Unit 4: Iteration
While Loops

Adapted from:
1) Building Java Programs: A Back to Basics Approach by Stuart Reges and Marty Stepp
2) Runestone CSAwesome Curriculum

https://longbaonguyen.github.io
Loops

A loop in programming, also called iteration or repetition, is a way to repeat one or more statements.

If you didn’t have loops to allow you to repeat code, your programs would get very long very quickly!

Using a sequence of code, selection (ifs), and repetition (loops), the control structures in programming, you can construct an algorithm to solve almost any programming problem.
**The while loop**

**while loop:** Repeatedly executes its body as long as a logical test is true.

```java
while (test) {
    statement(s);
}
```

When the test condition is false, we exit the loop and continue with the statements that are after the body of the while loop.

If the condition is false the first time you check it, the body of the loop will not execute.
public class LoopTest1{
    public static void main(String[] args) {
        // 1. initialize the loop variable
        int count = 1;
        // 2. test the loop variable
        while (count <= 5) {
            System.out.println(count);
            // 3. change/update the loop variable
            count++;
        }
    }
}

What is the value of count after the loop? (Answer: 6)
Curly braces mark the body of methods, for loops and conditional blocks. They are not necessary if the body or the block consists of only one statement.

Without curly braces to denote a while loop body, by default the body only contains one statement.

The following are equivalent.

```java
int x = 1;
while(x <= 10){
    x++; 
}

int x = 1;
while(x <= 10)
    x++; 
```
The following are equivalent.

```java
int x = 1;
if(x <= 10) {
    System.out.println(x);
}

int x = 1;
if(x <= 10)
    System.out.println(x);
```
The following are NOT equivalent. What is the output for each?

```java
int x = 7;
while(x <= 10){
    x++;
    System.out.print(x);
}
```

Output: 
8 9 10 11

```java
int x = 7;
while(x <= 10)
    x++;
    System.out.println(x);
```

Output: 
11
int count = 1;

while(count > 0){
    count++;
}
System.out.println(count);

Output:
-2147483648  (Integer.MIN_VALUE)

- count will exceed the Integer.MAX_VALUE for an integer and will wrap back to the negative side to Integer.MIN_VALUE.
int count = 10;

while (count < 13)
    System.out.println(count + " ");
    count++;

– Infinite loop!! the count++; statement is not part of the body of the while loop. It will repeatedly print 10 over and over again.
int count = 10;

while(count < 13){
    System.out.println(count + " ");
    count++;
}

– Correct!

Output:
10 11 12
Basic Loop Algorithms

Important basic algorithms that use loops:

1) Compute a sum of a series (list of numbers)
2) Determine the frequency with which a specific criterion is met (for example, divisibility)
3) Identify the individual digits in an integer

We will do an example of each of the above.
Compute a sum

Write a loop to compute the sum:

\[ 1 + 2 + 3 + \ldots + 99 + 100 \]

```java
int sum = 0;
int number = 1;
while (number <= 100) {
    sum += number;
    number++;
}
```

cumulative sum: A variable that keeps a sum in progress and is updated repeatedly until summing is finished.

- The `sum` in the above code is an attempt at a cumulative sum.
Write a loop to determine how many numbers from 1327 to 4542 that are multiples of 3 but not multiples of 5.

```java
int count = 0;
int num = 1327;
while(num <= 4542){
    if(num % 3 == 0 && num % 5 != 0){
        count++;
    }
    num++;
}
```
Extracting Digits

Code that can extract digits from a number is useful. (last four digits of a social security number, whether digits form a valid credit card number)

The trick is to repeatedly modulo 10 and integer divide by 10. For example:

```c
int num = 1347;
int ones = num % 10;  // extract the last digit: 7
num /= 10;  // remove the last digit, num = 134
int tens = num % 10;  // extract the last digit: 4
num /= 10;  // remove the last digit again, num = 13
int hundreds = num % 10;  // extract the last digit: 3
num /= 10;  // num = 1
int thousands = num % 10;  // 1
```
For numbers of arbitrary lengths, we can use a while loop to implement the previous algorithm!

```java
Scanner console = new Scanner(System.in);
System.out.print("Enter number: ");
int number = console.nextInt();
while(number != 0){
    // extract last digit
    System.out.println(number % 10);
    number /= 10; // remove last digit
}
```

Output:
Enter a number: 2348
8
4
3
2
You can write all these methods on the same repl on repl.it.

Write a static method `countFactors` that accepts in integer parameter returns the number of factors of the integer.

- `countFactors(24)` returns 8 because 1, 2, 3, 4, 6, 8, 12, and 24 are factors of 24.

Write a static method `isPrime` which returns whether or not an integer is prime. This method **must call** `countFactors`.

- Example: `isPrime(27)` returns false and `isPrime(47)` returns true.
Write a static method `countPrimes` that accepts an integer parameter `n` and returns the number of primes from 2 to `n`.

- `countPrimes(24)` returns 9 because 2, 3, 5, 7, 11, 13, 17, 19, 23 are primes less than or equal to 24.

1) How many primes are less than 1,000,000?
Write a static method named `fourHeads` that repeatedly flips a coin until four heads *in a row* are seen. You should use `Math.random()` to give an equal chance to a head or a tail appearing. Each time the coin is flipped, what is seen is displayed (H for heads, T for tails). When four heads in a row are flipped a congratulatory message is printed. Here are possible outputs of two calls to `fourHeads`:

```
T T T H T H H H H
Four heads in a row!

T H T H T T T T H H T H H H H
Four heads in a row!
```
References

1) Building Java Programs: A Back to Basics Approach by Stuart Reges and Marty Stepp

2) Runestone CSAwesome Curriculum: https://runestone.academy/runestone/books/published/csawesome/index.html

For more tutorials/lecture notes in Java, Python, game programming, artificial intelligence with neural networks:

https://longbaonguyen.github.io