Crash Course: Repetition Structures 3

BONITATEM·ET·DISCIPLINAM·ET·SCIENTIAM·DOCE·ME

1885

ST. MARY'S HIGH SCHOOL

In this crash course

- Break statements and infinite loops
  - Example 9
  - Example 10
- Break vs. Continue
  - Example 11
- Nested loops
  - Example 12
  - Example 13
  - Example 14
- Summary

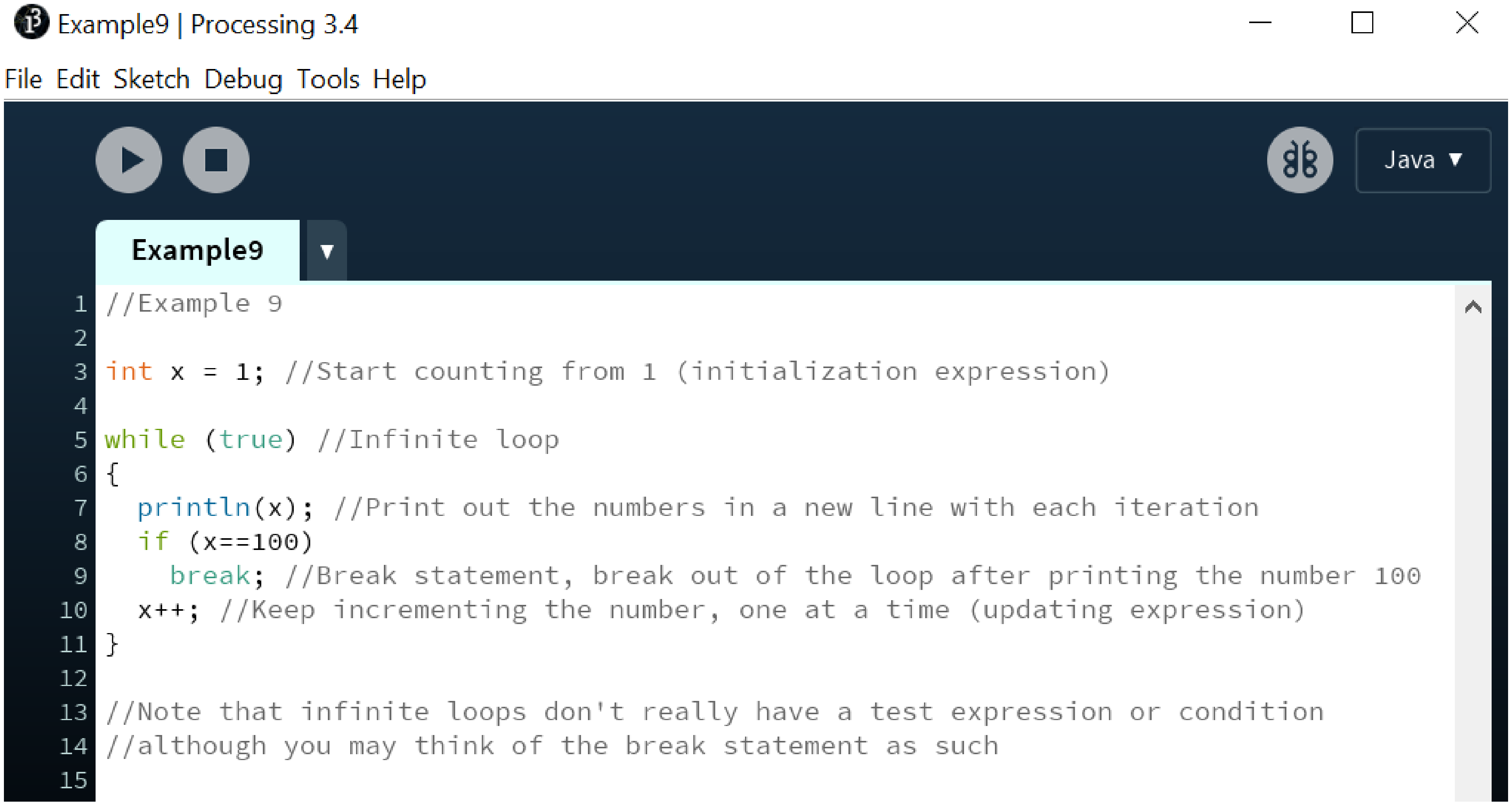
Break statements and infinite loops

- An infinite loop is one in which the test expression or condition is never met, so the loop keeps going indefinitely
- These are usually introduced with the statement “while(true)”
- This is generally not a good coding practice so we introduce break statements that “break out of the loop”
- A break statement is essentially a condition that will force your program to exit the loop

