Midterm Problems
Boolean Expressions
Lecture 14

Borrowing from slides by Alan Hu, Kurt Eiselt, Paul Carter, and Tamara Munzner

Reading Assignments
■ Reading for this week: conditions
  ■ Edition 2: Ch. 6.1-6.4
  ■ Edition 3: Ch. 5.1-5.4

■ Reading for next week: looping/iterations
  ■ Edition 2: Ch. 6.1-6.5
  ■ Edition 3: Ch. 7.1-7.5

Objectives
■ Discuss midterm problems
■ Understand equivalence between complex logical expressions

Recap: javadoc Method Example

```java
/**
 * Sets the die shape, thus the range of values it can roll.
 * @param numSides the number of sides of the die
 */
public void setSides(int numSides) {
    sides = numSides;
}

/**
 * Gets the number of sides of the die.
 * @return the number of sides of the die
 */
public int getSides() {
    return sides;
}
```

Recap: javadoc Class Example

```java
/** Die: simulate rolling a die
 * @author: CPSC 111, Section 206, Spring 05-06
 * @version: Jan 31, 2006
 * This is the final Die code. We started on Jan 24,
 * tested and improved in on Jan 26, and did a final
 * cleanup pass on Jan 31.
 */
```

Recap: Java’s if statement

■ Syntax:
  ```java
  if ( condition ) statement;
  ```
■ Examples:
  ```java
  if (tax < 0) tax = 0;
  ```
  ```java
  if (age < 19)
      System.out.println("Sorry, you may not buy alcohol.");
  ```
Recap: Age 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 " + "are " + (age + 5) + ".");
            System.out.println("You don't look a day over " + (age - 10) + "!");
        } else {
            System.out.println("You don't look a day over " + (age - 10) + "!");
        }
    }
}
```

Recap: Java’s if statement with else

- Syntax:
  ```java
  if ( condition ) statement;
  else statement;
  ```
- Example:
  ```java
  if (age < 19) {
    System.out.println("You may not buy alcohol.");
  } else {
    System.out.println("Buy beer!");
  }
  System.out.println("This lecture sponsored by Molson");
  ```

Recap: Conditional Statement

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

Recap: Boolean Expressions

- **Boolean expression**: test which returns either true or false when evaluated
- 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
  - logical
Recap: Relational Operators
- Compares two values (operands)
- Operators (See Appendix F.)
  - `==` 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

Recap: 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("that was a silly example");
if (a != b)
    System.out.println("nope!");
```
- Note we can use arithmetic operator inside boolean expression

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

Recap: Logical AND
- Logical AND of values a and b evaluates to
  - true if both a and b are true
  - false otherwise
  - "If you are smart and beautiful, then I want to meet you”
```
a    b    a && b
false false false
false true false
true false false
true true true
```

Recap: 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
  - "If you are mean or annoying, please go away."
```
a    b    a || b
false false false
false true true
true false true
true true true
```

Recap: Logical NOT
- Logical NOT of value a evaluates to
  - true if a is false
  - false if a is true
```
a    ! a
false true
true false
```
Recap: Logical Operator Examples

- is !(b > a)) the same as
  - (a > b) NO!
  - (a >= b) YES!
  - (b < a) NO! (Same as (a > b))

Precedence and Associativity of Logical Operators

- Logical operators have different precedence:
  - ! has highest precedence
  - && has 2nd highest precedence
  - || has lowest precedence
- All logical operators are left associative, i.e.
  - a && b && c is identical to (a && b) && c

Equivalence of Complex Boolean Expressions

- Using boolean tables, we can transform complex expressions, and possibly simplify them
- For example:
  - Can we simplify !(a && b) ?

Answer:
- Yes, equivalent to (a || b)

Equivalence of Complex Boolean Expressions

- In general:
  - !(1a) is identical to a
  - !(a && b) is identical to !(a) || !(b)
  - !(a || b) is identical to !(a) && !(b)
  - a && (b || c) is identical to (a && b) || (a && c)
  - a || (b && c) is identical to (a || b) && (a || c)
  - a || (b && c) but is not identical to (a || b) && (a || c)

Equivalence of Complex Boolean Expressions

- Therefore:
  - ! is a bit like the unary –
  - && is a bit like *
  - || is a bit like +