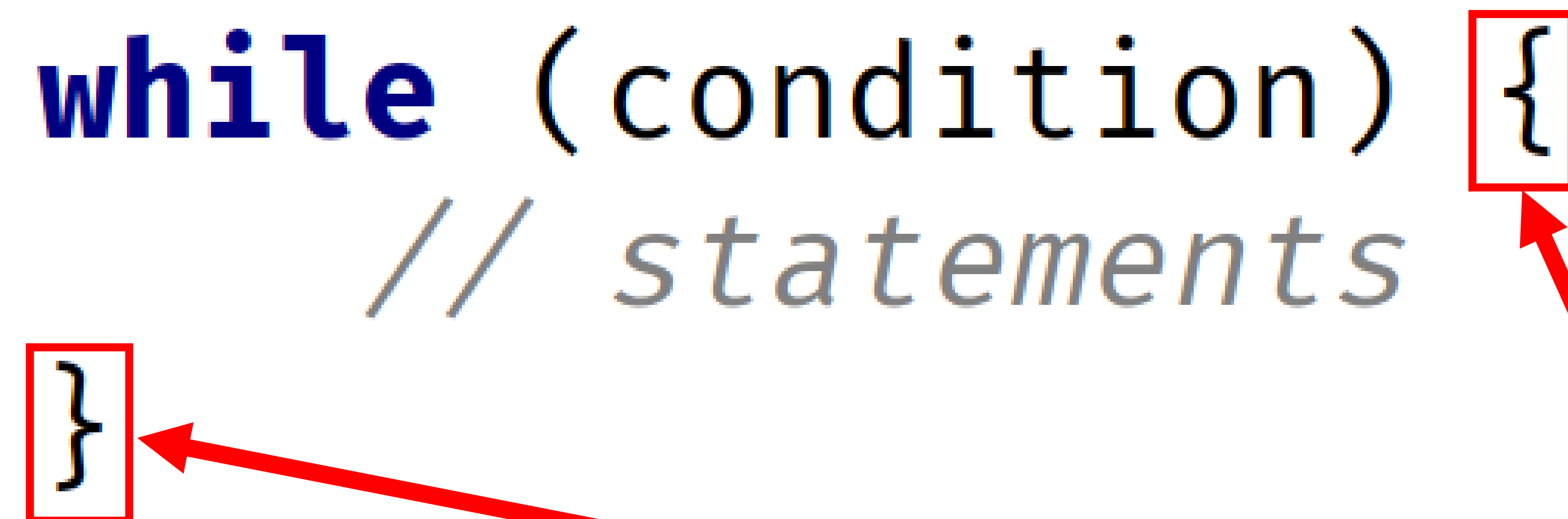


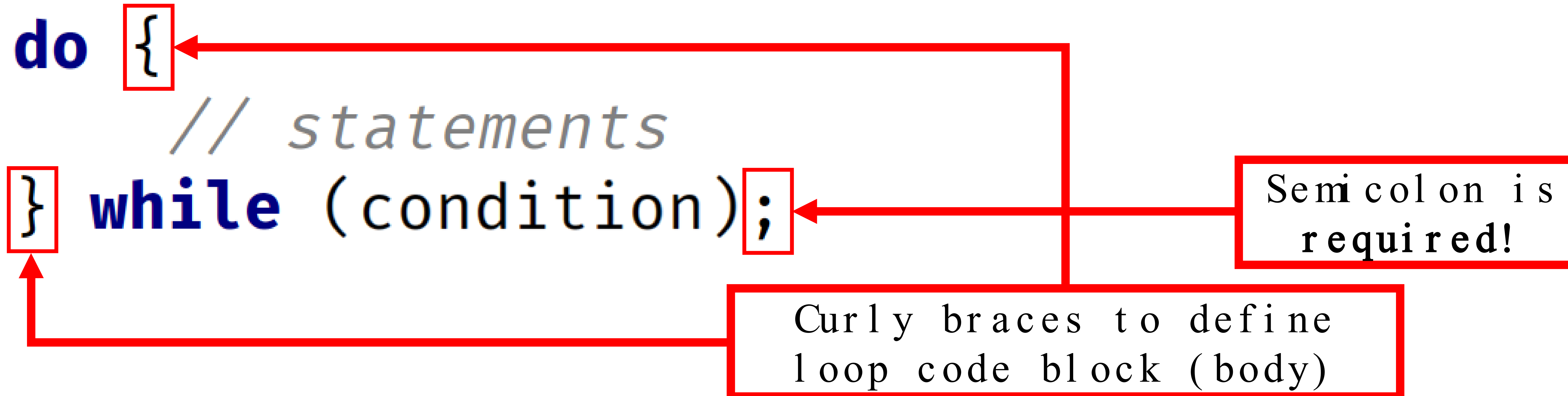
The while Statement Recap

```
while (condition) {  
    // statements  
}
```

