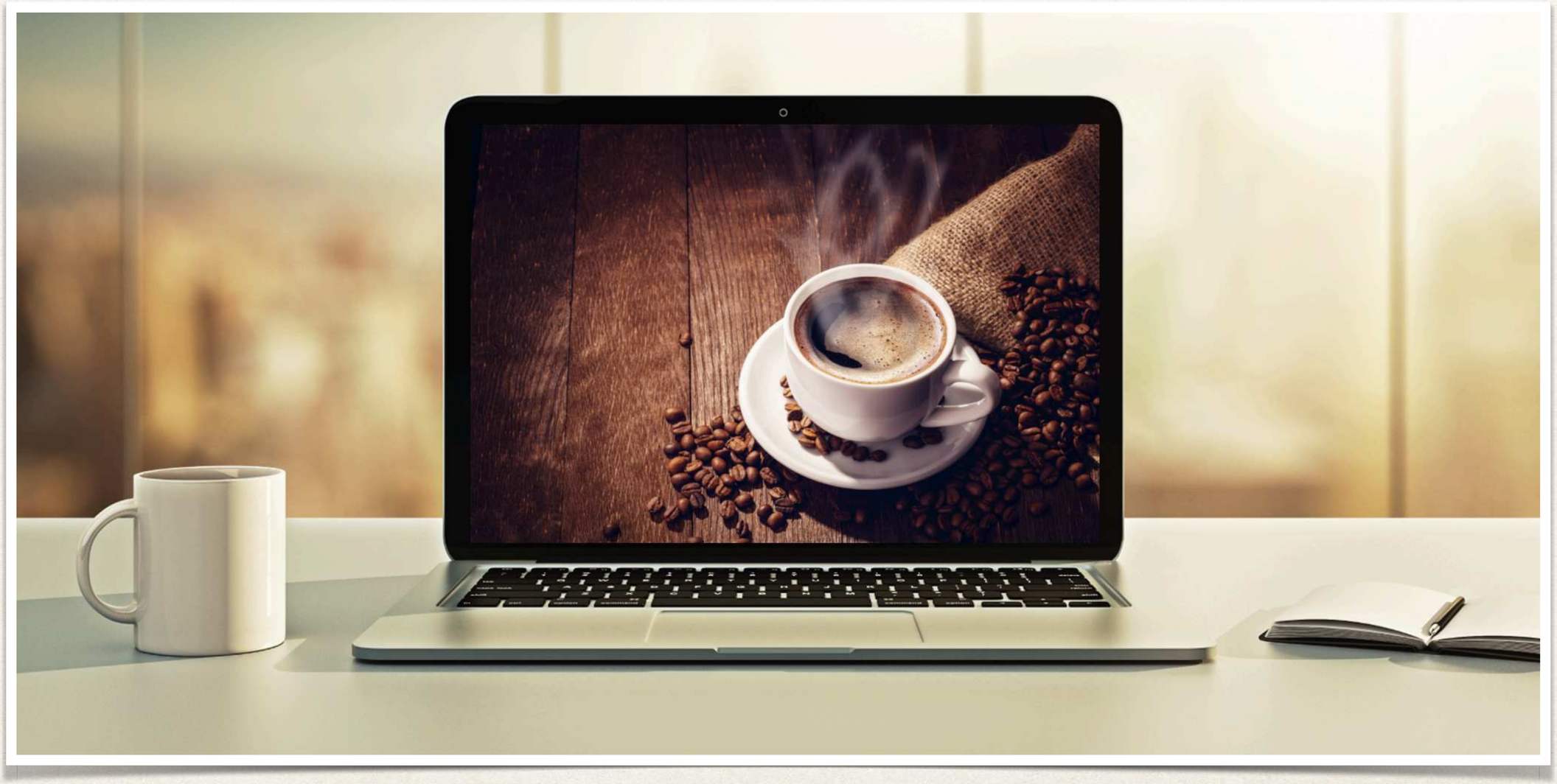
for loops



`for` loops

- `for` loops are useful for repeating a section of code multiple times
- For example, if we want to print a statement multiple times
- More efficient to use a `for` loop instead of multiple println

