What we will cover:
Greatest Common Divisor
Factorial
*break* and *continue*

Finish Lecture 12 timing section

Greatest Common Divisor
You can find the greatest common divisor of two numbers by checking the values between 1 and the two numbers to find the largest number that evenly divides both numbers. We know that 1 will divide both numbers, so we start at two and test values in between until we have reached one of the two numbers.

```java
int n1, n2, gcd = 1, k = 2;
Scanner input = new Scanner(System.in);
System.out.print("Enter the first integer: ");
n1 = input.nextInt();
System.out.print("Enter the second integer: ");
n2 = input.nextInt();

while (k <= n1 && k <= n2) {
    if (n1 % k == 0 && n2 % k == 0) {
        gcd = k;
    }
    k++;
}
System.out.printf("The greatest common divisor of %d and %d is %d", n1, n2, gcd);
```

Factorial
The factorial of a number, denoted by \( n! \), is the product of all positive numbers from 1 to \( n \), inclusive. The special case of the factorial is \( 0! \) which equals 1. Mathematically, factorial is defined as:

\[
n! = \prod_{k=1}^{n} k
\]

Or recursively defined as:

\[
n! = \begin{cases} 
1, & \text{if } n = 0 \\
(\ n \times (n - 1)!), & \text{if } n > 0 
\end{cases}
\]

\( 0! = 1 \)
\( 3! = 1 \times 2 \times 3 = 6 \)
\( 5! = 1 \times 2 \times 3 \times 4 \times 5 = 120 \)

Some code to calculate a factorial:

```java
int number = 0, product = 1;
Scanner input = new Scanner(System.in);
System.out.print("Please enter a number: ");
number = input.nextInt();

for (int i = 1; i <= number; i++) {
    product *= i;
}

System.out.printf("%d! = %d", number, product);

Notice that the special case of 0! is handled in both definitions and the program by essentially declaring that product of nothing is one.

**break and continue**
The keywords `break` and `continue` can be used in a loop statement to affect the execution of the loop. Using break in a loop will end execution of the loop immediately. This is similar to its behavior in a switch statement.

```java
int sum = 0;
int number = 0;
while (number < 20) {
    number++;
    sum += number;
    if (sum >= 100) {
        break;
    }
}

System.out.println("Number: " + number);
System.out.println("Sum: " + sum);
```

If the sum reaches 100 or greater, the loop is terminated and the control of the program jumps to the print statements.

The keyword `continue` can be used to end the current iteration immediately and start the next one. This means it will skip all code below the `continue` and start the next iteration of the loop.

```java
int sum = 0;
int number = 0;
while (number < 20) {
    number++;
    if (number == 10 || number == 11) {
        continue;
    }
    sum += number;
}

System.out.println("Sum: " + sum);
```

This code will not add 10 or 11 to the sum because of the `continue` keyword.

Loops can be written without the use of `break` and `continue`. Unless the use of these keywords drastically improves readability or simplifies a problem, you should simply use appropriate conditions to control your loop.