**News**
- Assignment 2 out
- it's challenging, start now!!

**Reading**
- This week: Chapter 7 all (7.1-7.4)
- Next week: 8.1, 8.5-8.7, topics 6.3 and 6.4

**String Comparison Followup**
- Why did `(name == "Kermit")` work? vs. object comparison with `equals` method
- Strings are special case
  - `intern` method stores them in central table, then equality check with `"=="` works
  - Strings are often but not always interned automatically

**Recap: While Statement**
```java
while (boolean expression) {
    body
}
```
- **Body** of loop can be
  - single statement
  - whole block of many statements in curly braces
- **Control flow**
  - body executed if expression is true
  - then boolean expression evaluated again
  - if expression still true, body executed again
  - repetition continues until expression false
  - then processing continues with next statement after loop

**Recap: If Versus While Statements**
```
if (boolean expression) {
    statement
}
```
```
while (boolean expression) {
    statement
}
```
- how if statement works
- how while statement works
Recap: Loop Termination

- Loop boolean expression aka termination condition
- Logic of termination condition must match logic in loop body for updating variables used in condition
- If termination condition always true, infinite loop never ends
- If termination condition always false, body never executed

Objectives

- Understand when and how to use for loops
- nested loops

Fun With Loops

```java
public class BeerSong
{
    public static void main (String[] args)
    {
        int beerNum = 99;
        String word = "bottles";
        while (beerNum > 0) {
            if (beerNum == 1) {
                word = "bottle";
            }
            System.out.println(beerNum + " word of beer on the wall.");
            System.out.println("If one of the bottles");
            System.out.println("should happen to fall...");
            beerNum = beerNum - 1;
            if (beerNum < 1) {
                System.out.println("No more bottles of beer on the wall.");
            }
        }
        System.out.println("End of demonstration");
    }
}
```

Fun With Loops

```java
import java.util.Scanner;
public class BeerSong2
{
    public static void main (String[] args)
    {
        int beerNum = 99;
        String word = "bottles";
        boolean keepgoing = true;
        String answer;
        Scanner in = new Scanner(System.in);
        while (beerNum > 0 && keepgoing) {
            if (beerNum == 1) {
                word = "bottle";
            }
            System.out.println(beerNum + " word of beer on the wall.");
            System.out.println("If one of the bottles");
            System.out.println("should happen to fall...");
            beerNum = beerNum - 1;
        }
    }
}
```

Fun With Loops

```java
System.out.println("Continue? [y/n]: ");
answer = in.nextLine();
if (!answer.equals("n")) {
    keepgoing = false;
}
if (beerNum < 1) {
    System.out.println("No more bottles of beer on the wall.");
}
```

Other Loop 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");
    }
}
```

- Equivalent to...
**Other Loop Statements**

```java
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");
    }
}
```

- ...this loop using for statement

**For Statement**

```java
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");
    }
}
```

- for statement

**Header has three parts**
- separated by semicolons

**Initialization**: first part
- executed only one time, at beginning

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

**Increment**: third part
- executed at end of loop body
- Despite name, arbitrary calculation allowed
- could decrement, for example!
**For Versus While Statement**

- **Initialization**
  - boolean expression
  - statement
  - increment

- **Boolean Expression**
  - true
  - false

- **Statement**

- **Flowchart Notes**
  - Flowcharts can be somewhat deceptive
  - Need initialization and incrementing/modifying in while loop too
  - Although syntax does not require it in specific spot

---

**For Versus While Statement**

- Anything that can be done with one type of loop can be done with another
  - **For** and **while** are equivalent
- **For** statement convenient when
  - Loop should be executed specific number of times
  - Number can be determined before loop starts
- **While** statement convenient when
  - Don’t know yet how many times to execute loop body
  - But can check if it’s time to end loop as you go

---

**Four Things Needed In Any Loop**

- **Initialize**
  - Test to see when looping stops
    - True
    - False
  - Do useful stuff
  - Get closer to termination

- **Stop Condition**
  - Flowcharts can be somewhat deceptive
  - Need initialization and incrementing/modifying in while loop too
  - Although syntax does not require it in specific spot

---

**Four Things Needed In Any Loop**

- Give starting values to one or more variables used in loop
- Test to see when looping stops
- Do useful stuff
- Get closer to termination

- How loops work in general
Four Things Needed In Any Loop

- Give starting values to one or more variables used in loop
- Test to see when looping stops
- One or more useful operations here
- Change something to move process closer to termination

Yet Another Loop Statement

```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` version

Yet Another Loop Statement

```java
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");
    }
}
```

- `for` version

Do Statement

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

- `do` version: not quite equivalent
  - termination test at end, so body executed at least once

Four Things Needed In Any Loop

- Give starting values to one or more variables used in loop
- Test to see when looping stops
- One or more useful operations here
- Change something to move process closer to termination

how loops work in general
**Do Statement**

- Body always executed at least once

- initialize
- do useful stuff
- test
- get closer to termination

order of four things can change, but need them all

**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
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);
        }
    }
}
```

- 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

```
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

```
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 need another loop to print numbers in row

```
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();
        }
    }
}
```

**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();
        }
    }
}
```
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();
        }
    }
}
```
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();
        }
    }
}
```

i
1
j
3

i
2
j
4

i
2
j
4

i
1
j
4

i
1
j
4

i
2
j
4

i
2
j
4
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();
        }
    }
}
```

```java
i
1
2
3
j
1
2
3
```

Nested Loops

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

i 2
j 3

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();
        }
    }
}

Nested Loops

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

i 2
j 3

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();
        }
    }
}
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();
        }
    }
}
```

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

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();
        }
    }
}
```

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

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();
        }
    }
}
```

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

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();
        }
    }
}
```

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

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();
        }
    }
}
```

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

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();
        }
    }
}
```

```
  1 2 3
  2 4 6
  3 6
```
Nested Loops

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

```
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();
        }
    }
}
```

```
ij32
12  32463_  
```
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();
        }
    }
}
```

| i | 3 |
| j | 4 |

Practice Problem

- Write program using loop to simulate flipping a coin one million times
  - keep track of how many times it's heads up and how many heads down
  - print results
  - Make version for each loop type
    - while, for, do

```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();
        }
    }
}
```