Basic Example

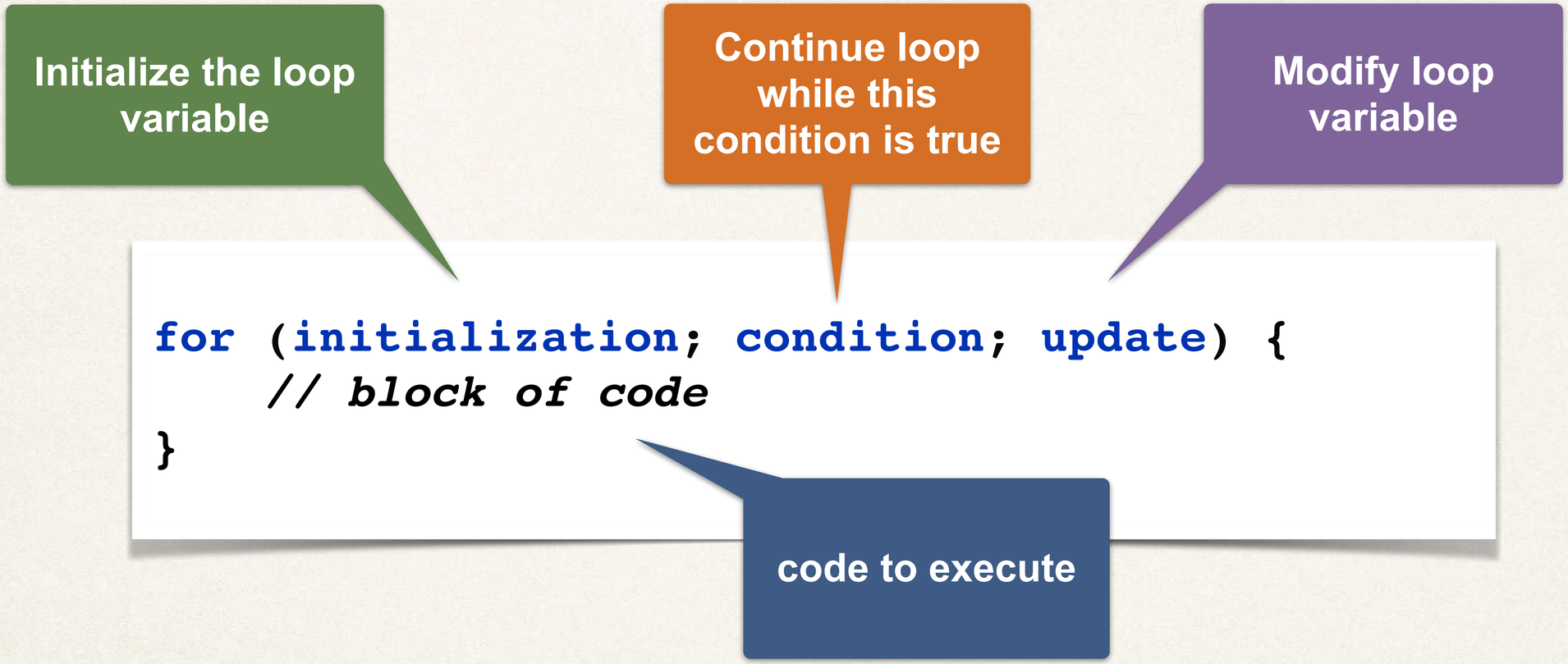
```
for (int i = 0; i < 100; i++) {
    System.out.println("Hello World");
}
```

