Java Software Solutions

Chapter 5 - Conditionals and Loops

Boolean Expressions

Conditional Statements

- Allows us to choose which statement will be executed next
- Based on a Boolean Expression
  - A statement that reduces to the value true or false
- Conditional Statements:
  - If
  - If-else
  - switch

Equality and Relational Operators

Operator	Meaning
= =	equal to
! =	not equal to
<	less than
< =	less than or equal to
>	greater than
> =	greater than or equal to

Logical Operators

Operator	Description	Example	Result
!	logical NOT	! a	true if a is false and false if a is true
&&	logical AND	a && b	true if a and b are both true and false otherwise
||	logical OR	a || b	true if a or b or both are true and false otherwise

The if Statement

Blocks and Indentation

- Remember, white space has no special meaning in Java
- While it's best practice to indent different blocks, indenting a line does not make it part of a different execution block
- Blocks are defined by {}

if

```
if (x > y){
    //do something
}
```

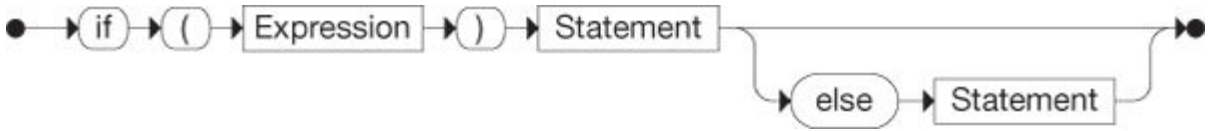
- If you only have to run a single line, then the {} are optional
  - That said, it's best to use them if you're not sure, as you may need to modify your code later

if else

```
if (x > y){
        //do something
} else {
        //do a different thing
}
```

- Like the if statement by itself, the {} are optional at any point if you only need 1 line

If-Else Visualized



Nesting if-else

- Extremely common
- Just put one if-else inside another
- Let the {} be your guide on where one starts and the other begins

```
if(x > y){
        if(x < z){
                System.out.println(“x is greater than y but less than z”);
        }
}
```

Comparing Data

Primitives vs. Objects

- Primitive values (int, float, bool, double, char) are compared by value
  - int x = 5, y =5;
  - x == y //returns true
- Objects are compared by their reference
  - Dog dog1 = new Dog(“Jerry”);
  - Dog dog2 = new Dog(“Jerry”);
  - dog1 == dog2 //returns false

Weird String Stuff

- Strings are objects in Java, but Java tries to hide that from you as much as possible
- Every string you define will create a new String object in the background
  - However, if the string is the same as an existing one, Java will just reference the existing one instead of creating a new one
- It’s best practice to use .equals() and .compareTo() with strings

Weird Float Stuff

- Floating point values are not accurate
- Even if two different calculations should return the same values, if they're floats, it's not guaranteed

```
if (Math.abs(f1 - f2) < TOLERANCE)
    System.out.println("Essentially equal");
```

Object Comparison

- All objects have a .equals() method
  - Use this to test equality on objects instead of ==
  - Each object defines what .equals() means for it
- Objects may have a compareTo method
  - Returns a negative integer if the “comparer” is less than the “compared”
  - Returns 0 if both are equal
  - Returns a positive integer if the “comparer” is greater than the “compared”

comparer.compareTo(compared);

The while Statement

The While Statement

```
while (total > max)
{
   total = total / 2;
   System.out.println("Current total: " + total);
}
```

break and continue

- break - special statement that exits a code block
- continue - “breaks” only one iteration of a loop, but keeps control flow inside of the loop

They’re neat tricks, but should generally be avoided in your code

Iterators

The Iterable Interface

- We won't go into too much detail yet…
- Guarantees that some objects have certain methods:
  - hasNext() //returns a boolean
  - next() //returns next… thing

Reading Files with Scanner

- Scanner is flexible and can work with different inputs

Scanner fileScanner = new Scanner(new File(“stuff.txt”));

- Since Scanner implements Iterable, we can use hasNext() and nextLine() to iterate through all the lines in a file

The ArrayList Class

ArrayList

- Remember lists in Python?
  - It’s like that, but with more rules
- Allows you to store and iterate over multiple values

```
ArrayList<String> myList = new ArrayList<String>();
myList.add(“Yo”);
myList.add(“asdf”);
myList.get(0); //returns “Yo”, the 0th item in the list
```

JavaFX - Determining Event Sources

Let’s play with ActionEvents to decouple our logic