Lecture 6: 
If-Else-If and Loops

Last time:
1. Finish Scanner
2. if statements
Today:
1. More on if
2. Project assigned
3. Named constants in Java
4. Loops

Nested If/Elves can be
Ugly and Confusing!

- Too many lines with only one { }
- Easy to get lost in indentation
- However, we can use Java’s “innermost if” rule for elses to program more clearly!

The “Dangling Else” Problem

- Which “if” an “else” is associated with can be ambiguous!
- Java rule: else is associated with “innermost” possible if
- Good programming practice: when in doubt, use { … }
- WE WILL USE { … } FOR ALL IF, IF-ELSE, IF-ELSE-IF, STATEMENTS
Cascading Elses

- A common programming paradigm:
  - “if something is true, do one thing”
  - “otherwise, if something else is true, do another thing”
  - “otherwise, if something else entirely is true, do yet another thing”
  - “otherwise, take a default action”
- How might we program this?

A Common Programming Idiom

- “Idiom” = “convention”
- if (C1) {
  
  S1;

  } else if (C2) {
  
  S2;

  ...

  } else {
  
  Sn;

  }

- Note indentation and curly bracket conventions!

In Projects

- You must use meaningful variable names and good indentation.
- Java convention indenting
  - showing the purpose
  - braces in the correct places with respect to the lines
- Java convention of capitalization of identifiers
- Fully blocked if statements
- Lines less than or equal to 80 columns
- You must use “named constants” for any literal values that will not change during program execution.
Variable Name Conventions

- What is legal for variable names?
  - Letters, digits, $, _
  - Can’t start variable name with digit
  - Avoid reserved words
- Use camelCase:
  - Variables & Methods use lower-case for first letter
  - Classes/Interfaces use upper-case for first letter

Variable Name Conventions: Examples

- Naming Conventions: Standards developed over time.
  - Variables and methods: Start with lowercase, and use uppercase for each new word:
    ```java
    String favoriteClass
    ```
  - Class names: Start with uppercase and uppercase for each new word:
    ```java
    String JOptionPane
    ```
  - Named constants (variables whose value never changes): All uppercase with underscores between words:
    ```java
    int MAX_LENGTH
    int DAYS_PER_WEEK
    double BOILING_POINT
    ```
- Make variable names not too long, not too short
  - Bad: `theCurrentItemBeingProcessed`
  - Good: `currentItem`

Meaningful Variable Names

- Choose names for your variables to reflect their purpose
- Bad
  ```java
  String string = "";
  System.out.println ("Enter name: ");
  string = sc.next();
  if (string.equals ("John Doe")) ..
  ```
- Good
  ```java
  String name = "";
  System.out.println ("Enter name: ");
  name = sc.next();
  if (name.equals ("John Doe")) ..
  ```
Named Constants in Java

- Programs often contain literals (= values)
  - `if (temp >= 97 && temp <= 99)`
  - `System.out.print ("Enter integer: ");`
- If same value should be used in several places, how to ensure consistency?
  - Check on temperature may be performed more than once
  - Same prompt might be printed in several places
- Java answer: named constants

```
final int MAX_OK_TEMP = 99;
```

- Just like a regular variable declaration, except...
- Special term `final`
- Necessity of initial value
- Any variable name will work, but convention is to use all capitals
- Difference with regular values: assignment attempt leads to error!

```
final int MIN_OK_TEMP = 97;
final int MAX_OK_TEMP = 99;
```

```
if (temp >= MIN_OK_TEMP && temp <= MAX_OK_TEMP) ...
```

```
final String INT_PROMPT = "Enter integer: ";
System.out.print (INT_PROMPT);
```

Examples
Loops in Java

- So far our programs execute every program statement at most once
- Often, we want to perform operations more than once:
  - “Sum all numbers from 1 to 10”
  - “Repeatedly prompt user for input”
- Loops allow statements to be executed multiple times. Loop types in Java:
  - while
  - do-while
  - for
- We will study while, do-while now, for-loop later

while and do-while Loops

- while and do-while loops contain:
  - A statement, called the body
  - A boolean condition
  - Idea: the body is executed as long as the condition is true

- while-loop: The condition is tested before each body execution
  ```java
  while (condition) {
    body
  }
  ```

- do-while-loop: The condition is tested after each body execution
  ```java
  do {
    body
  } while (condition);
  ```

- Main difference: do-while loop bodies always executed at least once because it is “bottom tested” rather than “top tested”

Example 11

```java
public class Example11 {
  public static void main(String[] args) {
    int i = 1;
    while (i <= 10) { // Loop counter
      System.out.println(i);
      i = i + 1; // Increment of loop counter ensures progress toward loop termination
    }
  }
}
```
Infinite Loops

- Loops can run forever if condition never becomes false
- Be careful when programming loops!
  - Add statements for termination into loop body first
  - Make sure these statements are at end of body
  - e.g.

```java
while (i <= 10) {
    System.out.println(i);
    i = i + 1;
}
```

Example 11b: Lots of Looping

```java
public class Example11b {
    public static void main(String[] args) {
        int i = 1;
        long total = 0;
        while (i <= 1000000) {
            total = total + i;
            i = i + 1;
        }
        System.out.println("Total is: "+ total);
    }
}
```

Example 12: do-while

```java
public class Example12 {
    public static void main(String[] args) {
        int i = 1;
        do {
            System.out.println(i);
            i = i + 1;
        } while (i <= 10);
    }
}
```
Variables, Blocks and Scoping

- Variables can be declared anywhere in a Java program
- When are the declarations active?
  - After they are executed
  - Only inside the block in which they are declared
- Scope rules formalize which variable declaration are active when
  - Global variables: scope is entire program
  - Local variables: scope is a block

Example 13

```java
import java.util.Scanner;
public class Example13 {
    public static void main(String[] args) {
        int i = 1;
        Scanner sc = new Scanner(System.in);
        do {
            System.out.print("Enter an integer from 1 to 10: ");
            int answer = sc.nextInt();
            if (answer < 1 || answer > 10) {
                System.out.println("Try again.");
            } else {
                System.out.println("Good job.");
            }
        } while (answer < 1 || answer > 10);
    }
}
```

Nested Loops

- while, do-while are statement constructors (like if and if-else: they use blocks)
- Loops can thus be used inside other loops!
Example 14

```java
public class Example14 {
    public static void main(String[] args) {
        int rowNumber = 1;
        while (rowNumber < 10) {
            int colNumber = 1;
            while (colNumber < 10) { System.out.print((rowNumber + colNumber) % 2); colNumber = colNumber + 1; }
            System.out.println(); rowNumber = rowNumber + 1; }
    }
}
```

In Projects

- You must use meaningful variable names
  - it must tell the purpose of that variable – what it is meant to hold
  - it cannot have so much abbreviation that only you can read it
- You must use Java convention indenting and brace placement
  - the indenting show the purpose in nesting
  - with braces in the “Java determined” places with respect to the lines of code
- Java convention of capitalization of identifiers
  - variables and methods start with lower case
  - classes and interfaces start with upper case
  - variables, methods, classes and interface use camelCase
  - constants are all uppercase with underscores between words
- You must have “Fully Blocked” if statements and looping structures
  - You must have all lines less than or equal to 80 columns of text
  - You must use “named constants” for any literal values that will not change during program execution.