Break statements and infinite loops

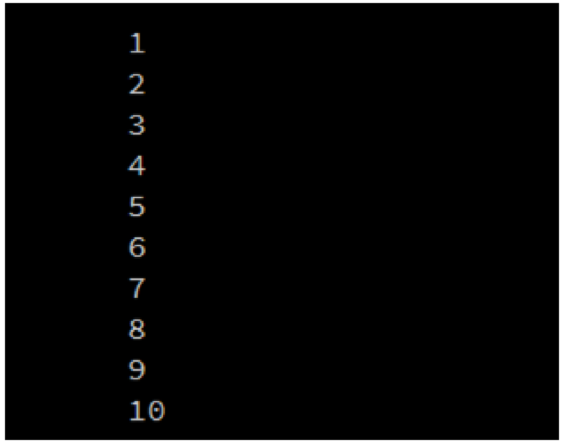
- Let’s say you have a program that starts at 1 and just keeps counting up from there
- How can we make it stop when it gets to 100?

Break statements and infinite loops – Example 9



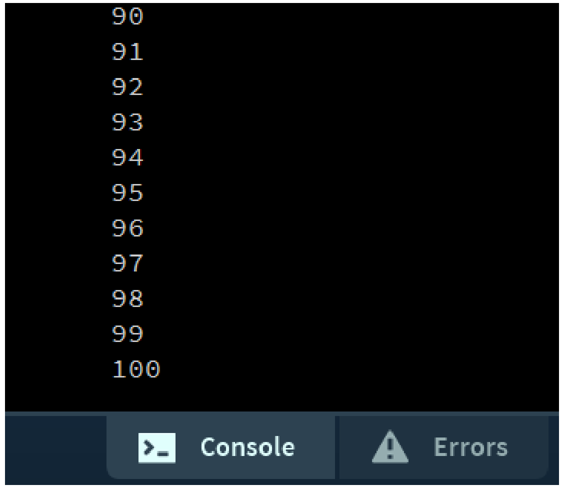
```
//Example 9

int x = 1; //Start counting from 1 (initialization expression)

while (true) //Infinite loop
{
  println(x); //Print out the numbers in a new line with each iteration
  if (x==100)
    break; //Break statement, break out of the loop after printing the number 100
  x++; //Keep incrementing the number, one at a time (updating expression)
}

//Note that infinite loops don't really have a test expression or condition
//although you may think of the break statement as such

