Lecture 5: If Statements

Last time:
1. Variables and types
2. Expressions in Java
3. User input with Scanner objects

Today:
1. Finish Scanner
2. if statements
Objects

- From Example 5:
  
  ```java
  Scanner sc = new Scanner(System.in);
  ```
  
  - `sc` is a variable
  - Its type is ...Scanner?

- What's going on?
  
  - `Scanner` is a class defined in `java.util.Scanner`
  - `System.in` is a predefined `object` for keyboard input
  - `new Scanner(System.in)` creates a new `object` in the `Scanner` class and assigns it to `sc`

- Object?
  
  - A bundle of data (`instance variables`) and operations (`methods`)
  - A class defines both instance variables and methods for objects
  - A class is also a type for objects
  - `new` creates new objects in the given class

- We will learn (much) more about objects later
Control Flow and Conditionals

- **Control flow**: the order in which statements are executed
  - General rule: top to bottom
- **Conditional statements** permit control flow to be dependent on (true/false) conditions
  - if
  - if-else
The if Statement

- Form:
  \[
  \text{if ( <boolean-expression> ) <statement>}
  \]

- Example:
  \[
  \text{if (inchesOfSnow > 7) }
  \]
  \[
  \text{System.out.println( "Go home" );}
  \]

- The println statement is executed only if the variable “inchesOfSnow” is greater than 7
- Otherwise, it is skipped
Java and White Space

- You can add:
  - carriage returns
  - spaces
  - tabs

  wherever you want in Java

- Properly used, this makes your program easier to read and understand
Example 6

```java
public class Example6 {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter an integer: ");
        int i = sc.nextInt();

        if (i < 0){
            System.out.println("That was a negative number! ");
        }
        System.out.println("The number was "+i);
    }

}
The if-else Statement

- **Form:**
  
```java
if ( <condition> ) <statement 1>;
else <statement 2>;
```

- **Example:**
  
```java
if (inchesOfSnow > 7) {
    System.out.println("Go home");
} else {
    System.out.println("Go to school");
}
```

- If “inchesOfSnow” > 7, the first println statement is executed and the second is skipped.
- Otherwise (i.e. inchesOfSnow ≤ 7), the first println statement is skipped and the second is executed.
Indentation Convention for if-else

The if-else class of statements should have the following form:

- if (condition) {
    statements;
}
- if (condition) {
    statements;
} else {
    statements;
}
- if (condition) {
    statements;
} else if (condition) {
    statements;
} else {
    statements;
}
See Sun Code Conventions on Resource Page!

Example 7

```java
public class Example7 {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter an integer: ");
        int i = sc.nextInt();

        if (i < 0) {
            System.out.println("That was a negative number!");
        } else {
            System.out.println("That was a non-negative number!");
        }
        System.out.println("The number was: " + i);
    }
}
```
Blocks

- What happens?
  ```
  if (i > 10)
    i = 10;
    saturate = true;
  else
    i = i+1;
  ```
- Desired: both `i`, `saturate` are set if `i > 10`
- Actual: syntax error!
  - Only one statement can be associated with `if`
  - The `saturate` assignment statement is not part of the `if`
  - The `else` has no `if` to belong to!
- Blocks solve this problem
What Blocks Are

- Blocks are sequences of statements “glued together” into one
- Form:
  ```
  {
    <statement 1>;
    <statement 2>;
    ...
  }
  ```
- Example revisited
  ```
  if (i > 10) {
    i = 10;
    saturate = true;
  } else {
    i = i+1;
  }
  ```
Indentation Conventions for Blocks

- Either
  
  ```java
  if (...) {
    statement 1;
    statement 2;
    ...
  }
  ```

- Or
  
  ```java
  if (...) {
    statement 1;
    statement 2;
    ...
  }
  ```

This is what we will use
(Sun code convention)
Example 8

public class Example8 {
  
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter an integer: ");
        int i = sc.nextInt();

        if (i < 0) {
            System.out.println("That was a negative number! ");
            System.out.println("I prefer positive ones, so I'll fix it...");
            i = -i;
        } else {
            System.out.println("That was a positive number! ");
            System.out.println("That makes me happy.");
        }
        System.out.println("The number was " + i);
    }
}
Logical Operators

Used for forming more complex conditions.

- “and”  
  \[
  \text{if} \ ( \text{temp} \geq 97 \ \&\& \ \text{temp} \leq 99) \ {\}
  \begin{array}{l}
  \text{System.out.println(”Patient is healthy”);} \\
  \end{array}
  \]

- “or”   
  \[
  \text{if} \ ( \text{months} \geq 3 \ \| \ \text{miles} \geq 3000) \ {\}
  \begin{array}{l}
  \text{System.out.println(”Change your oil”);} \\
  \end{array}
  \]

- “not”  
  \[
  \text{if} \ ( \text{! phone.equals(”301-555-1212”)}) {\}
  \begin{array}{l}
  \text{System.out.println(”Sorry, wrong number”);} \\
  \end{array}
  \]
Example 9

```java
public class Example9 {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter an integer from 1 to 10: ");
        int i = sc.nextInt();

        if (i >= 1 && i <= 10) {
            System.out.println("Good job!");
        } else {
            System.out.println("You didn't follow instructions!");
        }
    }
}
```
Statement Constructors and Nesting

- if, if-else, {...} are *statement constructors*
  - They take statement(s) and convert them into a new statement
  - Example:

    ```java
    if (i >= 1 && i <= 10) {
        System.out.println("Good job!");
    } else {
        System.out.println("Oops!");
    }
    ```

  - Two “sub-statements” come in
  - A single big statement (if … else …) comes out

- Implications: if statements, etc. can also appear inside (“be nested within”) one another
Example 10

```java
public class Example10 {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter an integer: ");
        int i = sc.nextInt();

        if (i < 0) {
            System.out.println("I don't like negative numbers!");
            if (i < -100) {
                System.out.println("Also... that one is REALLY negative!");
            }
        } else {
            if (i < 100) {
                System.out.println("That's a small one.");
            } else {
                System.out.println("That's a nice big number!");
            }
        }
    }
}
```
Java, Eclipse and Uninitialized Variables

• Eclipse will complain if you try to use an uninitialized variable:
  ```java
  int i;
  System.out.println ("i is " + i);
  ```
  What is value of i?

• This feature interacts strangely with if/else statements sometimes

• Good programming practice: always initialize new variables
Example 10b

```java
public class Example10b {

    public static void main(String[] args) {
        int x;
        Scanner scanner = new Scanner(System.in);

        String s = scanner.next();

        if (s.equals("dog")) {
            x = 10;
        }
        System.out.println("x is " + x);
    }
}
```
Example 10d

```java
public class Example10d {

    public static void main(String[] args) {
        int x;
        boolean foundDog = false;  // this is an example of a "flag"
        Scanner scanner = new Scanner(System.in);

        String s = scanner.next();

        if (s.equals("dog")) {
            x = 10;
            foundDog = true;
        }

        if (!foundDog) {
            x = 12;
        }
        System.out.println("x is " + x);
    }
}
```