**News**

- Welcome back!
  - resume lectures, labs, tutorials, office hours
- Midterm and Assignment 1 returned
  - pick up after class if you don’t have yet
  - midterm solutions posted on WebCT
- Assignment 2 posted soon
  - probably later today

**Recap: Comparing Strings**

- Relational operator == is wrong way to compare

```java
String name1 = "Bubba";
String name2 = "Bubba";
System.out.println(name1 == name2);  // prints false
```

- `equals` method is right way to compare Strings

```java
String name1 = "Bubba";
String name2 = "Bubba";
System.out.println(name1.equals(name2));  // prints true
```

- why? diagrams will help

**Recap: Short-Circuting Evaluation**

- Java evaluates complex expressions left to right
  - **short-circuiting**: Java stops evaluating once value is clearly true or false
  - aka lazy evaluation

```java
if ((b > a) && (c == 10))
    System.out.println("when b<=a short-circuit");
else
    System.out.println("when b>a short-circuit");
```

- Corollary: avoid statements with side effects

```java
if ((b > a) || (c++))
    System.out.println("Danger Will Robinson!");
```
Recap: Conditional Syntax

if (boolean expression) statement
else if (boolean expression) statement
  optional: zero, one, or many
else statement
  optional
if, else are reserved words
parentheses mandatory
statement can be
  single line
  block of several lines enclosed in { }

Recap: Comparing Floats/Doubles

Relational operator for equality not safe for floating point comparison
if (.3 == 1.0/10.0 + 1.0/10.0 + 1.0/10.0)
System.out.println("Beware roundoff error");

Check if difference close to 0 instead
if (Math.abs(f1 - f2) < TOLERANCE)
System.out.println("Essentially equal.");

Recap: Comparing Characters

Safe to compare character types with relational operators
char c = 'a';
char d = 'b';
if (c == d)
  System.out.println("they match");

Recap: Switch Syntax

switch (expression) {
  case value:
    statements
    break;
  case value:
    statements
  default:
    statements
}

switch, case, break are reserved words
expression and value must be int or char
value cannot be variable
break important, or else control flow continues to next set
statements can be one line or several lines
default executed if no values match expression

Objectives

Practice with conditionals
Understand basic loops

public class NestTest3 {
  public static void main (String[] args) {
    respondToName("Flocinaucinihilipiliphication");
    respondToName("Supercalifragilisticexpialidocious");
    respondToName("Ambrose");
    respondToName("Kermit");
    respondToName("Miss Piggy!!!");
    respondToName("Spot");
    respondToName("me");
  }
  public static void respondToName(String name) {
    System.out.println("You're named "+ name);
    if (name.length() > 20) {
      System.out.println("Gosh, long name");
      System.out.println("Keeping typists busy...");
    } else if (name.length() > 30) {
      System.out.println("Over the top");
    } else if (name.length() < 10) {
      System.out.println("You're a frog");
    } else if (name.equals("Spot")) {
      System.out.println("You're spotted");
    } else if (name.length() < 3) {
      System.out.println("Concise!");
    }
  }
}


Repetition, Iteration, Loops
- Computers good at performing same task many times
- Loops allow repetitive operations in programs
  - aka iteration statements, repetition statements
- Loops handy in real life too

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.
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...

Washing Hair
- Lather
- Rinse
- Repeat

While Statement
```
while (boolean expression)    
    body
```
- Simplest form of loop in Java
- **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
If Versus While Statements

boolean expression
statement
true
false

How if statement works

These diagrams called flowcharts

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

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");
    }
}
```

boolean expression

■ How can loop boolean change from false to true?

■ These diagrams called flowcharts
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 body**

- **statement after while**
  - control flow resumes here when boolean is false

---

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");
    }
}
```

- **trace what happens when execute**

---

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**

---

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**
  - counter

---

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**
  - counter

---

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**
  - counter

---

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**
  - counter
  - Is counter <= limit? yes
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
"The square of 1 is 1" printed on monitor

Limit: 3  Counter: 2  Is counter <= limit? yes
"The square of 2 is 4" printed on monitor

Limit: 3  Counter: 3  Is counter <= limit? yes
"The square of 3 is 9" printed on monitor
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");
    }
}
```

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");
    }
}
```

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");
    }
}
```

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");
    }
}
```

Climbing Stairs Again

- 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.
- 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...
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");
    }
}

- change termination condition

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");
    }
}

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

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 + 2;
        }
        System.out.println("End of demonstration");
    }
}

- process gets closer to termination condition
  - but never satisfies condition, keeps going past it
Another while Example

```java
public class PrintFactorials {
    public static void main (String[] args) {
        int limit = 10;
        int counter = 1;
        int product = 1;
        while (counter <= limit) {
            System.out.println("The factorial of " + counter + " is " + product);
            counter = counter + 1;
            product = product * counter;
        }
        System.out.println("End of demonstration");
    }
}
```

Questions?

- accumulate product