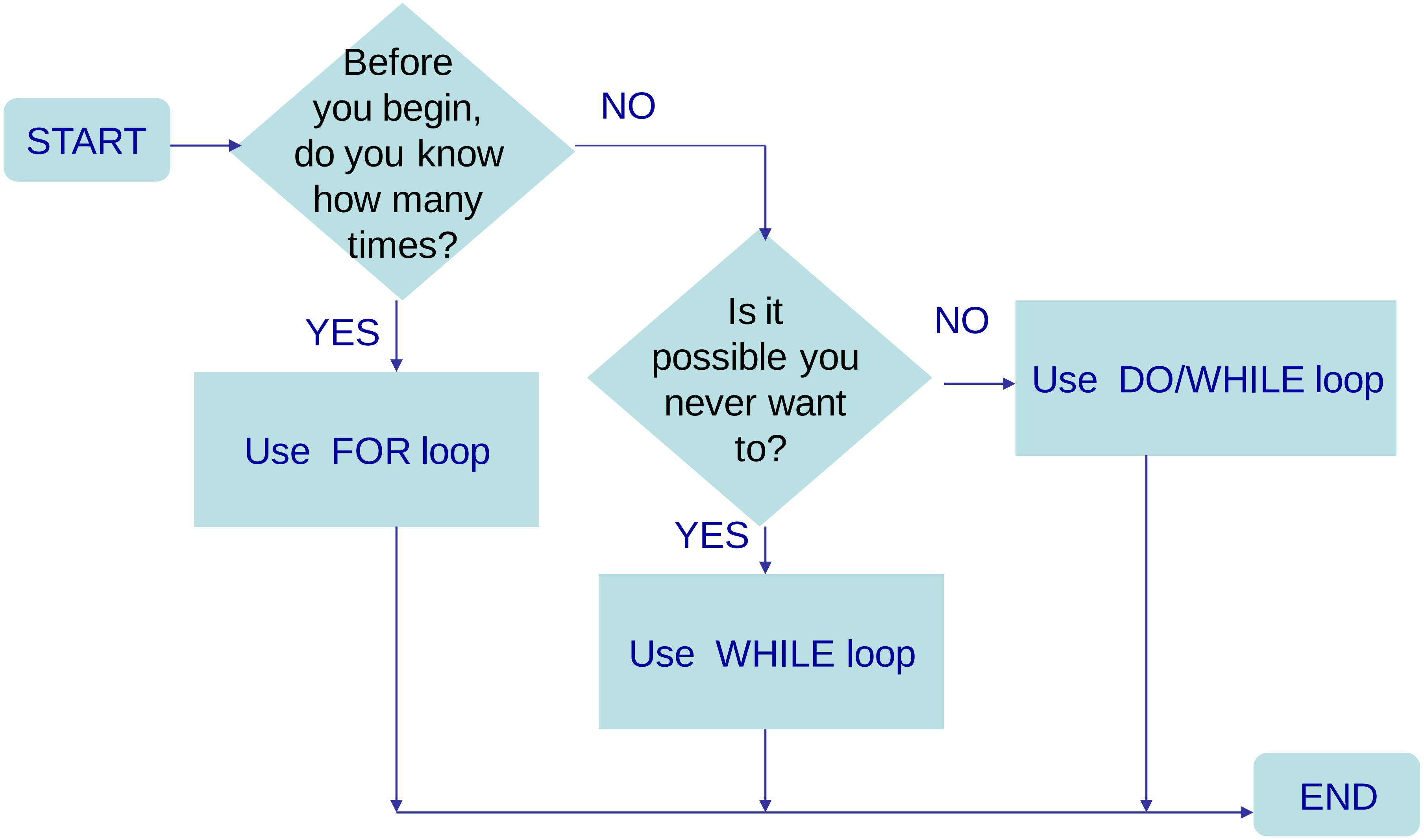
Looping – First Review

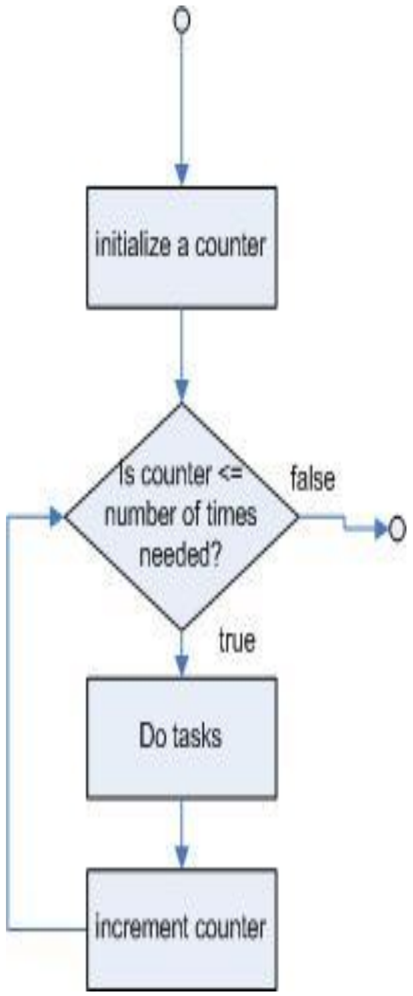
Objectives:

- Introduce the 3 types of loops: while, for and do/while
- Demonstrate the importance of loops in problem solving
- encourage a systematic approach to loop design
- trace code with loops

Doing it a LOT of times!!


