Lecture 14
The *do*-statement

• Syntax

```
do
  ONE-statement
while ( loop-continuation-condition ) ;
```

```
do
  { 
    statement1
    statement2
    ....
  }
while ( loop-continuation-condition ) ;
```
Flow chart representing a **while-statement**:

- **Body of do-loop**
- **Statement 1**
- **Statement 2**
- **Do-statement terminates**
Example:

```java
int a = 1;
do {
    System.out.println(a);
    a++;
} while (a <= 4)
System.out.println("Done");
```

Flow chart of this program:
Break and continue statement used in a do-statement

- The `break` statement will cause the do-statement to terminate immediately:

```plaintext
do
{
.... break;
....
}
while ( loop-continuation-condition ) ;
```

The execution will continue with the statement following the do-statement.

- The `continue` statement will cause the do-statement to jump to the end of the body:

```plaintext
do
{
.... continue;
....
}
while ( loop-continuation-condition ) ;
```

The execution will continue with testing the loop-continuation-condition.

If the condition is true, the do-statement is repeated, and otherwise (if the condition is false), the do-statement is terminated.
Minimizing Numeric Errors

```java
1  public class TestSum {
2      public static void main(String[] args) {
3          // Initialize sum
4          float sum = 0;
5
6          // Add 0.01, 0.02, ..., 0.99, 1 to sum
7          for (float i = 0.01f; i <= 1.0f; i = i + 0.01f)
8              sum += i;
9
10         // Display result
11         System.out.println("The sum is " + sum);
12      }
13  }
```

```java
// Initialize sum
double sum = 0;

// Add 0.01, 0.02, ..., 0.99, 1 to sum
for (double i = 0.01; i <= 1.0; i = i + 0.01)
    sum += i;
```
Adding a very small number to a very big number can have no effect if the result requires more precision than the variable can store. For example, the inaccurate result of 100000000.0 + 0.000000001 is 100000000.0. To obtain more accurate results, carefully select the order of computation. Adding smaller numbers before bigger numbers is one way to minimize errors.
Revisit: Command line arguments

• java className arg1 arg2 ...

• All arguments are passed as String.
Nested loop

- Nested loops consist of an outer loop and one or more inner loops. Each time the outer loop is repeated, the inner loops are reentered, and started anew.
Using *nested for-loops* in simple graphical applications

- [http://www.mathcs.emory.edu/~cheung/Courses/170/Syllabus/07/graphics.html](http://www.mathcs.emory.edu/~cheung/Courses/170/Syllabus/07/graphics.html)
Nested loop example: parsing

- http://www.mathcs.emory.edu/%7Echeung/Courses/170/Syllabus/07/parsing.html