Repeat this block of code 100 times

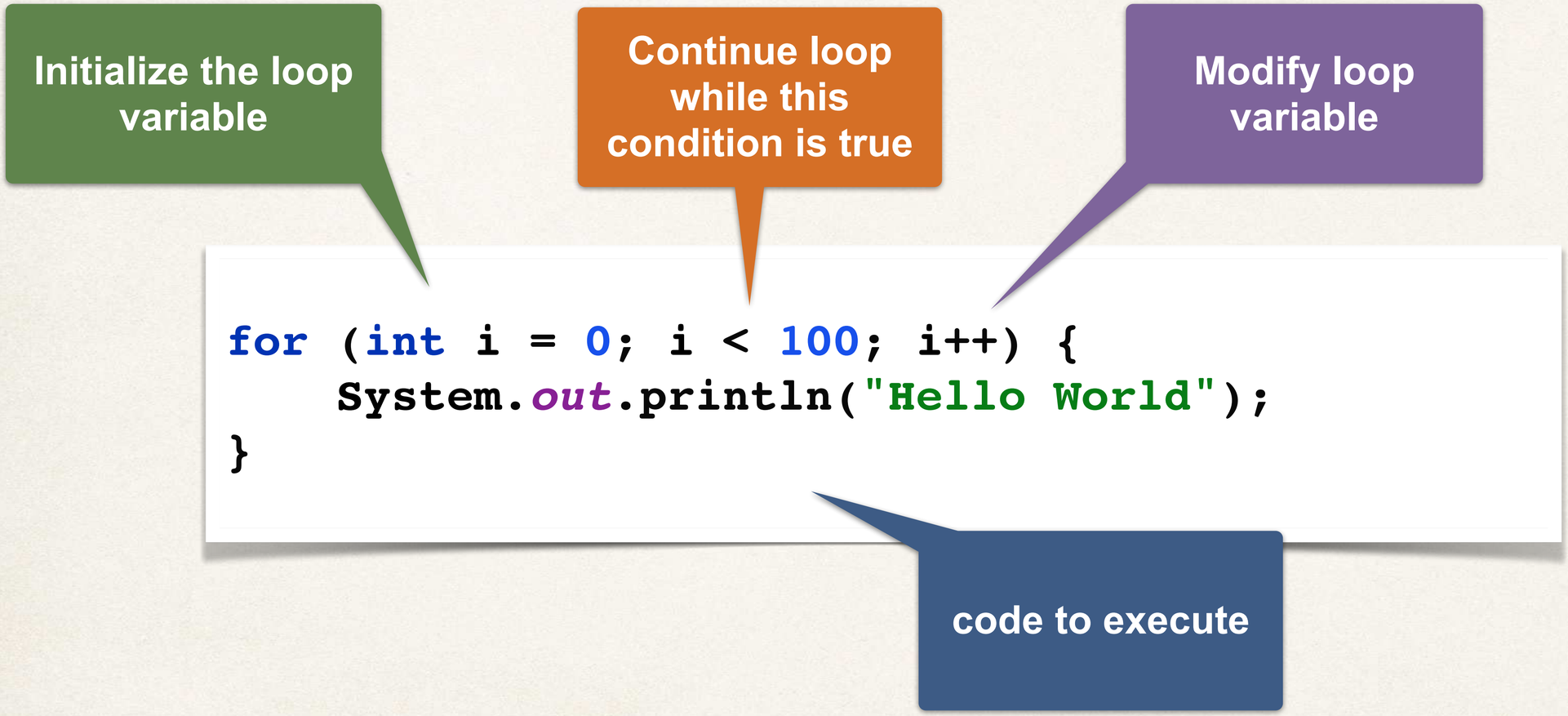
Print this message 100 times

```
Hello World
Hello World
Hello World
Hello World
Hello World
…
…
```

General Syntax



Applying Syntax to our example



Display Loop Variable

Loop 1 to 5

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

```
Counter: 1
Counter: 2
Counter: 3
Counter: 4
Counter: 5
```

