Consider the following statements. What are the values of these statements?

- My name is Amy and I am a student.
- It is 10:00am, or we are in class right now.
- I am not a criminal.

The values are all either true or false

Boolean algebra: mathematical expressions in which the variables and the answer take one of two values: \{true, false\}

Boolean operators

- &&: logical AND
  - p && q
- ||: logical OR
  - p || q
- !: logical NOT
  - !p

Boolean operators (cont.)

- p&&q: True only if both p and q are true
- p||q: True if either p or q is true (or both)
- !p: True only if p is false
**Truth table**

- A list of all possible inputs for a boolean expression, along with the output for each combination of inputs
- Example:

| p | q | p&&q | p||q | \!p | \!q |
|---|---|------|------|-----|-----|
| T | T | T    | T    | F   | F   |
| T | F | T    | F    | F   | T   |
| F | T | F    | T    | T   | F   |
| F | F | F    | F    | T   | T   |

**Comparison operators**

- >  greater than
- >= greater than or equal to
- <  less than
- <= less than or equal to
- == equal to
- != not equal to

**Java expressions**

- Typically, we’ll use comparison operators and boolean operators together
- Examples:
  - If age is greater than 30 and less than 65, print “You are an old geezer”
  - If a is less than zero and b is less than zero and c is less than zero, print out “All three numbers are negative”
  - If y is less than or equal to 40 or y is not a multiple of 2, add 50 to y.

**Java expressions (cont.)**

- age is greater than 30 and less than 65:
  - `(age > 30) && (age < 65)`
- a, b, and c are less than 0:
  - `(a < 0) && (b < 0) && (c < 0)`
- y is less than or equal to 40 or is not a multiple of 2:
  - `(y <= 40) || !(y%2 = 0)`
Conditional statements

- Sometimes we want to be able to alter the “flow” of a program, based on certain conditions or certain input values:
  - if user selects a color from the color menu, change the foreground color to that color
  - if the account balance is below $500, charge a penalty
  - if the input is invalid, allow the user to try again
  - if a 6 is rolled on the dice, the player loses a turn.

if statement

- Format:
  ```java
  if (<condition>) {
    <statement(s)>;
  }
  ```
- Example:
  ```java
  int a = 5, b = 6;
  if ((a+b)%3) != 0) {
      System.out.println("The sum of a and b is not divisible by 3");
  }
  ```

if-else statement

- Format:
  ```java
  if (<condition1>) {
    <statement(s)>;
  } else if (<condition2>) {
    <statement(s)>;
  }...
  else {
    <statement(s)>;
  }
  ```
Notes

- `<condition>` is a boolean expression (a statement that is either true or false)
  - typically, this will involve a comparison
- `<statement(s)>` means any number of Java statements that should be executed if this condition is true

Examples

```java
String answer = JOptionPane.showInputDialog(null, "What is your age?");
int age = Integer.parseInt(answer);
if (age < 0) {
    JOptionPane.showMessageDialog(null, "Invalid age entered!");
} else if (age < 30) {
    JOptionPane.showMessageDialog(null, "Hello, spring chicken!");
} else if (age < 60) {
    JOptionPane.showMessageDialog(null, "You're not as young as you think!";)
} else {
    JOptionPane.showMessageDialog(null, "Looking good for your age!");
}
```

More complex conditional statements

- Consider the following statements:
  - If $x$ is greater than 50 and less than 100, then subtract 50 from $x$
  - Else, if $x$ is less than or equal to 50 and greater than 10, then subtract 10 from $x$
  - Else, add 25 to $x$
- Q: What does this look like in Java code?

```java
if (x>50 && x<100) {
    x = x - 50;
}
else if (x<=50 && x>10) {
    x = x - 10;
}
else {
    x = x + 25;
}
```
Notes

- else if and else statements are optional
- be careful not to confuse equality operator (==) with assignment operator (=)!
- if more than one condition is true in an if-else statement, only the first one is executed
- the program continues on with its normal operation after the if-else block

“Short-circuit” evaluation

- In a complex condition, only as much of the condition is evaluated as is needed. In other words, if the answer is clear after evaluating the first part of the expression, then none of the remaining parts of the expression are evaluated.
- Examples: \( x=5, y=8 \)
  - \( (x<4 || y>9) \): both sides of the || must be evaluated
  - \( (x<4 && y>5) \): only the left hand side is evaluated; since it's false and the operator is &&, the expression is obviously false
  - \( (x>4 || ((x%2)==0) || ((y%2)==0)) \): only the leftmost side is evaluated; since it's true and the operators are ||, the expression is obviously true

Summary

- conditional statements allow us to alter the execution of a program based on user input, variable values, etc.
- boolean expressions are mathematical expressions that evaluate to either true or false
- conditional statements typically involve both comparison operators and boolean operators
- next up: the switch statement