Let's look at labeled break with a slightly more simple example.

1. Let's first start from beginning where we don't use labeled break. Let's consider nested for loop shown below where outer-for loop is using the variable i while inner-for loop is using the variable j. Also, the variable num is incremented in inner-for. Here the *body* of **outer-for** iterates 2 times, i.e., for values i = 0 & i = 1. When i becomes 2, the condition in outer-for loop will be false, i.e., condition is i < 2, but since i is equal to 2 now, i < 2 will be false (as 2 < 2 is false) and so the for loop exits/terminates.

One thing should be clear in this nested for loop: when i = 0, inner-for loop runs 3 times (for values j = 0; j = 1 and j = 2) taking the num value to 3. Similarly, when i = 1, inner-for loop once again runs 3 times (again for the values j = 0; j = 1 and j = 2) incrementing the num value from 3 to 6. So, when outer-for terminates, the final value of num will be 6. Assuming this is clear let's move forward to understand labeled break

```
int num = 0;

for (int i = 0; i < 2; i++) {

  for (int j = 0; j < 3; j++) {

     num++;

  }

}
```

2. **Labeled break**: Recall that if there is a nested for-loop and if a break statement is used in an inner-for loop, then control will break out of the inner-for loop and will continue execution with the statement that immediately follows inner-for loop. Once all statements following inner-for loop are executed, control goes back into outer-for and the condition in outer-for is checked to see if we need to continue iterating with the body of outer-for.

Now, break; exits only immediately enclosing for-loop. Now, from within inner-for loop, if we want to directly exit outerfor loop, then we use labeled-break. Let's consider below example, which is almost similar to above nested for.

```
int num = 0;

xyz: for (int i = 0; i < 2; i++) {

  for (int j = 0; j < 3; j++) {

    if (i == 1 && j == 1) {

      break xyz;

    }

    num++;

    System.out.println("i: " + i + ", j: " + j + ", num: " +  num);

  }

}
```

Here outer-for is labeled with xyz and we are exiting outer-for when the condition (i == 1 && j == 1) is true in the inner for loop, i.e., by doing **break xyz;**. Here final num value will be 4. Let's see:

When i = 0, inner-for iterates 3 times (for j = 0, j = 1 and j=2). All 3 times the condition (i == 1 && j == 1) fails as i is 0 and not 1. So, we never enter the if-statement's body and the value num will be incremented to 3.

Now, when i = 1, the body of inner-for should again iterate 3 times (for j = 0, j = 1 and j=2). Here, when j is 0, the condition (i == 1 && j == 1) will fail as i is 1, but **j is 0.** So, num will become 4. Next, when j becomes 1, the condition (i == 1 && j == 1) will become true as i is 1 and j is 1 and this time break xyz; will be executed and the outer-for will be terminated, i.e., inner-for will not be iterated and similarly no more iterations for outer-for too. If it is still not clear, you can also copy the above code into a main method and run and see the final value of num that gets printed.

For labeled continue, instead of breaking the outer-for, the control continues with the next value of i, i.e., inner-for will not be iterated for the current value of i.