START
Before you begin, do you know how many times?
NO
YES
Use FOR loop
Is it possible you never want to?
NO
Use DO/WHILE loop
YES
Use WHILE loop
END

Do you know before you start how many times you want to repeat?



Use a for loop:

```
for (int counter = 1; counter <= 14; counter ++) {
   // Do it
}
```

```
for (int counter = 0; counter < 14; counter ++) {
   // Do it
}
```

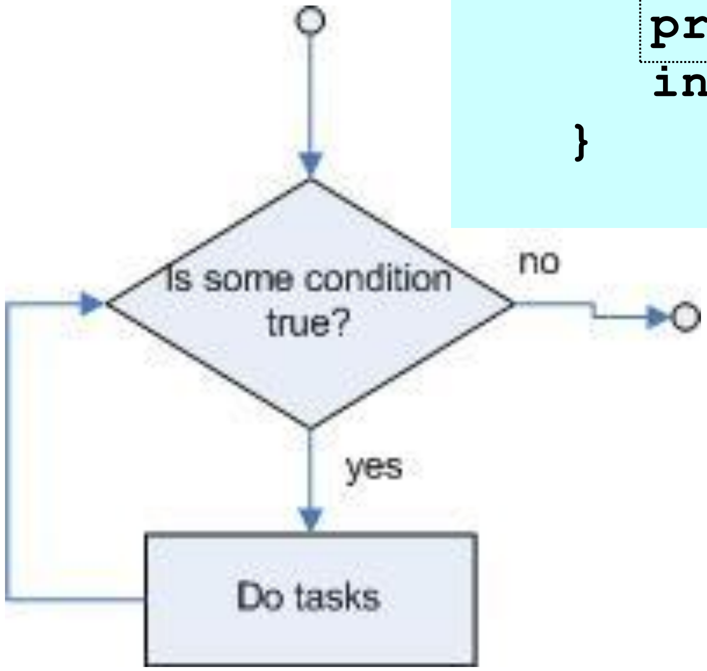
```
for (int i = 0; i < 14; i ++) {
   // Do it
}
```

Yes!! **i** is allowed here.

Do you have a sentinel to indicate how many times to repeat?

Use a while loop:

```
int sum = 0;
int product = 1;
int inputNumber = stdin.nextInt();
while (inputNumber != 0) {
    sum = sum + inputNumber;
    product = product * inputNumber;
    inputNumber = stdin.nextInt();
}
```



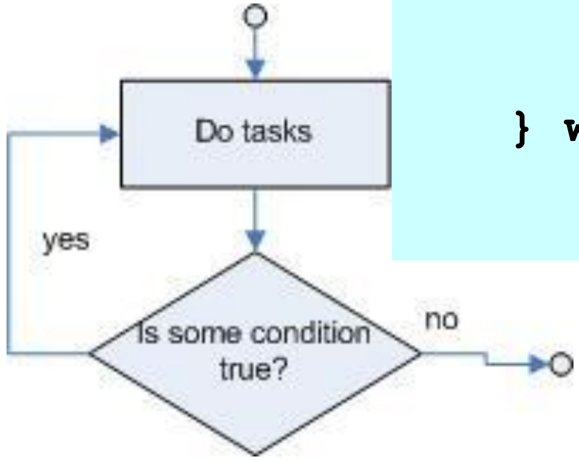
A cute Shortcut:

```
sum += inputNumber;
    product *= inputNumber;
```

