In this section, we will cover using conditional logic, which allows our programs to determine when a piece of code should or should not be executed.

Conditional Logic

CSCI 1250 Study Guide

Schneider, Michael Joseph

If Statement

As seen in the below example, an "if" statement contains a conditional statement and surrounds a block of code. The surrounded block of code will only execute if the conditional statement evaluates to true.

```
int number = 5 * 2;

if(number < 20)//Conditional statement

{

      System.out.println("True statement");

}
```

In the example, our conditional statement is "number < 20". This means that "True statement" will only be printed to the screen if the value of number is less than 20.

Else Statement

An else statement can be placed after an if statement. It does not contain a conditional statement. This is because an else statement's code is only executed if the paired "if" statement evaluates to false. An else statement, therefore, must be paired with an "if" statement. It cannot be on its own.

```
int number = 5 * 5;

if(number < 20)//Conditional statement

{

      System.out.println("True statement");

}

else

{

      System.out.println("False statement");

}
```

Else If Statement

An “Else If” statement follows an “If” statement. Think of it as a secondary “If” statement. The conditional statement of an “Else If” is only checked if the preceding “If” statement evaluates to false.

```
int number = 5;
if(number <= 0)
{
      System.out.println(“Less Than Or Equal To Zero”);
}
else if(number <= 5)
{
      System.out.println(“Less Than Or Equal To Five”);
}
```

Just like “Else” statements, an “Else If” statement must be paired with an “If” statement. It cannot be on its own.

If, Else If, Else

We can use all three logic statements together. But there is a specific ordering that must be followed. We always start with “If” statement. This “If” statement can then be followed by as many “Else If” statements as we need. Finally, we can add an “Else” statement (but a maximum of only 1 “Else” Statement). The below example includes regular expression comments to help explain this relation.

```
if(conditional statement) // {1} Only One
{
      //If Code
}
else if(conditional statement)// * Zero or More
{
      //Else if
}
else // ?  Zero or One
{
      //Else
}
```

Below is an If, Else If, Else, example. In the example, the code will check the value entered into “number”. It will then print specific statements based on the value of “number”.

```
Scanner kb = new Scanner(System.in);
int value = kb.nextInt();
if(value == 0)
{
     System.out.println(“Zero!”);
}
else if(value < 5)
{
     System.out.println(“1 – 4!”);
}
else if(value < 10)
{
     System.out.println(“5 – 9!”);
}
else
{
     System.out.println(“10 – Infinity!”);
}
```

Switch Statement

A Switch Statement uses "cases" instead of conditional statements to determine what code should be executed. A "Switch" statement checks the value of an int, byte, short, char or String[1]. Each case represents a value for the variable being checked. Each case will need a "break" statement to ensure that the following cases aren't also executed! The only exception is the last "case". The last "case" doesn't need a "break" statement, because it doesn't have any following code to skip. The syntax for a switch statement is as follows:

```
datatype variable = value;
```

The variable being checked must be declared <u>and</u> initialized before it can be used in the switch statement. This is because we are checking its value. You can't check a variable's value if it doesn't have one.

value refers to the data entry being checked. This could be a variable, literal value, or even input from the user.

```
switch(variable)
{
        case value :
                //Code here
                break;
        case otherValue :
                //Code here
                break;//optional
}
```

value is a literal value that matches the data type of *variable*. Each case *value* must be unique. For example if *variable* is a:

- String – case "Hello" :
- int – case 1 :
- char – case 'A'

Each case statement must be followed with a " : " . This is part of Java syntax. The case statements <u>do not</u> use curly braces { }, while this will compile it is not necessary. Technically, you can add curly braces just about anywhere in a Java program.

[1] Older versions of the textbook state that a switch statement cannot use a String object. This is partially true. The functionality for using a String in a switch statement was only added in Java 7 (late 2011/early 2012). Which means older versions of Java cannot use Strings in switch statements. This shouldn't affect your code in class, but if you use a String object you must be using a version of Java 7+.

Switch Statement – Default Case

A default case is analogous to the “else” statement. It is only executed if all cases evaluate to false. Like the “else” statement, it is optional. It is not required for a switch statement to be used.

```
datatype variable = value;

switch(variable){

      case value:

            //Code here

            break;

      case otherValue:

            //Code here

            break;

      default:

            //Code here


}
```

The default case does not have a break statement. It is, by convention, the last “case” of a switch statement and doesn’t need to use “break” to skip any code.