Change Loop Increment

Loop 0 to 20
In increments of 5

```
for (int i = 0; i <= 20; i=i+5) {
    System.out.println("Counter: " + i);
}
```

```
Counter: 0
Counter: 5
Counter: 10
Counter: 15
Counter: 20
```

Count Down

Loop 5 down to 0

```
for (int i = 5; i >= 0; i--) {
    System.out.println("Counter: " + i);
}
```

```
Counter: 5
Counter: 4
Counter: 3
Counter: 2
Counter: 1
Counter: 0
```

Nested Loops

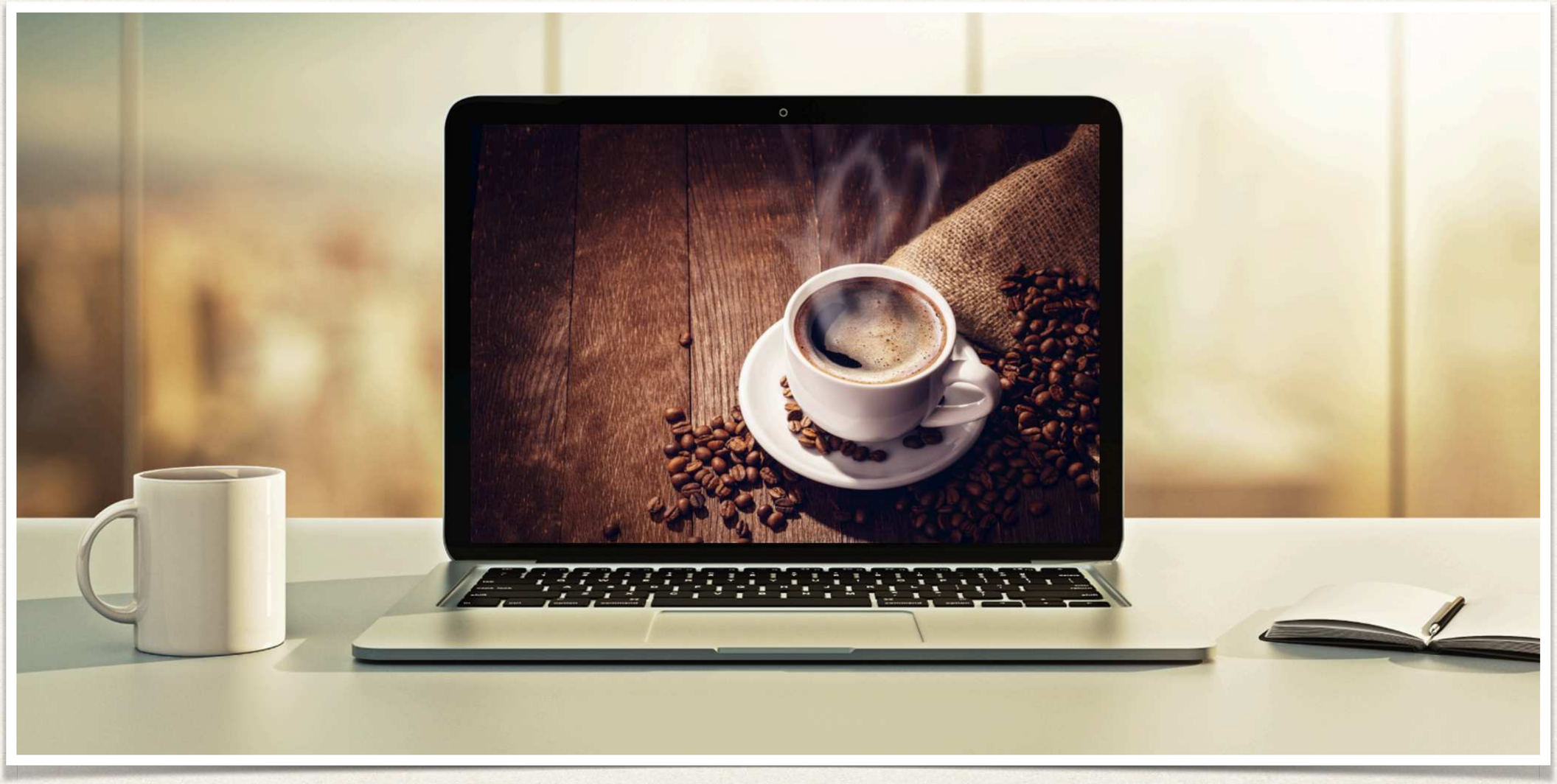
- This example will print the multiplication table: 5x5

```
for (int row = 1; row <= 5; row++) {

    for (int col = 1; col <= 5; col++) {
        int value = row * col;
        System.out.print(value + "\t");
    }

    System.out.println();

}
```

