Reading

- This week: Chapter 6 all (6.1-6.4)

News

- Midterm tonight: Tue Feb 7, 18:30 - 20:00
  - Geography 100 & 200
  - Seating by last name
    - A-Kim in 200
    - Kirtz-Z in 100
    - Id card face up on desk
    - Every other seat, sit where exam is laid out
  - Closed book/notes/calculator
  - Reminder: no labs or tutorials this week

Recap: Formal vs. Actual Parameters

- Formal parameter: in declaration of class
  ```java
  public class Point {  //...
    public void setPosition(int x, int y) {
      xCoord = x; yCoord = y;
    }
  }
  ```
- Actual parameter: passed in when method is called
  ```java
  public class PointTest {
    public static void main(String [] args) {
      //...
      tester.setPosition(3,4);
    }
  }
  ```

Recap: Scope

- Variable scope: block of code it's declared in
  - block of code is defined by braces {}
- Class scope: accessible to any class member
  - fields accessed by all class methods
- Local scope: method parameters and variables declared within method body

Recap: Shorthand Operators

- Java shorthand
  - count++; // same as count = count + 1;
  - count--; // same as count = count - 1;
  - note no whitespace between variable name and operator
- Similar shorthand for assignment
  - tigers += 5; // like tigers=tigers+5;
  - lions -= 3; // like lions=lions-3;
  - bunnies *= 2; // like bunnies=bunnies*2;
  - dinos /= 100; // like dinos=dinos/100;
Recap: Data Conversion

- Math in Java: it depends!
  
  ```java
  int a = 1 / 3;         // a is 0
double b = 1 / 3;      // b is 0.0
int c = 1.0 / 3.0;     // Java’s not happy
double d = 1.0 / 3.0;  // d is 0.333333333
  ```

Recap: Automatic Conversion

- Done implicitly if widening
- Assignment conversion: converted because value of one type assigned to variable of other type
  
  ```java
double b = 1 / 3;
  ```
- Promotion: converted because expression contains mixed types
  
  ```java
  int hours_worked = 40;
double pay_rate = 5.25;
double total_pay = hours_worked * pay_rate;
  ```

Recap: Static Variables

- Static variable shared among all instances of class
  
  - "belongs" to class, not instances
  - only one copy of static variable for all objects of class
  - thus changing value of static variable in one object changes it for all others objects too!

  Memory space for a static variable established first time containing class is referenced in program

Recap: Static Methods

- Static method "belongs" to the class itself
  
  - not to objects that are instances of class
  - aka class method
- Do not have to instantiate object of class in order to invoke static method of that class
- Can use class name instead of object name to invoke static method

Recap: Static Example

```java
public class Giraffe {

  private static int numGiraffes;
  private double neckLength;
  public Giraffe(double neckLength) {
    this.neckLength = neckLength;
    numGiraffes++;
  }

  public void sayHowTall() {
    System.out.println("Neck is "+ neckLength);
  }

  public static int getGiraffeCount() {
    return numGiraffes;
  }
}
```
Static Example

```java
public class Giraffe {
    private static int numGiraffes;
    private double neckLength;
    public Giraffe(double neckLength) {
        this.neckLength = neckLength;
        numGiraffes++;
    }
    public void sayHowTall() {
        System.out.println("Neck is "+neckLength);
    }
    public return numGiraffes;
}static int getGiraffeCount() {
    numGiraffes++;
    return numGiraffes;
}
```

> Note that Giraffe is class name, not object name!

Objectives

- Understand how static methods work
- Understand how to use conditionals
- Understand how boolean operators work
**Conditional Statement**

- **Boolean expression**: test that returns true or false
- **Conditional statement**: choose which statement will be executed next based on boolean expression
- **Example**

```java
if (age < 20)
    System.out.println("Really, you look like you are " + (age + 5) + ".");
```

**Conditional Example**

```java
import java.util.Scanner;
public class Feelgood
{
    public static void main (String[] args)
    {
        int age;
        Scanner scan = new Scanner (System.in);
        System.out.println ("Enter your age: ");
        age = scan.nextInt();
        if (age < 20)
            System.out.println("Really, you look like you 
            + " + (age + 5) + ".");
        if (age >= 20)
            System.out.println ("You don't look a day over " + (age - 10) + "!");
    }
}
```

**Conditional In Depth**

- Within method, statements usually executed top to bottom
  - one after the other
- Change control flow with conditional statement

```java
if (age < 20)
    System.out.println("Really, you look like you are " + (age + 5) + ".");
```

- Choice hinges on evaluation of boolean operator

**Boolean Expressions**

- **Boolean expression**: test which returns either true or false when evaluated
  - aka conditional
- **Consists of operands and operators**, like arithmetic expression
  - but operators only return true or false when applied to operands
- **Two different kinds of operators**
  - relational
    - sometime split into relational and equality
  - logical
Relational Operators

- Tests two values (operands)

- Operators
  - `==` equal
    - returns true if they are equal, false otherwise
  - `!=` not equal
    - returns true if they are not equal, false otherwise
  - `<` less than
  - `<=` less than or equal to
  - `>` greater than
  - `>=` greater than or equal to

Equality Example

```java
int a = 3;
int b = 6;
int c = 10;
if (a == b)
  System.out.println("these two values are equal");
if ((b - a) == a)
  System.out.println("b is the same as a");
if (a != b)
  System.out.println("nope!");
```

Logical Operators

- Way to combine results from relational operators into single test
- AND, OR, and NOT
  - in terms from math or philosophy class

- Operators
  - `&&` logical AND
  - `||` logical OR
  - `!` logical NOT

Logical AND

- Logical AND of values a and b evaluates to
  - true if both a and b are true
  - false otherwise

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>a &amp;&amp; b</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>

Logical OR

- Logical OR of values a and b evaluates to
  - true if either a or b are true
  - true if both are true
  - false otherwise

  | a     | b     | a || b |
  |-------|-------|--------|
  | false | false | false  |
  | false | true  | true   |
  | true  | false | true   |
  | true  | true  | true   |

Logical NOT

- Logical NOT of value a evaluates to
  - true if a is false
  - false if a is true

<table>
<thead>
<tr>
<th>a</th>
<th>! a</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
</tr>
</tbody>
</table>
int a = 3;
int b = 6;
int c = 10;

if ((b > a) && (c == 10))
    System.out.println("this should print");

if (!(b > a))
    System.out.println("this should not print");

if !(b > a)
    System.out.println("what happened?");

Logical Operator Examples

- is (! (b > a)) the same as
  - (a > b)
  - (a >= b)
  - (b < a)

Questions?