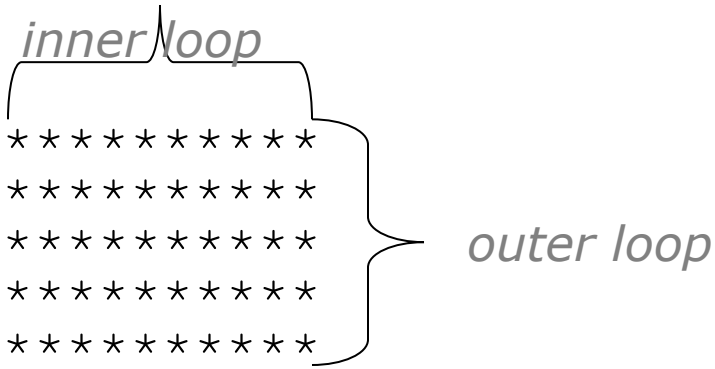
Nested `for` loops

Nested loops

- **nested loop**: A loop placed inside another loop.
- The nested loops are commonly referred to as the **inner loop** and **outer loop**.

```
for (int i = 1; i <= 5; i++) {
    for (int j = 1; j <= 10; j++) {
        System.out.print("*");
    }
    System.out.println();   // to end the line
}
```

- Output:



Nested for loop exercise

- What is output by the following code?

```
int row;
int col;

for(row = 2; row <= 3; row = row + 1) {
        for(col = 0; col <= 1; col = col + 1) {
                System.out.print("" + row + col + " ");
   }
  }
```

- Output: 20 21 30 31

Nested `for` loop exercise

- What is the output of the following nested `for` loops?

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

- Output:

```
*
**
***
****
*****
```

Nested `for` loop exercise

- What is the output of the following nested `for` loops?

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

- Output:

```
1
22
333
4444
55555
```

Common errors

- Both of the following sets of code produce *infinite loops*:

```
for (int i = 1; i <= 5; i++) {
    for (int j = 1; i <= 10; j++) {
        System.out.print("*");
    }
    System.out.println();
}
```

```
for (int i = 1; i <= 5; i++) {
    for (int j = 1; j <= 10; i++) {
        System.out.print("*");
    }
    System.out.println();
}
```

CHALLENGEACTIVITY
4.7.1: Nested loops: Indent text.

Print numbers 0, 1, 2, ..., userNum as shown, with each number indented by that number of spaces. For each printed line, print the leading spaces, then the number, and then a newline. Hint: Use i and j as loop variables (initialize i and j explicitly). Note: Avoid any other spaces like spaces after the printed number. Ex: userNum = 3 prints:

```
0
 1
  2
   3
```

```
import java.util.Scanner;
public class NestedLoop {
   public static void main (String [] args) {
      Scanner scnr = new Scanner(System.in);
      int userNum;
      int i;
      int j;

      userNum = scnr.nextInt();

   for (i = 0; i <= userNum; i++) {
   for (j = 0; j < i; j++) {
      System.out.print(" ");
   }
   System.out.println(i);
}

   }
}
```

4.7.2: Nested loops: Print seats.

Given numRows and numColumns, print a list of all seats in a theater. Rows are numbered, columns lettered, as in 1A or 3E. Print a space after each seat, including after the last. Use separate print statements to print the row and column.

Ex: numRows = 2 and numColumns = 3 prints:

1A 1B 1C 2A 2B 2C

```
import java.util.Scanner;
public class NestedLoops {
   public static void main (String [] args) {
      Scanner scnr = new Scanner(System.in);
      int numRows;
      int numColumns;
      int currentRow;
      int currentColumn;
      char currentColumnLetter;

      numRows = scnr.nextInt();
      numColumns = scnr.nextInt();

   for (currentRow = 1; currentRow <= numRows; currentRow =
currentRow + 1) {
   currentColumnLetter = 'A';
   for (currentColumn = 1; currentColumn <= numColumns;
currentColumn = currentColumn + 1) {
      System.out.print(currentRow);
      System.out.print(currentColumnLetter + " ");
      ++currentColumnLetter;
   }
}

      System.out.println("");
   }
}
```

Break and Continue

- A **break statement** in a loop causes an immediate exit of the loop.
- A **continue statement** in a loop causes an immediate jump to the loop condition check.
- A break statement or continue statement can sometimes yield a loop that is easier to understand.

```
public class Main {
  public static void main(String[] args) {
    for (int i = 0; i < 10; i++) {
      if (i == 4) {
        break;
      }
      System.out.println(i);
    }
  }
}
```

Outout:

0

1

2

3

```
public class Main {
  public static void main(String[] args) {
    for (int i = 0; i < 10; i++) {
      if (i == 4) {
        continue;
      }
      System.out.println(i);
    }
  }
}
```

Out put:

```
0
1
2
3
5
6
7
8
9
```