```

Break statements and infinite loops – Example 9



```
1
2
3
4
5
6
7
8
9
10
```

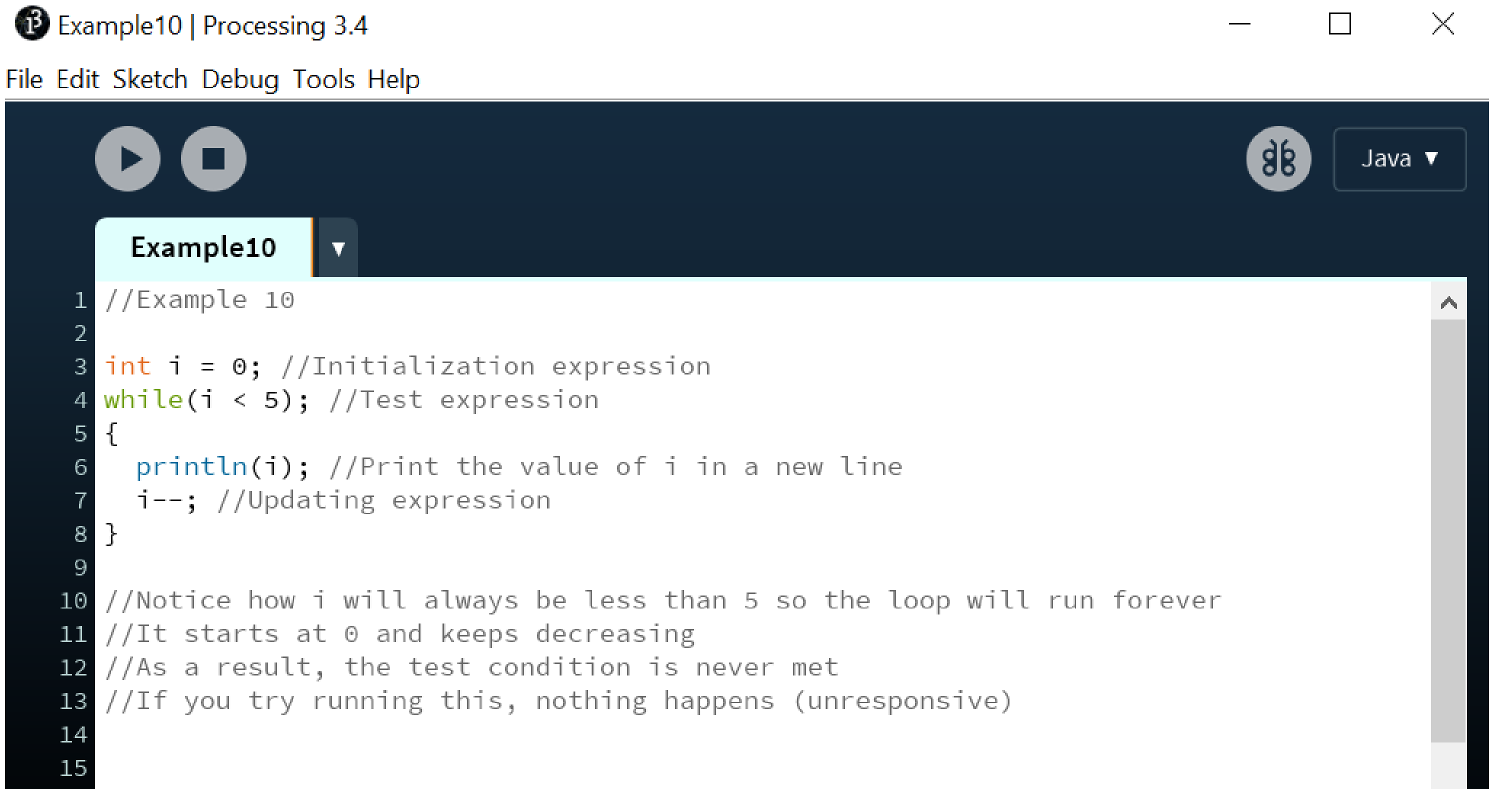
⋮



```
90
91
92
93
94
95
96
97
98
99
100
```

Break statements and infinite loops – Example 10

- Let’s see one more example of an infinite loop where the test expression or condition is never met



```
//Example 10

int i = 0; //Initialization expression
while(i < 5); //Test expression
{
  println(i); //Print the value of i in a new line
  i--; //Updating expression
}

//Notice how i will always be less than 5 so the loop will run forever
//It starts at 0 and keeps decreasing
//As a result, the test condition is never met
//If you try running this, nothing happens (unresponsive)
```

