

Tamara Munzner

#### Conditionals II

Lecture 16. Wed Feb 10 2010

borrowing from slides by Kurt Eiselt

http://www.cs.ubc.ca/~tmm/courses/111-10

#### **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 nonstatic variable
- Static method can reference static variables
- because static variables exist independent of specific obiects

# 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.sqrt(36) > Math.random() 6.0 0.7843919693319797 > Math.sin(90) > Math.random() 0.8939966636005579 0.4253202368928023 > Math.sin(Math.toRadians(90)) > Math.pow(2,3) > Math.max(54,70) 8.0 > Math.pow(3,2) > Math.round(3.14159) 9.0 > Math.log(1000) 6.907755278982137 > Math.log10(1000)

3 0

#### **Recap: Conditional Statement**

- Conditional statement: choose which statement will be executed next based on boolean expression
- changes control flow
- Example
- if (age < 20)
   System.out.println("Really, you look like you are "</pre> + (age + 5) + ".");

#### **Recap: Boolean Expressions**

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

13

#### **Recap: Relational Operators**

- Tests two values (operands)
- Operators
  - returns true if they are equal, false otherwise
  - note: do not confuse this with = != not equal

  - returns true if they are not equal, false otherwise < less than</p>
  - <= less than or equal to</p> > greater than

  - >= greater than or equal to

#### **Recap: 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

# (b < a)</p>

**Logical Operator Examples** 

■ is (! (b > a)) the same as

(a > b)

(a >= b)

10

#### **Objectives**

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

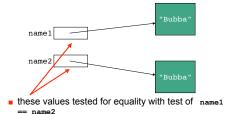
# **Comparing Strings**

- How do we test for equality between Strings?
- - 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

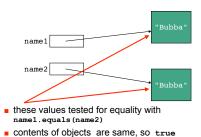
- Relational operator == is wrong way to compare
- String name2 = "Bubba": System.out.println(name1 == name2); // prints false
- Equals method is right way to compare Strings
- String name1 = "Bubba"; String name2 = "Bubba"; System.out.println(name1.equals(name2)); // prints true
- why? diagrams will help

#### **Comparing Strings**



- two different pointers (references), so false

#### **Comparing Strings**



# **Short-Circuting Evaluation**

- Consider again expression
- if ((b > a) && (c == 10)) System.out.println("this should print");
- Java evaluates left to right
  - if (b>a) is false, does value of (c == 10) matter? • no! result of && must be false since one operand already evaluated to false
  - short-circuiting: Java does not evaluate
  - aka lazy evaluation

# **Short-Circuting Evaluation**

- Consider different expression
- if ((b > a) || (d == 10)) System.out.println("this should print");
- Java evaluates left to right
  - if (b>a) is true, does value of (c == 10) matter?
- no! result of || must be true since one operand already evaluated to true

#### If Syntax

- Syntax
  - reserved word if
  - followed by boolean expression enclosed in parentheses
  - followed by statement
- if (x == y)
   System.out.println("x equals y! ");
- - if boolean evaluates to true, statement is executed
  - otherwise statement is skipped, execution continues with statement immediately following if statement

14

15

11

#### **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!");
- if boolean evaluates to true, first statement is executed
- otherwise (if boolean evalutes to false), statement following

# **Block Statements**

**Nested If Syntax** 

public class NestTest2

- Often want to do many actions, not just one, based on
- 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");
      System.out.println("x is not equal to y");
System.out.println("I'm depressed");
System.out.println("How about you?");
```

18

#### **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("How about you?");
```

### **Nested If Syntax**

public class NestTest

Statements within if-else statements can themselves be if-

```
public static void main (String[] args)
         System.out.println("all three values the same");
         System.out.println("y is not equal to z");
 else System.out.println("x is not equal to y");
```

**Comparing Floating Point Numbers** 

#### **Nested If Syntax**

■ Multiple else statements also legal

```
// statements
   // statements
else if ( Boolean expression 3 )
  // statements
  // statements
```

```
public static void main (String[] args)
 int x = 1; int y = 3; int z = 2;
     System.out.println("all three values the same");
 else if ((x == y) && (y != z))
     System.out.println("x is not equal to y");
```

■ Rewriting NestTest using multiple else statements

21

17

# **Comparing Floating Point Numbers**

- Is 0.3 the same thing as 1.0/10.0 + 1.0/10.0 + 1.0/10.0 ???
- Let's try it out...

22

Is 0.3 the same thing as 1.0/10.0 + 1.0/10.0 + 1.0/10.0 ???

19

23

27

- No very close, but not exactly what you expect . 0.300000000000000004
- 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

#### **Comparing Characters**

You can compare character types with relational operators

```
'a' < 'b'
```

- Remember, cannot compare Strings with relational operators
  - or any other objects!
- must use methods like equals

### **Switch Syntax**

Use switch statement to get program to follow one of several different paths based on single value

```
switch (finalMark)
       System.out.println("You get an A");
       break;
       System.out.println("You get a B");
       break;
       System.out.println("You get a C");
       break;
   default:
       System.out.println("See you next year");
```

25

29

### **Switch Syntax**

 Expression should be int, char (or enumerated type)

```
switch (f
   case 4
       System.out.println("You get an A");
   case 3:
       System.out.println("You get a B");
       break;
   case 2:
       System.out.println("You get a C");
       hreak.
   default:
       System.out.println("See you next year");
```

26

#### Case values cannot be variables

```
switch (finalMark)
   case f:
       System.out.println("You get an A");
       break;
   case 3:
       System.out.println("You get a B");
       break;
   case 2:
       System.out.println("You get a C");
   default:
       System.out.println("See you next year");
```

#### **Switch Syntax**

■ Default statement optional, but very good idea

```
switch (finalMark)
       System.out.println("You get an A");
   case 3:
       System.out.println("You get a B");
   case 2:
       System.out.println("You get a C");
       break;
       System.out.println("See you next year");
```

# **Switch Syntax**

Break statements really important

```
switch (finalMark)
       System.out.println("You get an A");
        break;
       System.out.println("You get a B");
        System.out.println("You get a C");
        System.out.println("See you next year");
```

30

# **Switch Syntax**