Here's the basic syntax of the `for` loop:

```java
for (initialization; condition; update) {
    // Code to be executed in each iteration
}
```

Here's how the `for` loop works in more detail:

1. **Initialization**:
   The loop starts with an initialization step, where you initialize a loop control variable. This step is executed only once at the beginning.

2. **Condition**:
   The loop continues as long as the condition is `true`. If the condition evaluates to `false`, the loop stops executing.

3. **Update**:
   After each iteration, the update step is executed. It typically updates the loop control variable, moving the loop closer to the termination condition.

4. **Code Execution**:
   The code block within the loop is executed as long as the condition remains `true`. After each iteration, the update step is executed, and then the condition is evaluated again.

Here's an example of using the `for` loop to iterate through numbers from 1 to 5:

```java
for (int i = 1; i <= 5; i++) {
    System.out.println(i);
}
```

In this example:
- The initialization sets `i` to 1 at the start.
- The condition checks if `i` is less than or equal to 5. If `i` exceeds 5, the loop stops.
- After each iteration, `i` is incremented by 1 using the `i++` increment operator.

Enhanced `for` Loop (for-each Loop):

Java also provides an enhanced version of the `for` loop called the "for-each" loop, which is designed for iterating over arrays or collections.

Here's the syntax of the enhanced `for` loop:

```java
for (datatype element : array) {
    // Code to be executed for each element
}
```

Here's an example of using the enhanced `for` loop to iterate over an array of names:

```java
String[] names = {"Alice", "Bob", "Charlie"};

for (String name : names) {
    System.out.println(name);
}
```

In this example, the loop iterates over each element (name) in the `names` array, making it especially useful when you need to process elements sequentially.

The `for` loop is versatile and widely used for performing repetitive tasks, such as iterating through arrays, generating sequences, or performing calculations. It's essential to understand its structure and usage for effective programming.