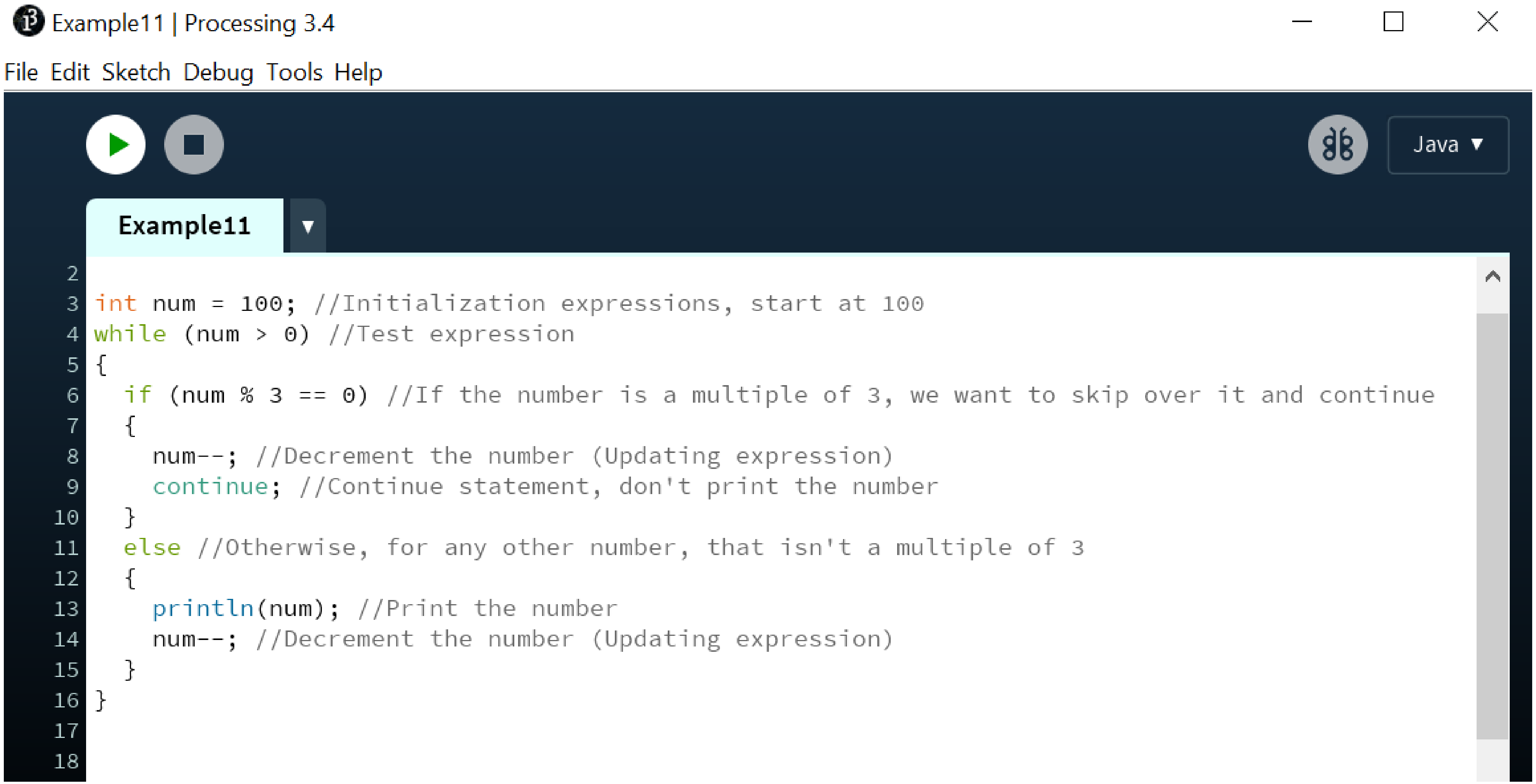
Break vs. Continue

- But what if you don’t want to break out the loop completely?
- What if you just want to skip one of the iterations?
- To accomplish this, we can use a continue statement

Break vs. Continue – Example 11

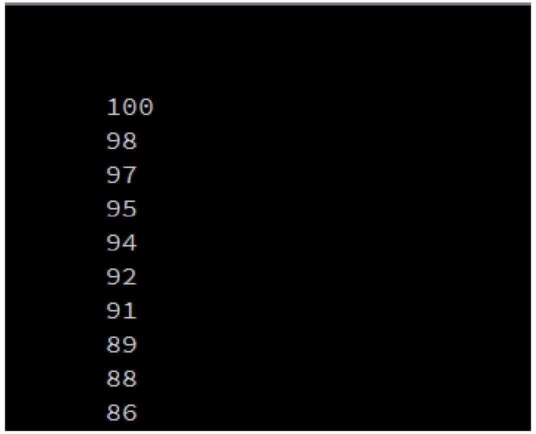
- Imagine you need to write code that will count down the digits from 100 to 1, printing them out as you go but skipping all numbers that are divisible by 3, i.e. multiples of 3
- You can skip these with a continue statement