\t: tab character for alignment

```
1   2   3   4   5
2   4   6   8   10
3   6   9   12  15
4   8   12  16  20
5   10  15  20  25
```

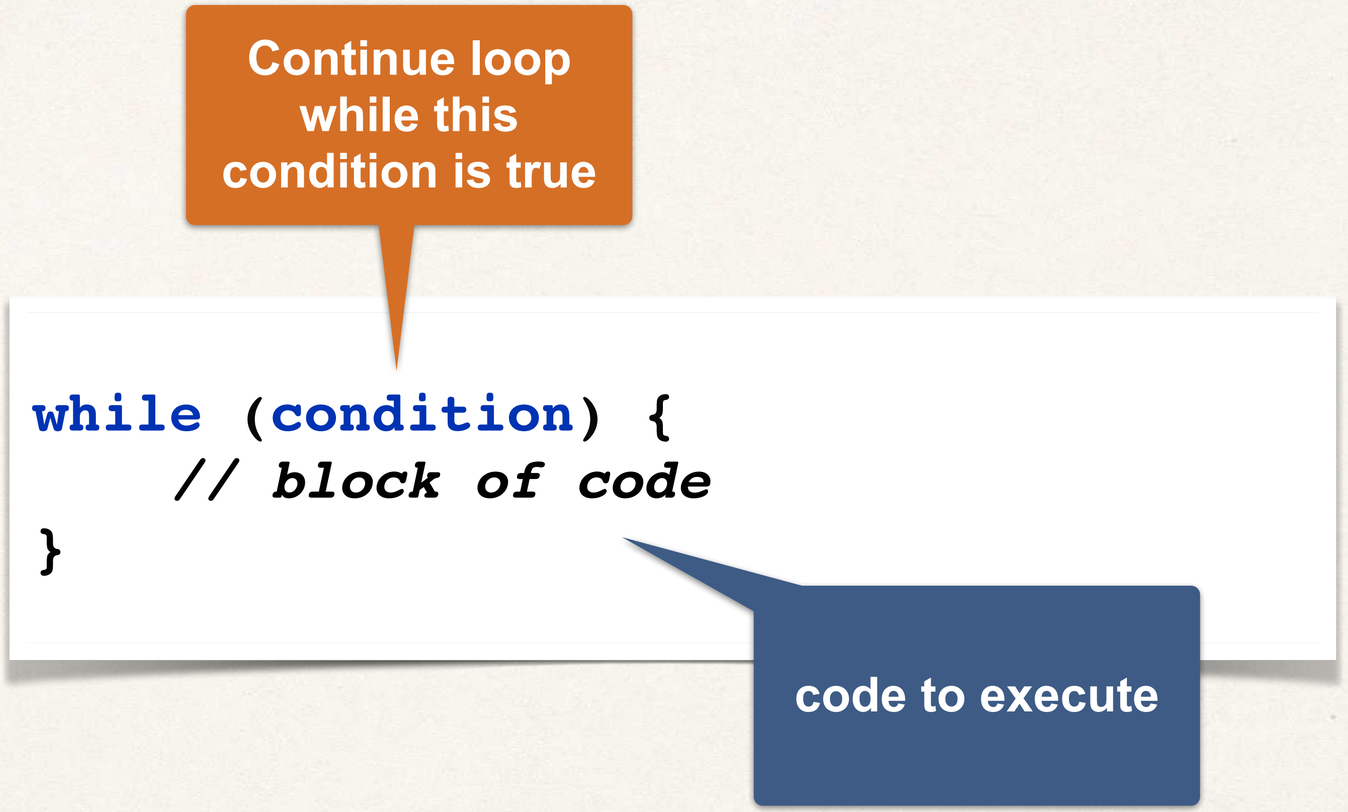
`while` loops



`while` loops

- `while` loops are useful for repeating a section of code multiple times
- Useful, when the number of iterations are unknown beforehand

General Syntax



Example: Prompt the User

```
Scanner scanner = new Scanner(System.in);

boolean done = false;

while (!done) {

    System.out.println("Hello world");

    System.out.print("Are we done? (Y/N): ");
    String userInput = scanner.nextLine();

    if (userInput.equalsIgnoreCase("Y")) {
        done = true;
    }

    System.out.println();
}

scanner.close();
```

Loop while the user wants to continue

```
Hello world
Are we done? (Y/N): N

Hello world
Are we done? (Y/N): N

Hello world
Are we done? (Y/N): Y

Process finished with exit code 0
```

Which One - `for` loop or `while` loop???

- `for` loops are useful when the number of iterations are known
- `while` loops are useful, when the number of iterations are unknown beforehand
- `while` loops are useful, if the condition could change based on user input or other conditional logic (ie account balance, game level, user Y/N, …)

Keep shopping until your balance falls below xxx

Keep playing until you reach level zzz

Prompt: Y/N

`do-while` loops

- A variant of the `while` loop
- The conditional is placed at the end of the loop
- Useful, when you want to execute the block of code at least once

General Syntax

code to execute
at least once

```
do {
    // block of code
} while (condition);
```

