Comparing Strings
- These values tested for equality with `name1.equals(name2)`.
- Contents of objects are same, so true.

Recap: Static Methods
- Static methods do not operate in context of particular object.
- cannot reference instance variables because they exist only in an instance of a class.
- Compiler will give error if static method attempts to use non-static variable.
- Static method can reference static variables.
- Because static variables exist independent of specific objects.

Recap: Static Methods in java.math
- Java provides you with many pre-existing static methods.
- Package java.lang.Math is part of basic Java environment.
- You can use static methods provided by Math class.
- Examples:
  - `Math.abs(34)`
  - `Math.sin(90)`
  - `Math.tan(Math.toRadians(90))`
  - `Math.max(54,70)`
  - `Math.round(3.14159)`

Recap: Relational Operators
- Tests two values (operands).
- Operators:
  - `==` equal
  - `!` not equal
  - `<` less than
  - `<=` less than or equal to
  - `>` greater than
  - `>=` greater than or equal to

Recap: Logical Operators
- Way to combine results from relational operators into single test.
- Syntax:
  - `x && y` and
  - `x && y`
- Logical operator examples:
  - `!(b > a)` is the same as
  - `(a > b)`
  - `(a >= b)`
  - `(b < a)`

Objectives
- Understand how to compare objects and primitive data types.
- Understand syntax to use for conditionals and switch statements.

Recap: Relational Operators
- Test if two values are equal.
- Returns true if values are equal, false otherwise.
- Do not confuse this with `x = y`.
- `>` returns true if `x` is greater than `y`.
- `<` returns true if `x` is less than `y`.
- `>=` returns true if `x` is greater than or equal to `y`.
- `<=` returns true if `x` is less than or equal to `y`.

Comparing Strings
- How do we test for equality between Strings?
- Reminder:
  - Strings are sequences of alphanumeric characters.
  - Create with constructor.
  - String firstname = new String("Donald");
  - or with shortcut
  - String lastname = "Duck";
- Strings are objects, not primitive types.

Comparing Strings
- Consider another expression.
  - `if ((b > a) && (c == 10))`.
- Java evaluates left to right.
  - If `(b > a)` is false, does value of `(c == 10)` matter?
  - Not result of `&&` must be false because one operand already evaluated to false.
  - Short-circuiting: Java does not evaluate `aka lazy evaluation`.

Short-Circuiting Evaluation
- Consider different expression.
  - `if ((b > a) || (d == 10))`.
- Java evaluates left to right.
  - If `(b > a)` is true, does value of `(d == 10)` matter?
  - Not result of `||` must be true since one operand already evaluated to true.

If Syntax
- Syntax:
  - `if` followed by boolean expression enclosed in parentheses.
  - `if` followed by statement.
- Results:
  - If boolean evaluates to true, statement is executed.
  - Otherwise statement is skipped, execution continues with statement immediately following if statement.
If-Else Syntax
- If statement may include optional else clause
  - reserved word else
  - followed by another statement
  - if (x == y) { System.out.println("x equals y"); } else { System.out.println("x is not equal to y"); }
- Results
  - If boolean evaluates to true, first statement is executed
  - otherwise (if boolean evaluates to false), statement following else is executed

Block Statements
- Often want to do many actions, not just one, based on condition
  - Replace single statement with many statements surrounded by curly braces
  - if (x == y) {
      System.out.println("x equals y");
      System.out.println("I'm happy");
    } else {
      System.out.println("x is not equal to y");
      System.out.println("I'm depressed");
      System.out.println("See you next year");
    }

Block Statements
- What if we leave out block in else clause?
  - if (x == y) {
      System.out.println("x equals y");
      System.out.println("I'm happy");
    } else {
      System.out.println("x is not equal to y");
      System.out.println("I'm depressed");
      System.out.println("See you next year");
    }

Nested If Syntax
- Statements within if-else statements can themselves be if-else statements

