Announcements

• PS 2 is due 9/29

• How many hours are you studying for this class each week?

• Reading assignment for this lecture: Chapter 5
Loops
Print "Welcome to Java!" 100 times

```
System.out.println("Welcome to Java!");
System.out.println("Welcome to Java!");
...
System.out.println("Welcome to Java!");
System.out.println("Welcome to Java!");
```

With a `while` loop

```
int count = 0;
while (count < 100) {
    System.out.println("Welcome to Java");
    count++;
}
```
while (loop-continuation-condition) {
    // loop-body;
    Statement(s);
}

int count = 0;
while (count < 100) {
    System.out.println("Welcome to Java!");
    count++;
}
Trace while Loop

```java
int count = 0;
while (count < 2) {
    System.out.println("Welcome to Java!");
    count++;
}
```
int count = 0;

while (count < 2) {
    System.out.println("Welcome to Java!");
    count++;
}

(count < 2) is true
int count = 0;
while (count < 2) {
    System.out.println("Welcome to Java!");
    count++;
}
Trace while Loop, cont.

```java
int count = 0;
while (count < 2) {
    System.out.println("Welcome to Java!");
    count++;
}
```

Increase count by 1
count is 1 now
int count = 0;

while (count < 2) {
    System.out.println("Welcome to Java!");
    count++;
}

(count < 2) is still true since count is 1
Trace while Loop, cont.

```java
int count = 0;
while (count < 2) {
    System.out.println("Welcome to Java!");
    count++;
}
```
Trace while Loop, cont.

```java
int count = 0;
while (count < 2) {
    System.out.println("Welcome to Java!");
    count++;
}
```

Increase count by 1  
count is 2 now
int count = 0;

while (count < 2) {
    System.out.println("Welcome to Java!");
    count++;
}

(count < 2) is false since count is 2 now
int count = 0;
while (count < 2) {
System.out.println("Welcome to Java!");
    count++;
}

The loop exits. Execute the next statement after the loop.
if vs. while

```java
int count = 0;
if (count < 100) {
    System.out.println("Welcome to Java");
    count++;
}
```

Prints once

```java
int count = 0;
while (count < 100) {
    System.out.println("Welcome to Java");
    count++;
}
```

Prints 100 times
public static void main(String[] args) {
    int n1 = (int)(System.currentTimeMillis() % 10);
    int n2 = (int)(System.currentTimeMillis() / 7 % 10);
    Scanner input = new Scanner(System.in);
    System.out.print("What is \( + \) n1 \( + \) \( + \) n2 \( + \) ? \( \)\);
    int sum = input.nextInt();
    System.out.println(n1 + " + " + n2 + " = " +
                       sum + " is " + (n1 + n2 == sum));
}

// Ask once!

public static void main(String[] args) {
    int n1 = (int)(Math.random() * 10);
    int n2 = (int)(Math.random() * 10);
    Scanner input = new Scanner(System.in);
    System.out.print("What is \( + \) n1 \( + \) \( + \) n2 \( + \) ? \( \)\);
    int sum = input.nextInt();
    while (n1 + n2 != sum) {
        System.out.println("Try again. What's \( + \) n1 \( + \) \( + \) n2 \( + \) ? \( \)\);
        sum = input.nextInt();
    }
    System.out.println("Correct!");
}

// Ask repeatedly until correct!
Caution

Don’t use floating-point values for equality checking in a loop control. Since floating-point values are approximations for some values, using them could result in imprecise counter values and inaccurate results. Consider the following code for computing $1 + 0.9 + 0.8 + \ldots + 0.1$:

double item = 1;
double sum = 0;
while (item != 0) { // No guarantee item will be 0
    sum += item;
    item -= 0.1;
}
System.out.println(sum);
For loops

for (initial-action; loop-continuation-condition; action-after-each-iteration) {
    // loop body;
    Statement(s);
}

int i;
for (i = 0; i < 100; i++) {
    System.out.println("Welcome to Java!");
}
Trace for Loop

```java
int i;
for (i = 0; i < 2; i++) {
    System.out.println("Welcome to Java!");
}
```
int i;
for (i = 0; i < 2; i++) {
    System.out.println("Welcome to Java!");
}

