Nested Loops
Lecture 16

Borrowing from slides by Alan Hu, Kurt Eiselt, Paul Carter, and Tamara Munzner

Reading Assignments
- Reading for this week: looping/iterations
  - Edition 2: Ch. 6.1-6.5
  - Edition 3: Ch. 7.1-7.5

Recap: Loops
- One more key idea that makes computers powerful:
  - Remembering things: variables
  - Calculating things: expressions
  - Making decisions: if
  - Repeating things: while, for
- Loops allow repetitive operations in programs
  - aka iteration, repetition

Recap: Climbing Stairs
- Am I at the top of the stairs?
  - No.
  - Climb up one step.
  - Am I at the top of the stairs?
  - No.
  - Climb up one step.
  - Am I at the top of the stairs?
  - No.
  - Climb up one step.
  - ...and so on...

Recap: while Statement
while ( boolean expression )
body
- Control flow
  - Is boolean expression true? If not, exit loop.
  - Execute body of loop.
  - Check again, is boolean expression still true? If not, exit loop.
  - Execute body of loop.
  - ... and so on ...
  - Repetition continues until expression false.
  - Then processing continues with next statement after loop

Recap: Flow Charts
how if statement works

how while statement works
Recap: Using `while` Statements

```java
public class WhileDemo
{
    public static void main (String[] args)
    {
        int limit = 3;
        int counter = 1;

        while (counter <= limit)
        {
            System.out.println("The square of " + counter + " is " + (counter * counter));
            counter = counter + 1;
        }
        System.out.println("End of demonstration");
    }
}
```

```
limit 3  counter 1
```

```
 limit 3  counter 2  Is counter <= limit? yes
```

```
Is counter <= limit? yes
```

```
The square of 1 is 1" printed on monitor
```
Recap: Using while Statements

```java
class WhileDemo {
    public static void main (String[] args) {
        int limit = 3;
        int counter = 1;
        while (counter <= limit) {
            System.out.println("The square of "+ counter + " is " + (counter * counter));
            counter = counter + 1;
        }
        System.out.println("End of demonstration");
    }
}
```

**limit** 3  **counter** 2  Is counter <= limit? yes

```
The square of 2 is 4" printed on monitor
```
Recap: Using while Statements

```java
public class WhileDemo {
    public static void main (String[] args) {
        int limit = 3;
        int counter = 1;
        while (counter <= limit) {
            System.out.println("The square of " + counter + ", is " + (counter * counter));
            counter = counter + 1;
        }
        System.out.println("End of demonstration");
    }
}
```

Is counter <= limit? NO!

```
limit 3 counter 4
```

Recap: Infinite Loops

```java
public class WhileDemo {
    public static void main (String[] args) {
        int limit = 3;
        int counter = 1;
        while (counter >= counter) {
            System.out.println("The square of " + counter + ", is " + (counter * counter));
            counter = counter + 1;
        }
        System.out.println("End of demonstration");
    }
}
```

- if termination condition always true, loop never ends
- infinite loop goes forever

Recap: Quasi-Infinite Loops

```java
public class WhileDemo {
    public static void main (String[] args) {
        int limit = 3;
        int counter = 1;
        while (counter <= limit) {
            System.out.println("The square of " + counter + ", is " + (counter * counter));
            counter = counter - 1;
        }
        System.out.println("End of demonstration");
    }
}
```

- good termination condition
- but process never gets closer to condition

Recap: Using while Statements

```java
public class WhileDemo {
    public static void main (String[] args) {
        int limit = 3;
        int counter = 3;
        while (counter <= limit) {
            System.out.println("The square of " + counter + ", is " + (counter * counter));
            counter = counter + 1;
        }
        System.out.println("End of demonstration");
    }
}
```

```
limit 3 counter 4
```

Objectives for Today

- Understand the concept of looping and how to write simple while loops
- Understand and use for loops
- Understand how to nest loops
do-while: a Variant of the while Loop

- In some situations, the body must be executed at least once, before the termination condition can be evaluated.
- Example:
  - Use the scanner class to ask the user for a password, repeat until password is correct.
  - This can be done with the do-while loop.

```java
Do
  body
while( boolean expression );
```

Another Looping Example: Loan Amortization

- How about computing your account balance after some number of years?

```java
double balance = initBalance;
int years = 0;
while (years < term) {
  intAccrued = balance*intRate/100;
  balance += (intAccrued - payment);
  years++;
}
```

Java Shorthand: for Loops

- Four main parts of a loop: 1. Initialize

```java
double balance = initBalance;
int years = 0;
while (years < term) {
  intAccrued = balance*intRate/100;
  balance += (intAccrued - payment);
  years++;
}
```
Java Shorthand: for Loops

- Four main parts of a loop: 2. Loop Test

```java
double balance = initBalance;
int years = 0;
while (years < term) {
    intAccrued = balance*intRate/100;
    balance += (intAccrued - payment);
    years++;
}
```

- Four main parts of a loop: 3. Compute stuff

```java
double balance = initBalance;
int years = 0;
while (years < term) {
    intAccrued = balance*intRate/100;
    balance += (intAccrued - payment);
    years++;
}
```

- Four main parts of a loop: 4. Update loop variable

```java
double balance = initBalance;
int years = 0;
while (years < term) {
    intAccrued = balance*intRate/100;
    balance += (intAccrued - payment);
    years++;
}
```

- Four main parts of a loop: 1. Initialize

```java
double balance = initBalance;
for (int years = 0; years < term; years++) {
    intAccrued = balance*intRate/100;
    balance += (intAccrued - payment);
}
```
Java Shorthand: for Loops

- Four main parts of a loop: 4. Update loop variable

```java
double balance = initBalance;

for (int years = 0; years < term; years++) {
    intAccrued = balance * intRate / 100;
    balance += (intAccrued - payment);
}
```

- for statement

- Header has three parts
  - separated by semicolons

```java
for (int counter = 1; counter <= 3; counter = counter + 1)
```

- boolean expression: second part
  - evaluated just before loop body, like in while

```java
for (int counter = 1; counter <= 3; counter = counter + 1)
```

- executed at end of loop body
- Despite name, arbitrary calculation allowed
- could decrement, for example!
Nested Loops

A loop is just a big statement, so we can put loops inside other loops, or loops inside if statements, etc.

Very simple for loop

```java
public class SimpleLoop {
    public static void main (String[] args) {
        for (int i = 1; i <= 3; i++) {
            System.out.println(i);
        }
    }
}
```

What does it do? Prints

```
1
2
3
```

What if for every number below, want multiplication table of value times 2, x3, etc?

```
1 2 3
2 4 6
3 6 9
```

For every number printed by loop above

```
1 2 3
2 4 6
3 6 9
```

For every number printed by loop above

```
1 2 3
2 4 6
3 6 9
```

want another loop to print numbers in a row, instead
 Nested Loops

- Very simple for loop

```java
public class SimpleLoop {
    public static void main (String[] args) {
        for (int i = 1; i <= 3; i++) {
            System.out.println(i);
        }
    }
}
```

For every number printed by loop above
- want another loop to print numbers in a row, instead

How do we do that?

 Nested Loops

- Put a loop inside a loop
- trace to see how it works

```java
public class NestedLoop {
    public static void main (String[] args) {
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {
                System.out.print((i * j) + "  ");
            }
            System.out.println();
        }
    }
}
```