Switch Syntax
- Use switch statement to get program to follow one of several different paths based on single value
  - switch (finalMark) {
      case 4: System.out.println("You get a C"); break;
      case 3: System.out.println("You get a B"); break;
      case 2: System.out.println("You get a A"); break;
      default: System.out.println("See you next year");
    }

Comparing Characters
- You can compare character types with relational operators
  - 'a' < 'b'
  - 'a' == 'b'
  - 'a' < 'A'
- Remember, cannot compare Strings with relational operators
  - Or any other objects!
  - must use methods like equals

Comparing Floating Point Numbers
- Is 0.3 the same thing as 1.0/10.0 + 1.0/10.0 + 1.0/10.0 ???
  - No - very close, but not exactly what you expect
  - 0.30000000000000004
  - Beware! Write tests for “darn near equal” like:
    - if (Math.abs(f1 - f2) < TOLERANCE) System.out.println("Essentially equal.");
      where TOLERANCE is small number appropriate to problem like 0.00000001

Switch Syntax
- Expression should be int, char (or enumerated type)
  
```
switch (finalMark) {
    case 4:
      System.out.println("You get a C"); break;
    case 3:
      System.out.println("You get a B"); break;
    case 2:
      System.out.println("You get a A"); break;
    default:
      System.out.println("See you next year");
}
```

Switch Syntax
- Case values cannot be variables
  
```
switch (finalMark) {
    case 4:
      System.out.println("You get a C"); break;
    case 3:
      System.out.println("You get a B"); break;
    case 2:
      System.out.println("You get a A"); break;
    default:
      System.out.println("See you next year");
}
```

Switch Syntax
- Break statements really important
  
```
switch (finalMark) {
    case 4:
      System.out.println("You get a C"); break;
    case 3:
      System.out.println("You get a B"); break;
    case 2:
      System.out.println("You get a A"); break;
    default:
      System.out.println("See you next year");
}
```

Switch Syntax
- Default statement optional, but very good idea
  
```
switch (finalMark) {
    case 4:
      System.out.println("You get a C"); break;
    case 3:
      System.out.println("You get a B"); break;
    case 2:
      System.out.println("You get a A"); break;
    default:
      System.out.println("See you next year");
}
```

Nested If Syntax
- Multiple else statements also legal

Comparing Floating Point Numbers
- Is 0.3 the same thing as 1.0/10.0 + 1.0/10.0 + 1.0/10.0 ???
  - No - very close, but not exactly what you expect
  - 0.30000000000000004
  - Beware! Write tests for “darn near equal” like:
    - if (Math.abs(f1 - f2) < TOLERANCE) System.out.println("Essentially equal.");
      where TOLERANCE is small number appropriate to problem like 0.00000001

Switch Syntax
- Expression should be int, char (or enumerated type)
  
```
switch (finalMark) {
    case 4:
      System.out.println("You get a C"); break;
    case 3:
      System.out.println("You get a B"); break;
    case 2:
      System.out.println("You get a A"); break;
    default:
      System.out.println("See you next year");
}
```

Switch Syntax
- Case values cannot be variables
  
```
switch (finalMark) {
    case 4:
      System.out.println("You get a C"); break;
    case 3:
      System.out.println("You get a B"); break;
    case 2:
      System.out.println("You get a A"); break;
    default:
      System.out.println("See you next year");
}
```

Switch Syntax
- Break statements really important
  
```
switch (finalMark) {
    case 4:
      System.out.println("You get a C"); break;
    case 3:
      System.out.println("You get a B"); break;
    case 2:
      System.out.println("You get a A"); break;
    default:
      System.out.println("See you next year");
}
```

Switch Syntax
- Default statement optional, but very good idea
  
```
switch (finalMark) {
    case 4:
      System.out.println("You get a C"); break;
    case 3:
      System.out.println("You get a B"); break;
    case 2:
      System.out.println("You get a A"); break;
    default:
      System.out.println("See you next year");
}
```