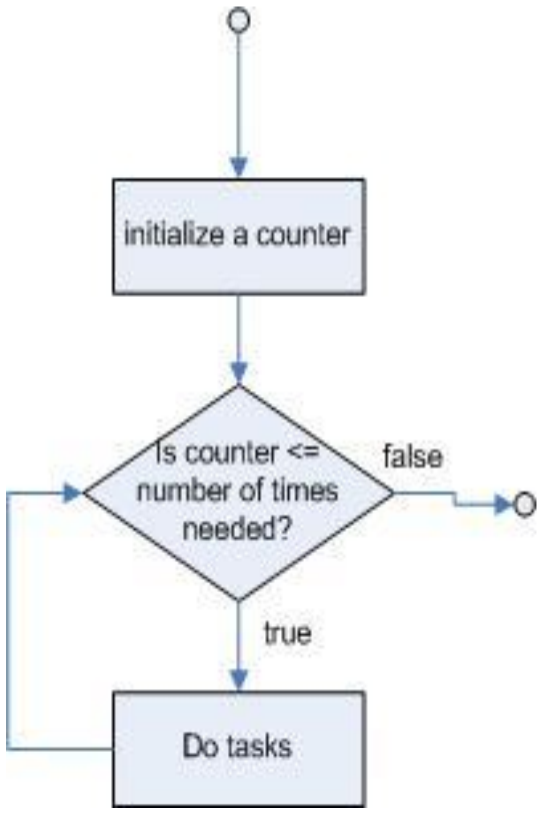
When (exactly) should you check that condition?

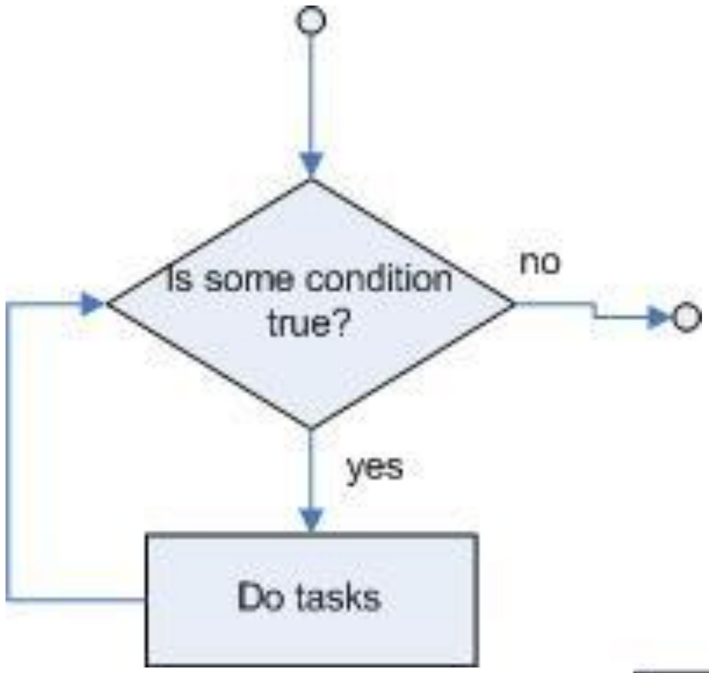
To check at the end: use a do/while loop:

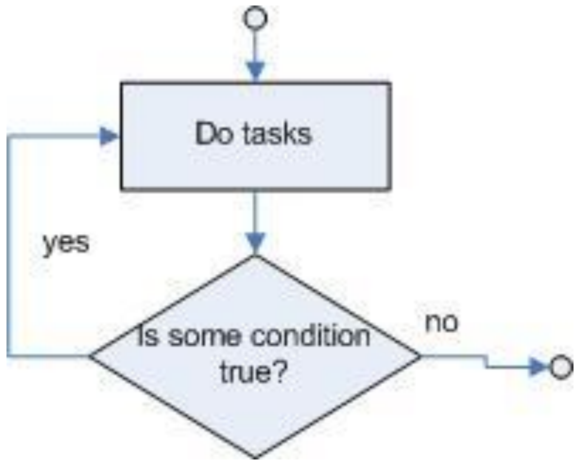
```
int sum = 0;
int product = 1;
int inputNumber;
do {
   inputNumber = stdin.nextInt();
   sum += inputNumber;
   if (inputNumber !=0) product *= inputNumber;
} while (inputNumber != 0);
```



The 3 Java Loops







Questions for Loop Design

Questions to consider in loop design and analysis

- What initialization is necessary for the loop's test expression?
- What initialization is necessary for the loop's processing?
- What causes the loop to terminate?
- What actions should the loop perform?
- What actions are necessary to prepare for the next iteration of the loop?
- What conditions are true and what conditions are false when the loop is terminated?
- When the loop completes what actions are need to prepare for subsequent program processing?

Trace: What does it do?

```
int valuesProcessed = 0;
double valueSum = 0;
double valueProduct = 1;

double value = stdin.nextDouble();

while (value > 0) {
  valueSum += value;
  valueProduct *= value;
  valuesProcessed++;
  value = stdin.nextDouble();
}

if (valuesProcessed > 0) {
  System.out.println(“Sum = " + valueSum);
  System.out.println(“   Product = " + valueProduct);
}
else {
  System.out.println("No list to use for calculations");
}
```

Nested Loops

```
int m = 2;
int n = 3;
for (int i = 0; i < n; ++i) {
  System.out.println("i is " + i);
  for (int j = 0; j < m; ++j) {
    System.out.println("   j is " + j);
  }
}
```

What does this one do?

```
System.out.print("Enter a positive number: ");
int number = stdin.nextInt();
do {
   int digit = number % 10;
   System.out.print(digit);
   number = number / 10;
} while (number != 0);
```

Problem 0

Generate a multiplication table for the values 1 times 1 through X times Y, where X and Y are values input by the user.

