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

```latex
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

boolean expression

true

statement

false

how while statement works

boolean expression

true

statement

false
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");
    }
}
```

- While statement
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
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
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      Is counter <= limit? yes
Recap: Using \texttt{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  Is counter <= limit? yes

"The square of 1 is 1" 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");
    }
}
```

limit 3 counter 2
Recap: Using \texttt{while} Statements

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

        \textbf{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
Recap: Using while Statements

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 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");
    }
}
```

limit 3  counter 3
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");
    }
}
```

<table>
<thead>
<tr>
<th>limit</th>
<th>counter</th>
<th>Is counter &lt;= limit?</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3</td>
<td>yes</td>
</tr>
</tbody>
</table>
Recap: Using while Statements

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

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

counter Is counter <= limit? yes

"The square of 3 is 9" 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");
    }
}
```

limit 3  counter 4
Recap: Using while Statements

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 4    Is counter <= limit? NO!
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 4  Is counter <= limit? NO!

"End of demonstration" printed on monitor
Recap: Infinite Loops

```
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: Infinite Loops

```
public class WhileDemo {
    public static void main (String[] args) {
        int limit = 9;
        int counter = 0;

        while (counter != limit) {
            System.out.println("The square of " + counter + " is " + (counter * counter));
            counter = counter + 2;
        }
        System.out.println("End of demonstration");
    }
}
```

- process gets closer to termination condition
- but never satisfies condition, keeps going past it
Recap: Quasi-Infinite Loops

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
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 );
  ```
do-while: a Variant of the while Loop

```java
import java.util.Scanner;
public class WhileDemo {
    public static void main (String[] args) {
        final String PASSWORD= "MyPassword";
        Scanner myScanner= new Scanner( System.in );
        String input;
        do {
            System.out.println( "Enter password: " );
            input= myScanner.nextLine();
        } while( !input.equals( PASSWORD ) );
        System.out.println("Password correct...");
    }
}
```
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

Loops like this are really common:

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

- Java has `for` loop statement to make it easy:

```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: 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

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: 3. Compute stuff

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: 4. Update loop variable

```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
- Only one type declaration in for loop

```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: 2. Loop Test

```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: 3. Loop body

```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:
  1. Initialization
  2. Condition
  3. Update loop variable
  4. Update loop variable

```java
double balance = initBalance;

for (int years = 0; years < term; years++) {
    intAccrued = balance*intRate/100;
    balance += (intAccrued - payment);
}
```
public class ForDemo
{
    public static void main (String[] args)
    {
        for (int counter = 1; counter <= 3; counter = counter + 1)
        {
            System.out.println("The square of " + counter + " is " + (counter * counter));
        }
        System.out.println("End of demonstration");
    }
}
public class ForDemo
{
    public static void main (String[] args)
    {
        for (int counter = 1; counter <= 3; counter = counter + 1)
        {
            System.out.println("The square of " + counter + " is " + (counter * counter));
        }
        System.out.println("End of demonstration");
    }
}
public class ForDemo
{
    public static void main (String[] args)
    {
        for (int counter = 1; counter <= 3; counter = counter + 1)
        {
            System.out.println("The square of " + counter + " is " + (counter * counter));
        }
        System.out.println("End of demonstration");
    }
}
public class ForDemo
{
    public static void main (String[] args)
    {

        for (int counter = 1; counter <= 3; counter = counter + 1)
        {
            System.out.println("The square of " + counter + " is " + (counter * counter));
        }

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

- **Increment**: third part
  - 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.
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);
        }
    }
}
```

- What does it do?
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);
        }
    }
}
```

- What does it do? Prints

1
2
3
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);
        }
    }
}
```

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

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

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

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

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();
        }
    }
}
```