Break vs. Continue – Example 11



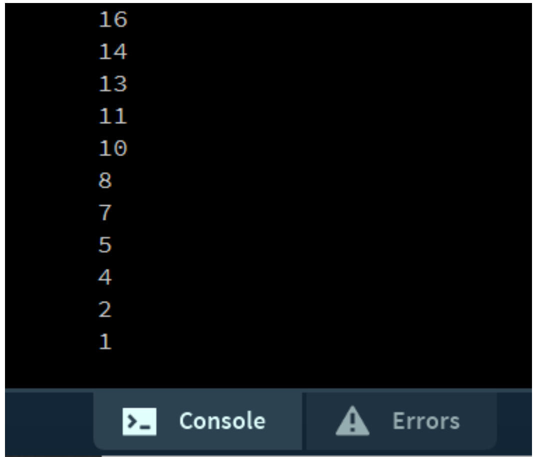
```

int num = 100; //Initialization expressions, start at 100
while (num > 0) //Test expression
{
  if (num % 3 == 0) //If the number is a multiple of 3, we want to skip over it and continue
  {
    num--; //Decrement the number (Updating expression)
    continue; //Continue statement, don't print the number
  }
  else //Otherwise, for any other number, that isn't a multiple of 3
  {
    println(num); //Print the number
    num--; //Decrement the number (Updating expression)
  }
}


```

Break vs. Continue – Example 11



```
100
98
97
95
94
92
91
89
88
86
```

⋮



```
16
14
13
11
10
8
7
5
4
2
1
```

Nested loops

- Another very important topic is nested loops
- Up until now, we've just been working with simple 1D loops
- A nested loop is when you have one loop inside of another (an inner loop and an outer loop)
- These are very useful when you're working in higher dimensions such as 2D, where you would require two variables instead of one (e.g. i and j instead of just i)

Nested loops

- Nested loops are commonly used for accessing data in 2D data structures such as 2D arrays (more on arrays in a different crash course)
- You can also apply the concept of nested loops to higher dimensions if necessary but we’ll stick to the 2D cases