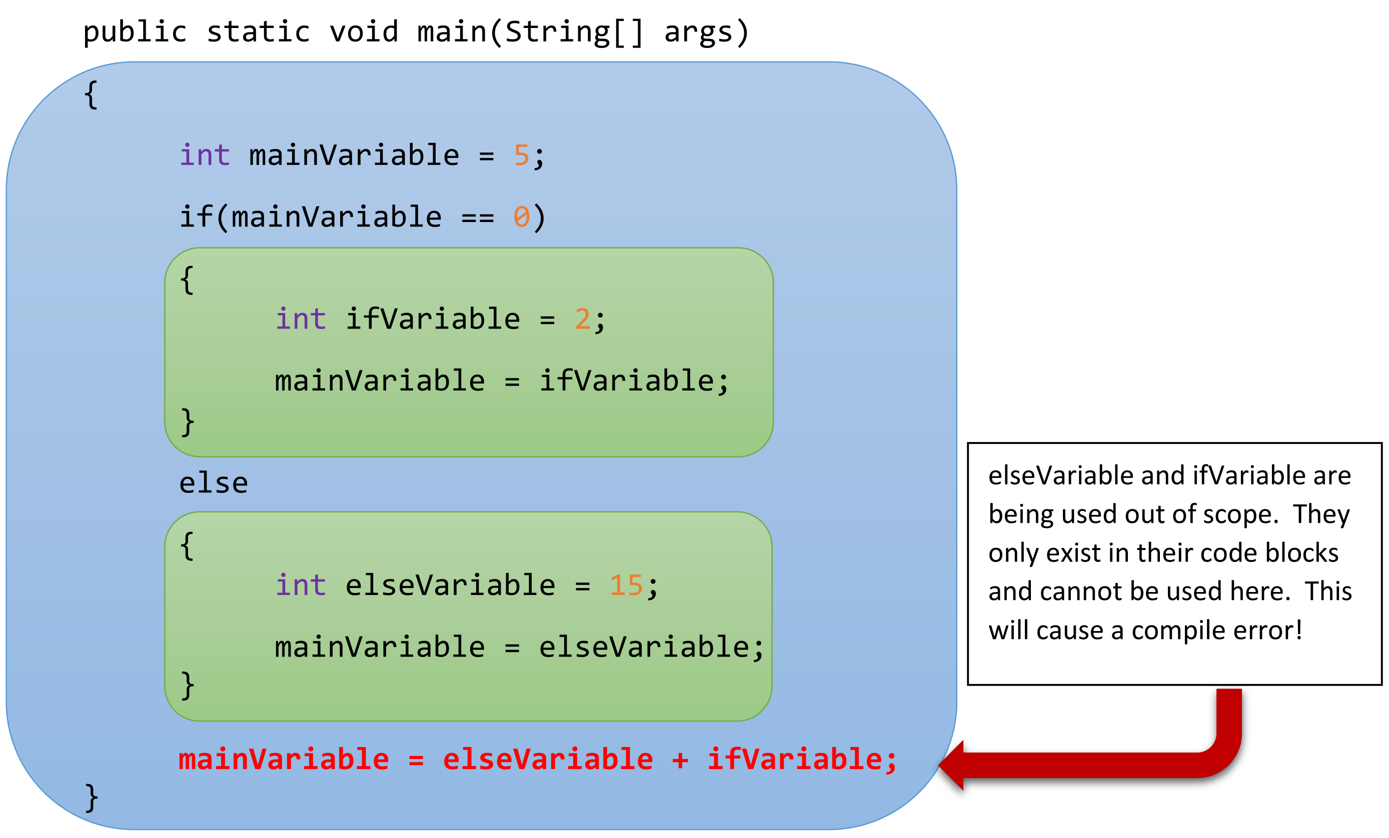
Switch vs If/Else If/Else Example

The following example first shows code using if/else if/else statements and then rewrites the code to be represented as a switch statement. The two are often interchangeable, in that any switch statement can be represented by if/else if/else statements.

```
Scanner kb = new Scanner(System.in);

int check = kb.nextInt();

if(check == 1)
{
      System.out.println(“One”);
}
else if(check == 2)
{
      System.out.println(“Two”);
}
else
{
      System.out.println(“All other values”);
}



switch(check)

{

      case 1:

            System.out.println(“One”);

            break;

      case 2:

            System.out.println(“Two”);

            break;

      default:

            System.out.println(“All other values”);

}
```

Variable Scope

In simplest terms, a set of curly braces “{ }” creates a code block. A code block creates a barrier. We can only use variables within the code block they were declared in. Using a variable outside of its code block is referred to as using the variable “out of scope”.



```
public static void main(String[] args)
{
     int mainVariable = 5;
     if(mainVariable == 0)
     {
          int ifVariable = 2;
          mainVariable = ifVariable;
     }
     else
     {
          int elseVariable = 15;
          mainVariable = elseVariable;
     }
     mainVariable = elseVariable + ifVariable;
}
```

elseVariable and ifVariable are being used out of scope. They only exist in their code blocks and cannot be used here. This will cause a compile error!

Example Exercises

ConditionalLogicExercises.java

ConditionalLogicAnswers.java