Continue loop
while this
condition is true

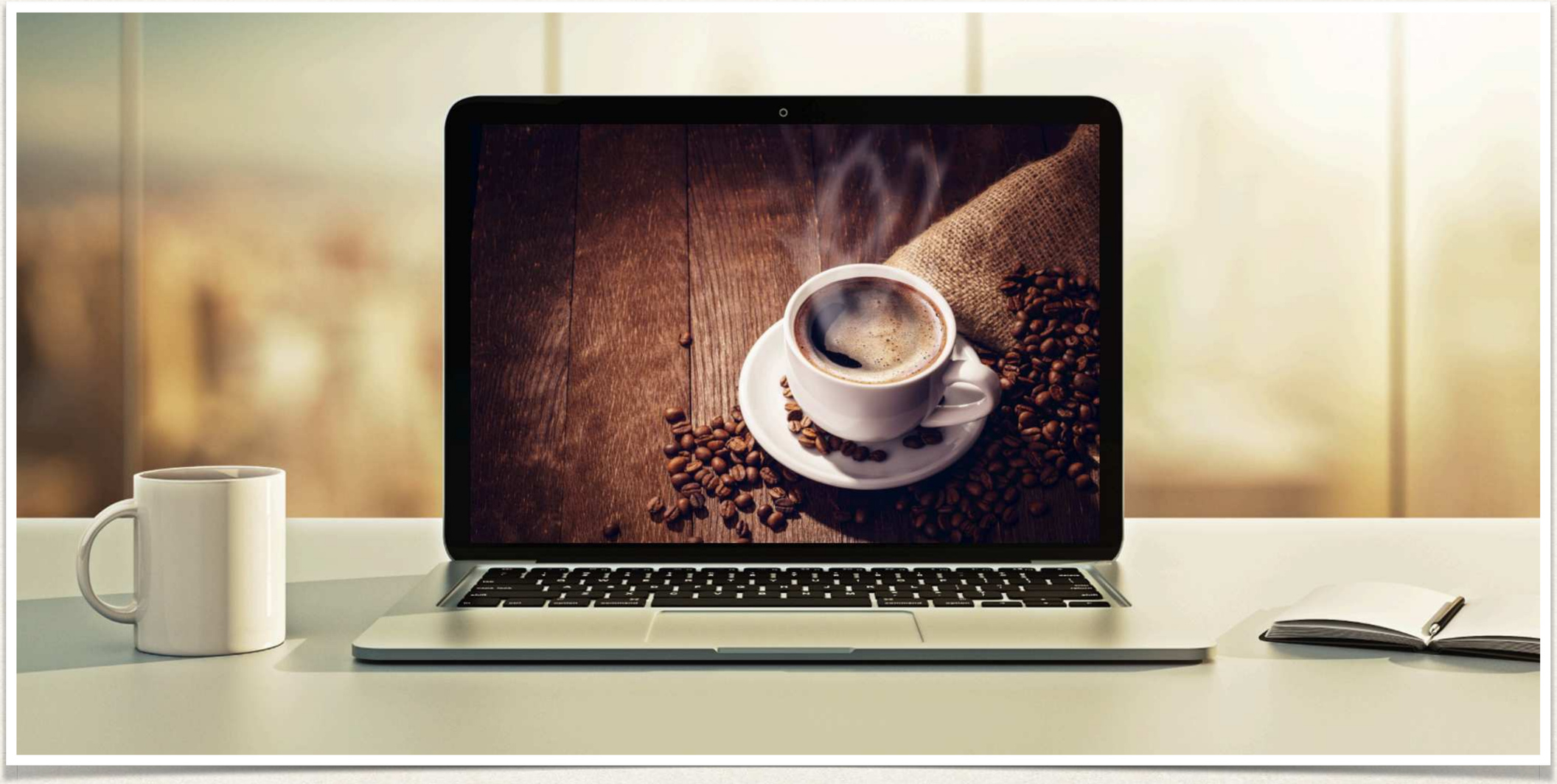
Example: do-while

```
Scanner scanner = new Scanner(System.in);

boolean done = false;

do {
    System.out.println("Hello world");

    System.out.print("Are we done? (Y/N): ");
    String userInput = scanner.nextLine();

    if (userInput.equalsIgnoreCase("Y")) {
        done = true;
    }

    System.out.println();
} while (!done);

scanner.close();
```

Loop while the user wants to continue

```
Hello world
Are we done? (Y/N): N

Hello world
Are we done? (Y/N): N

Hello world
Are we done? (Y/N): Y

Process finished with exit code 0
```

break and continue



`break` statement

- In loops, the `break` statement will exit out of the current loop

Example

```
for (int i = 1; i <= 10; i++) {

    if (i == 5) {
        break;
    }

    System.out.println("Counter: " + i);
}
```

Break at 5
exit the loop

```
Counter: 1
Counter: 2
Counter: 3
Counter: 4

Process finished with exit code 0
```

`continue` statement

- In loops, the `continue` statement will
  - Skip the remaing code in the block and continue with next iteration

- Think of it as “skipping over” current iteration

Example

```
for (int i = 1; i <= 10; i++) {

    if (i == 5) {
        continue;
    }

    System.out.println("Counter: " + i);
}
```

At 5, `continue`
skip remaining statements in
current iteration

Notice, we “skipped over” 5

```
Counter: 1
Counter: 2
Counter: 3
Counter: 4
Counter: 6
Counter: 7
Counter: 8
Counter: 9
Counter: 10

Process finished with exit code 0
```

Which One: `break` or `continue`???

- Use `break` when you want to terminate current loop
- Use `continue` when you want to skip remaining code in current iteration

Can also use `break` and `continue` with `while` and `do-while`