Nested loops – Example 12

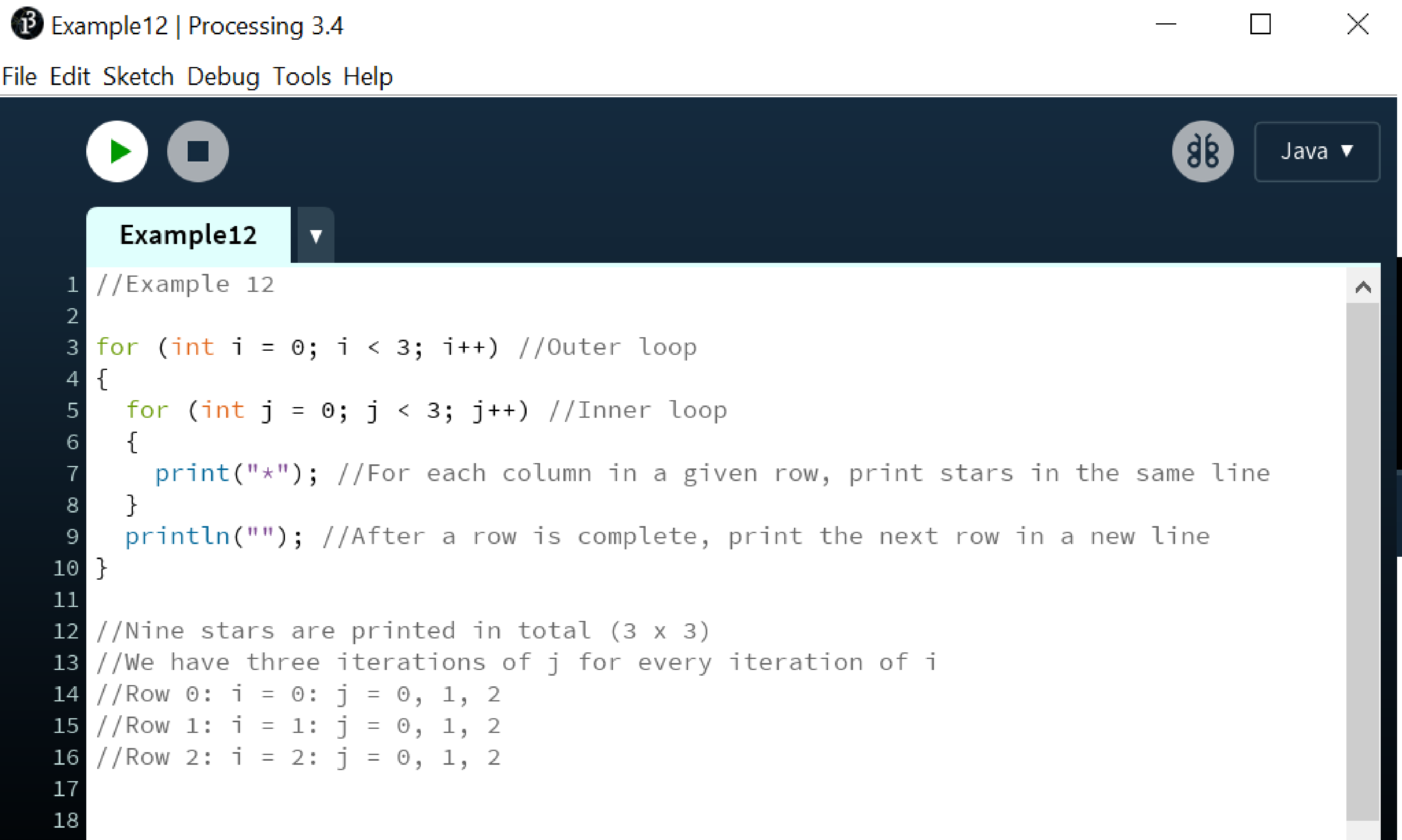
- Using nested loops, write a program that prints out 9 asterisks (*) to the console (3 rows of 3 columns)
- Before coding, plan this out on blank, indexed grid
- Let i be the row number from 0 to 2
- Let j be the column number from 0 to 2

Nested loops – Example 12

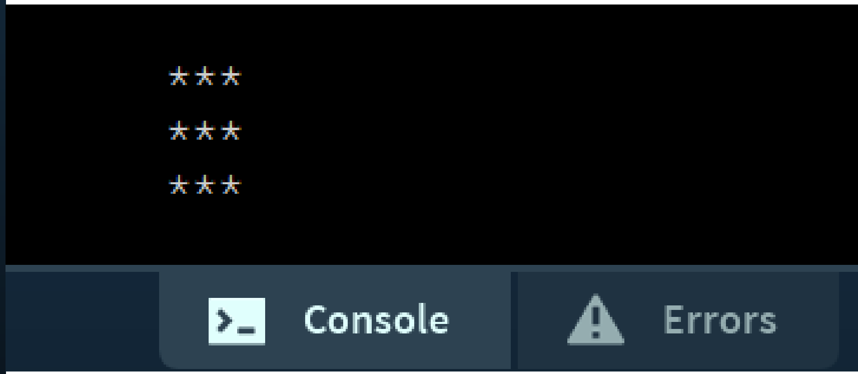
		j	
	(0, 0)	(0, 1)	(0, 2)
i	(1, 0)	(1, 1)	(1, 2)
	(2, 0)	(2, 1)	(2, 2)

		j	
	*	*	*
i	*	*	*
	*	*	*

Nested loops – Example 12



```
//Example 12

for (int i = 0; i < 3; i++) //Outer loop
{
  for (int j = 0; j < 3; j++) //Inner loop
  {
    print("*"); //For each column in a given row, print stars in the same line
  }
  println(""); //After a row is complete, print the next row in a new line
}

//Nine stars are printed in total (3 x 3)
//We have three iterations of j for every iteration of i
//Row 0: i = 0: j = 0, 1, 2
//Row 1: i = 1: j = 0, 1, 2
//Row 2: i = 2: j = 0, 1, 2
```

