Discussion-4: Repetition Control Structure (Loops).

Discussion Topic:
What are the key constructs required to create a loop? Identify two scenarios that may require two different types of loops. Be sure to provide specific details for each scenario that illustrate why the different types of loops are required. Actively participate in this discussion by providing constructive feedback on the scenarios and details posted by your peers.

My Post:

Hello class,

The key constructs required to create a loop in programming are:

- Loop initialization: A variable that initiates the loop
- Loop counter: A variable that increments and decrements
- Loop condition: A condition that is verified at each iteration

Java provides several loop statements that allow you to repeat a block of statements under specified conditions. The four main loop statements in Java are:

1. **For Loop**: It is a counter-controlled loop. It iterates over numbers by incrementing or decrementing a counter variable until a condition is met.
   Example:
   ```
   // Prints numbers from 1 to 5
   for (i = 1; i <= 5; i++) {
       System.out.println(i);
   }
   ```
2. **While Loop**: The loop executes the block of code as long as the specified condition is true.
   Example:
   ```
   // Prints numbers from 1 to 5
   int i = 1;
   while (i <= 5) {
       System.out.println(i);
       i++
   }
   ```
3. **Do-While Loop**: Similar to the while loop, with the difference that it guarantees the block of code to be computed at least once, as the condition is evaluated after the execution of the loop.
   Example:
   ```
   // Prints numbers from 1 to 5 at least once even if the condition is initially
   false
   int i = 1;
   do {
       System.out.println(i);
       i++
   } while (j <= 5);
   ```

4. **Enhanced For Loop** (also known as the for-each loop): It iterates through a list or string.
   Example:
   ```
   // Prints each element in an array
   int[] numbers = {1, 2, 3, 4, 5};
   for (int num : numbers) {
       System.out.println(num);
   }
   ```

Two scenarios that may require two different types of loops could be:

Scenario-1:
Iterate through the word "Mississippi" To count the number of occurrences of the letter 'i'.

```
String word = "Mississippi";
int count = 0;
for (int i = 0; i < word.length(); i++) {
    if (word.charAt(i) == 'i') {
        count++;
    }
}
System.out.println("Number of 'i' in Mississippi: " + count);
```

The for loop is ideal for counting occurrences of the letter 'i' in "Mississippi" because:

1. Known Iterations: You know the exact number of characters to iterate through.
2. Efficiency: Accesses each character directly by index, allowing precise checks and updates within a single loop pass.
3. Clarity: Combines initialization, condition checking, and incrementation in one compact statement, making the code easier to manage and less prone to errors.

Scenario-2:
If you need to validate a user's input to ensure it meets specific criteria, for example, a PIN code needs to be 4 digits long.

```
import java.util.Scanner;
public class PINValidation {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String pin;
        while (true) {
            System.out.print("Please enter a 4-digit PIN: ");
            pin = scanner.nextLine();
            // Check if the PIN is exactly 4 digits and numeric
            if (pin.matches("\\d{4}")) {  // Regex validation
                System.out.println("PIN You entered is a valid PIN.");
                break;  // exists the loop
            } else {
                System.out.println("Invalid PIN. It must be exactly 4 digits.");
            }
        }
        scanner.close();
```

```
        }
}
```
The while loop is ideal for validating a user's PIN input because:

1. Indeterminate Iterations: It handles an unknown number of user attempts effectively until a valid PIN is entered.
2. Immediate Feedback: Allows for immediate correction prompts if the PIN does not meet the criteria.
3. Simple Control Flow: Uses a straightforward loop with a condition at the start, ensuring the criteria are checked before processing continues.
4. Flexible: Efficiently manages multiple incorrect inputs by looping until the correct input is provided.

-Alex

References:

Colorado State University Global (2024). Module 4: Repetition control structure (loops). In Colorado State University Global, CSC320: Programming I.