Sample I/O screen

```
M U L T I P L I C A T I O N   T A B L E
This program.

Input:
  X ==> 3
  Y ==> 4

  1 X 1 = 1
  1 X 2 = 2
  1 X 3 = 3
  1 X 4 = 4
  2 X 1 = 2
  2 X 2 = 4
  2 X 3 = 6
```
etc

```
/* Title: M U L T I P L I C A T I O N   T A B L E
 * Purpose:  Prints out a multiplication table for all integers up to X times Y
* Date:   February 2007
 */
import java.util.Scanner;

public class MultiplicationTable {

    public static void main(String[] args) {
          // Initialize the input Scanner and Introduce the program
          Scanner stdin = new Scanner(System.in);
          System.out.println(" M U L T I P L I C A T I O N   T A B L E");

          makeMultiplicationTable(stdin);
    }

    public static void makeMultiplicationTable (Scanner standardIn) {

          //Get two values
          System.out.println("Input: ");
          System.out.print("  X ==> ");
          int X = standardIn.nextInt();
          System.out.print("  Y ==> ");
          int Y = standardIn.nextInt();

          //Make the multiplication table
          for (int counterX = 1; counterX <= X; counterX++) {
                    for (int counterY = 1; counterY <=Y; counterY++) {
                              System.out.print(counterX + " x " + counterY );
                              System.out.println(" = " + counterX * counterY);
                    }
          }

    }

}
```

Problem 1

- Write a piece of Java code that uses a `while` loop to repeatedly prompt the user to type a number until the user types a non-negative number, then square it.

  – Example log of execution:

```
Type a non-negative integer: -5
Invalid number, try again: -1
Invalid number, try again: -235
Invalid number, try again: -87
Invalid number, try again: 11
11 squared is 121
```

While loop answer

- Solution:

```
System.out.print("Type a non-negative integer: ");
int number = console.nextInt();

while (number < 0) {
    System.out.print("Invalid number, try again: ");
    number = console.nextInt();
}

int square = number * number;
System.out.println(number + " squared is " + square);
```

  – Notice that `number` has to be declared outside the loop in order to remain in scope.

Problem 2

Suppose that a rubber ball is dropped from the 6th floor of ECS, a height of 25.7 m above the level of the 1st floor concrete. Of course, the ball bounces for a while. Assume that on each bounce it comes back up to half its previous height. Let's write a program that simulates the behaviour of the ball. The program should display the number of each bounce and the height of that bounce, repeating until the height of the ball is very small (e.g., less than 1 millimeter.)

Sample I/O screen

```
B A L L   B O U N C E R
   by LillAnne Jackson
This program lists the height of the bounces of a ball, given a specified
   initial height.

Input:  Initial ball height (in m) ==> 25.7

  Bounce          Height (m)
  ----------      ---------
  0               25.7
  1               12.85
  2               6.425
  3               3.2125
  4               1.60125
  5               0.800625
  6               0.9003125
  7               0.45015625
  8               0.225078125
  9               0.1125380625
  10              0.05626903125
  11              0.028134515625
  12              0.0140672578125
  13              0.00703362890625
```

```
/* Title: Ball Bouncer
 * Purpose:This program lists the height of the bounces of a ball,
  * given a specified initial height.
 *
* Date:   February 2007
 */

import java.util.Scanner;

public class BallBouncer {

    public static void main(String[] args) {

            // Initialize the input Scanner and Introduce the program
            Scanner stdin = new Scanner(System.in);
            System.out.println(" B A L L   B O U N C E R");

            //Get initial height
            System.out.print("Input:  Initial ball height (in m) ==> ");
            double height = stdin.nextDouble();
            //initialize counter
            int counter = 0;

            //print out and recalculate height until final height has occurred
            while(height > 0.001) {
                    System.out.println("  " + counter + "     " + height);
                    height /= 2;
                    counter++;
            }
            System.out.println("  " + counter + "     " + height);

    }

}
```

Problem 3

It is sometimes useful to make a program pause for a specified length of time. For example, a program that displays information too quickly may need to pause to give a user time the read the information. Lets write a method that, given a length of time will make a program pause for that length of time. (This is a technique called *busy-waiting*.)

`java.lang` has a method `currentTimeMillis()`
 - It returns the number of milliseconds since January 1, 1970

Example for Problem 2

```
double then = currentTimeMillis();
double now = currentTimeMillis();
```