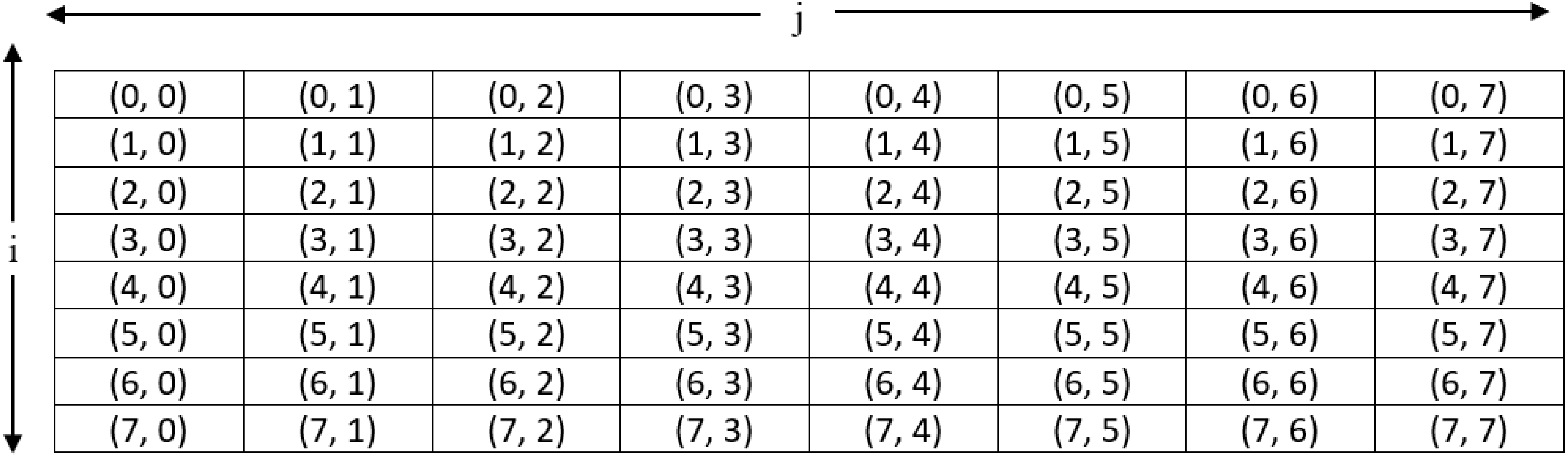


```
***
***
***
```

Nested loops – Example 13

- Write a program that uses nested loops to draw a chess/checker board with alternating black and white squares
- The board should be an 8 x 8 with squares of 100 x 100 units starting with a black square in the top left corner
- Before coding, plan this out on blank, indexed grid
- Let i be the row number from 0 to 7
- Let j be the column number from 0 to 7