The diagram shows the syntax of a while loop. The keyword 'while' is in bold blue, 'condition' is in a monospace font, and the opening curly brace '{' is in a red box. The comment '// statements' is in a lighter blue italicized font. The closing curly brace '}' is also in a red box. Two red arrows originate from a red-bordered callout box at the bottom right, pointing to the opening and closing curly braces.

Curly braces to define
loop code block (body)

The while Statement Recap



The while Statement Recap

```
int count = 1;
while (count <= 5) {
    System.out.println("count = " + count);
    count++;
}
```

The diagram illustrates the structure of a while loop with three labeled components in red boxes: **init** points to the initialization `count = 1;`, **condition** points to the loop condition `count <= 5`, and **increment** points to the increment statement `count++`.

init **condition** **increment**

```
for (int i = 0; i <= 5; i++) {
    System.out.println("i = " + 1);
}
```

The diagram illustrates the structure of a for loop with three labeled components in red boxes: **init** points to the initialization `int i = 0`, **condition** points to the loop condition `i <= 5`, and **increment** points to the increment statement `i++`.

The while Statement Recap

```
int number = 0;
while (number < 15) {
    number++;

    if (number <= 5) {
        System.out.println("Skipping number " + number);
        continue;
    }

    System.out.println("number = " + number);

    if (number >= 10) {
        System.out.println("Breaking at " + number);
        break;
    }
}
```

1

init

2

check condition
0 < 15 is true

3

number = 1

The while Statement Recap

```
int number = 0;
while (number < 15) {
    number++;

    if (number <= 5) {
        System.out.println("Skipping number " + number);
        continue;
    }

    System.out.println("number = " + number);

    if (number >= 10) {
        System.out.println("Breaking at " + number);
        break;
    }
}
```

4

check condition
1 < 15 is true

1

check condition
1 <= 5 is true

2

execute code
block

3

Continue with a
loop (bypass all
other code in the
block/body)

The while Statement Recap

```
int number = 0;
while (number < 15) {
    number++;

    if (number <= 5) {
        System.out.println("Skipping number " + number);
        continue;
    }

    System.out.println("number = " + number);

    if (number >= 10) {
        System.out.println("Breaking at " + number);
        break;
    }
}
```

number = 5

1 check condition
5 < 15 is true

2 number = 6

3 check condition
6 <= 5 is false

4 execute code
block

5 check condition
6 >= 10 is false

The while Statement Recap

```
int number = 0;
while (number < 15) {
    number++;

    if (number <= 5) {
        System.out.println("Skipping number " + number);
        continue;
    }

    System.out.println("number = " + number);

    if (number >= 10) {
        System.out.println("Breaking at " + number);
        break;
    }
}
```

number = 9

1 check condition
9 < 15 is true

2 number = 10

3 check condition
10 <= 5 is false

4 execute code
block

5 check condition
10 >= 10 is true

6 execute code
then break exits
the loop

The while and the do while

Now firstly, the while loop checks the condition at the start, before executing the block.

Compare that to the do while loop, where the code is executed at least once, and then the condition is checked.

Examine loop conditions carefully

When using loops, you want to carefully examine the conditions for terminating, or continuing a loop.

Check for endless, or infinite loops.

Check for conditions where a loop will never execute.

Continue and Break

The continue and break statements both interrupt normal loop processing.

The continue statement starts a new iteration, but continues to iterate through the loop.

The break statement exits the loop, at the point it's executed, and no longer completes any code in the loop, and won't continue iterating any longer.