Using Loops

- Sam earns $100 per day with a daily raise of $100.
- Sue earns $0.01 per day with a salary that doubles every day.
- How many days pass before Sue is being paid more?

```java
double sam = 100.0;
double sue = 0.01;
// day counter
int count = 0;
// loop until sue earns more
while ( sue < sam ) {
    // count one day at a time
    count++;
    // increase salaries
    sam += 100.0;
sue *= 2;
}
// print out final results
System.out.println( "..." );
```

Note: when this code is complete, we will have the rates of pay set for what each person will get on the next day, not on the day the earnings actually achieved parity.

Another Type: for Loops

- Another type, typically used for a fixed number of iterations

```
for( init; condition; update ) {
    instructions;
}
```

The instructions execute if condition is true.
Then do update step, check condition again, and repeat if still true.

Example: Tiling a Row with Squares

- Say we want to produce a chessboard pattern.
- How do we use a while loop to do it?

```java
int rep = 0;
int size = windowSize / numSquares;
while ( rep < numSquares ) {
    int x = rep * size;
    Rectangle rect = new Rectangle( ... );
    if ( ( rep % 2 ) == 0 ) {
        rect.setBackground( Color.red );
    }
    window.add( rect );
    rep++;
}
```

Every loop has four parts:
- Initialization – establish the state before looping
- Condition – a boolean to control when loop stops
- Main work – code to be repeatedly executed
- Update – code that moves the loop toward termination.
Another Type: **for** Loops

- A simple loop to add all the integers from 1 to 100

```java
int sum = 0;
for( int i = 1; i <= 100; i++ )
{
    sum = sum + i;
    System.out.println( sum );
}
```

### Tiling with a **for** Loop

- Again, we produce our row of tiles, using **for**
- Code may execute 0 or more times

```java
for ( int col = 0; col < numSquares; col++ )
{
    int x = col * size;
    Rectangle rect = new Rectangle(...);
    if ( col % 2 == 0 )
    {
        rect.setBackground( Color.red );
    }
    window.add( rect );
}
```

### Local Variables in Loops

- A variable *declared* inside a code block `{...}` is **local** to that block, whether the block is an entire class, a method, or a control structure (`if`, `else`, `for`, `while`, `do`, etc.)

#### This won't work!

```java
int n1 = 0;
int n2 = 0;
int fib = 1;
for ( int i = 1; i <= 10; i++ )
{
    n1 = n2;
    n2 = fib;
    fib = n1 + n2;
    System.out.print( i + " " + n1 + " " + n2 + " " + fib );
}
```

#### Neither will this!

```java
int n1 = 0;
int n2 = 0;
int fib = 1;
for ( int i = 1; i <= 10; i++ )
{
    n1 = n2;
    n2 = fib;
    fib = n1 + n2;
    System.out.print( i + " " + n1 + " " + n2 + " " + fib );
}
```

#### This will work properly now

```java
int n1 = 0;
int n2 = 0;
int fib = 1;
int i = 0;
for ( i = 1; i <= 10; i++ )
{
    n1 = n2;
    n2 = fib;
    fib = n1 + n2;
}
System.out.print( i + " " + fib );
```
More Examples of for Loops

There are many ways to get the same effect.

Consider printing out all integers between 5 and 10:

```java
for (int n = 1; n <= 10; n++)
    System.out.println(n);
```

Print all odd numbers from 1 to 100:

```java
for (int n = 1; n <= 100; n += 2)
    System.out.println(n);
```

This Week & Next

Meetings this week:
- Monday/Wednesday: CS Lab (16 Wing)
- Tuesday/Friday: regular classroom

Midterm 1: Monday, 02 March

Practice midterm: Available on my CS120 webpage

Office Hours: Wing 212
- Monday/Wednesday: 11:00 AM–12:00 PM
- Tuesday: 3:15 PM–4:15 PM
- Friday: 11:00 AM–11:40 AM

Lab and Tutor Hours: On my website