Week 2 – Topics

- Boolean Operators
- Conditional Logic
  - if → else
  - if → else if → else
  - Switch
- Loops
  - for Loop
  - while → Loop
  - do while → Loop
- User Input

Boolean Operators

Boolean Operators

Using programming to automate tasks means the computer needs a way to make decisions.

Decisions require comparing and evaluating information available and then deciding which way to proceed.

Boolean values are essential in these comparisons. With each decision we need the computer to make, we eventually answer in a yes or no manner - in Java, that yes or no is represented by: true or false.

The Java Boolean operator is also a data type just like int, float, or char. It is used where the condition true or false is needed, where the answer needs to be either 1 or 0. 1 being true and 0 being false.

List of Boolean operators:

Operator	Meaning
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
==	Equal (type matters)
!=	Not Equal

Content Link in LMS

Conditional Logic

if → else Conditionals

if Statement

```
if (/*Boolean Expression - Conditional expression that results in a Boolean value*/) {
  //code to run if Boolean expression in parentheses evaluates to true
}
```

if → else Statement

```
if (/*Boolean Expression #1*/) {
  //code to run if Boolean expression #1 in parentheses evaluates to true
} else {
  //code to run if Boolean expression #1 in parentheses evaluates to false
}
```

if → else if → else Statement

```
if (/*Boolean Expression #1*/) {
    //code to run if Boolean expression #1 in parentheses evaluates to true
} else if (/*Boolean Expression #2*/) {
    //code to run if Boolean expression #2 in parentheses evaluates to true
} else {
    //code to run if Boolean expression #1 & #2 in parentheses evaluates to false
}
```

```java
package conditionals;

class IfElseDemo {
    public static void main(String[] args) {

        int testscore = 76;
        char grade;

        if (testscore >= 90) {
            grade = 'A';
        } else if (testscore >= 80) {
            grade = 'B';
        } else if (testscore >= 70) {
            grade = 'C';
        } else if (testscore >= 60) {
            grade = 'D';
        } else {
            grade = 'F';
        }
        System.out.println("Grade = " + grade);
    }
}
```

switch Statement

There is also another programming construct we can use to create logical paths with multiple options in a similar fashion.

This construct is called a switch statement and is used to evaluate a variable and then provide multiple different code blocks that could be executed based on the value of the variable.

```
package conditionals;

public class SwitchDemo {
    public static void main(String[] args) {

        int month = 8;
        String monthString;
        switch (month) {
            case 1:  monthString = "January";
                     break;
            case 2:  monthString = "February";
                     break;
            case 3:  monthString = "March";
                     break;
            case 4:  monthString = "April";
                     break;
            case 5:  monthString = "May";
                     break;
            case 6:  monthString = "June";
                     break;
            case 7:  monthString = "July";
                     break;
            case 8:  monthString = "August";
                     break;
            case 9:  monthString = "September";
                     break;
            case 10: monthString = "October";
                     break;
            case 11: monthString = "November";
                     break;
            case 12: monthString = "December";
                     break;
            default: monthString = "Invalid month";
                     break;
        }
        System.out.println(monthString);
    }
}
```

Loops

For → Loop

```java
package loops;

class ForDemo {
    public static void main(String[] args) {

        /*
         * Basic for loop structure:
         *
         *   for (initialization; termination; increment) {
         *        statement(s)
         *   }
         */

        for (int i = 1; i <= 1000; i++){
            System.out.println("Count is: " + i);
        }
    }
}
```

Content Link in LMS

while → Loop and do while → Loop

```java
package loops;

class WhileDemo {
    public static void main(String[] args){
        int count = 1;
        while (count < 11) {
            System.out.println("Count is: " + count);
            count++;
        }

        System.out.println("--------------------");

        do {
            System.out.println("Count is: " + count);    // Notice that this code runs once even though the count is not <  11
            count++;
        } while (count < 11);
    }
}
```

Content Link in LMS

Branching

Branching - break vs. continue Statements

- Branching statements allow the flow of execution to jump to a different part of the program

- The common branching statements used within other control structures include:
  - break - used to break completely out of a loop
  - continue – used to break one iteration in a loop
  - return - exits from the current method, and control flow returns to where the method was invoked.

```java
package branching;

public class BreakContinue {

    public static void main(String[] args) {

        // Break example - used to jump out of a loop completely.
        System.out.println("*** Count loop with break");
        for (int i = 0; i < 10; i++) {

            if (i == 4) {
                break;
            }
            System.out.println(i);
        }

        // Continue example - used to break one iteration in the loop.
        System.out.println("-----------------------------------");
        System.out.println("*** Count loop with continue");
        for (int i = 0; i < 10; i++) {
            if (i == 4) {
                continue;
            }
            System.out.println(i);
        }
    }

}
```

Content Link on Branching

User Input

User Input

We need data to tell our programs to make decisions with; and up to this point, we've been hard coding data into variables to explore coding constructs.

However, the original source for most data is **user input**. In order to make decisions based on responses or data entered from a user, let's take a look at one way we can prompt a user to enter some data and then store the data in a variable to use in our code.

To receive **user input** in Java, we can use the Scanner object. It is in java.util package, so we need to import it.

```java
package user.input;

import java.util.*;

public class UserInputDemo {
    public static void main(String[] args)  {
        // System.in is a standard input stream
        Scanner sc= new Scanner(System.in);

        System.out.print("Enter username:  ");
        String username = sc.nextLine();

        System.out.print("Enter password: ");
        String password = sc.nextLine();

        if (username.equals("samy123") && password.equals("12345")) {
            System.out.println("Welcome back " + username);
        } else {
            System.out.println("***Inaccurate credentials!");
        } // end of else
        sc.close();
    }   // end of main()
}  // end of UserInputDemo class
```

*** Note: this method of interacting with a user is a temporary method for the purpose of being able to receive user input and is not the recommended way in live, production code to interact with users. We will learn additional ways later on.*

Content Link in LMS

Coding Challenge

Try the coding challenge from the User Input page in the LMS

Take the login code example with while loop and enhance it

Add a login retry attempt count that would enable the user to only enter the incorrect credentials a certain number of times before displaying a message like "You are locked out!" and ending the loop.

```java
package user.input;

import java.util.Scanner;

public class Login {

    public static void main(String[] args) {

        boolean loggedIn = false;
        Scanner sc= new Scanner(System.in);

        while (!loggedIn) {
            System.out.print("Enter username: ");
            String username = sc.nextLine();
            System.out.print("Enter password: ");
            String password = sc.nextLine();

            // Validate username and password
            if (username.equals("samy123") && password.equals("12345")) {
                System.out.println("*** Welcome back " + username);
                loggedIn = true;
            } else {
                System.out.println("*** Inaccurate credentials!");
            } // end of else
        } // end of while

        sc.close();
    }

}
```

https://github.com/ckiefriter1/Java-Code-Examples/blob/main/src/user/input/Login.java

Appendix