Execute initializer
i is now 0
Trace for Loop, cont.

```java
int i;
for (i = 0; i < 2; i++) {
    System.out.println( "Welcome to Java!");
}
```

(i < 2) is true since i is 0
Trace for Loop, cont.

```java
int i;
for (i = 0; i < 2; i++) {
    System.out.println("Welcome to Java!");
}
```
Trace for Loop, cont.

```java
int i;
for (i = 0; i < 2; i++) {
    System.out.println("Welcome to Java!");
}
```

Execute adjustment statement
i now is 1
Trace for Loop, cont.

```java
int i;
for (i = 0; i < 2; i++) {
    System.out.println("Welcome to Java!");
}
```

(i < 2) is still true since i is 1
Trace for Loop, cont.

```java
int i;
for (i = 0; i < 2; i++) {
    System.out.println("Welcome to Java!");
}
```

Print Welcome to Java
Trace for Loop, cont.

```java
int i;
for (i = 0; i < 2; i++) {
    System.out.println("Welcome to Java!");
}
```

Execute adjustment statement
i now is 2
int i;
for (i = 0; i < 2; i++) {
    System.out.println("Welcome to Java!");
}

(i < 2) is false since i is 2
int i;
for (i = 0; i < 2; i++) {
    System.out.println("Welcome to Java!");
}

Exit the loop. Execute the next statement after the loop.
Note

If the **loop-continuation-condition** in a **for** loop is omitted, it is implicitly true. Thus the statement given below in (a), which is an infinite loop, is correct. Nevertheless, it is better to use the equivalent loop in (b) to avoid confusion:

```
for ( ; ; ) {
   // Do something
}
```

(a)

---

```
while (true) {
   // Do something
}
```

(b)
Caution

Adding a semicolon at the end of the **for** clause before the loop body is a common **mistake**, as shown below:

```java
int i;
for (i=0; i<10; i++)
{
    System.out.println("i is "+i);
}
```
Caution, cont.

Similarly, the following loop is also wrong:

```java
int i=0;
while (i < 10);
{
    System.out.println("i is " + i);
    i++;
}
```

Logic Error
Which loop to use?

The two forms of loop statements, `while` and `for`, are expressively equivalent; that is, you can write a loop in any of these forms. For example, a `while` loop in (a) in the following figure can always be converted into the following `for` loop in (b):

```
while (loop-continuation-condition) {
    // Loop body
}
```

```
for ( ; loop-continuation-condition; ) {
    // Loop body
}
```

(a)                                       (b)

Equivalent

A `for` loop in (a) in the following figure can generally be converted into the following `while` loop in (b) except in certain special cases:

```
for (initial-action;
    loop-continuation-condition;
    action-after-each-iteration) {
    // Loop body;
}
```

```
initial-action;
while (loop-continuation-condition) {
    // Loop body;
    action-after-each-iteration;
}
```

(a)                                       (b)
Recommendations

• In general, a **for** loop may be used if the number of repetitions is known, as, e.g., when you need to print a message 100 times.

• A **while** loop may be used if the number of repetitions is not known, as in the case of reading the numbers until the input is 0.
Nested loops

Write a program that uses nested for loops to print the following patterns.

```
    ****
   ****
   ****
   ****
      *
      **
      ***
      ****
```

```
    ****
   ****
   ****
   ****
      **
      ****
      ******
      ********
```
Problem: Predicting the future tuition

Suppose that the tuition for a university is $10,000 this year and tuition increases 7% every year.

In how many years will the tuition be doubled?
Problem: Predicating the future tuition

double tuition = 10000;  int year = 0  // Year 0
tuition = tuition * 1.07;  year++;  // Year 1
     tuition = tuition * 1.07;  year++;  // Year 2
     tuition = tuition * 1.07;  year++;  // Year 3
...
Using `break` and `continue`

Examples for using the `break` and `continue` keywords:
public class TestBreak {
    public static void main(String[] args) {
        int sum = 0;
        int number = 0;

        while (number < 20) {
            number++;
            sum += number;
            if (sum >= 100)
                break;
        }
        System.out.println("The number is " + number);
        System.out.println("The sum is " + sum);
    }
}
public class TestContinue {
    public static void main(String[] args) {
        int sum = 0;
        int number = 0;

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

        System.out.println("The sum is " + sum);
    }
}