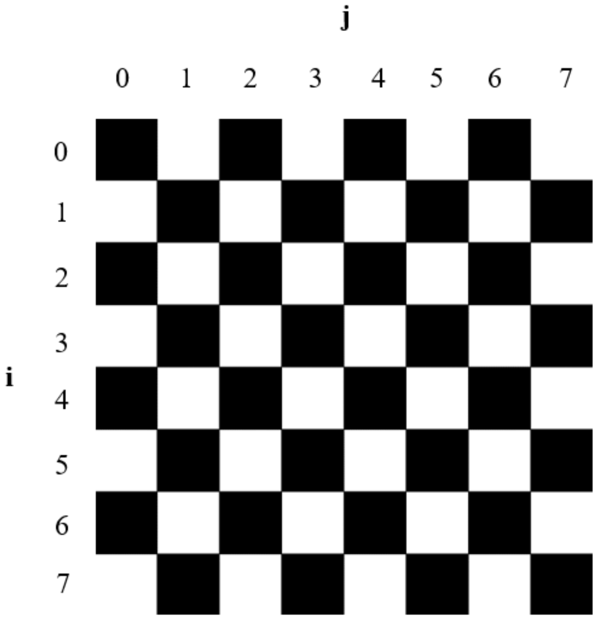
Nested loops – Example 13



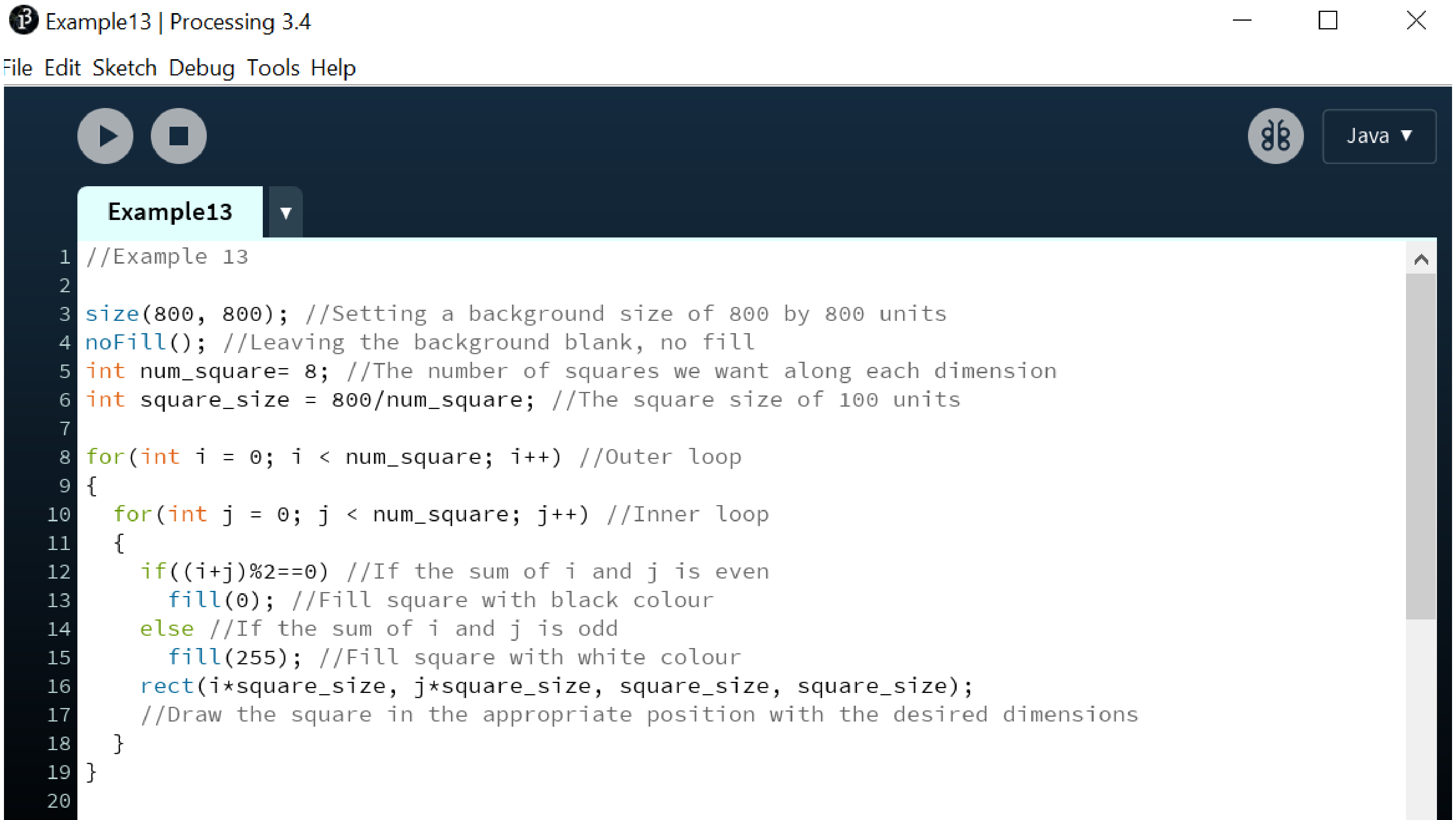
(0, 0)	(0, 1)	(0, 2)	(0, 3)	(0, 4)	(0, 5)	(0, 6)	(0, 7)
(1, 0)	(1, 1)	(1, 2)	(1, 3)	(1, 4)	(1, 5)	(1, 6)	(1, 7)
(2, 0)	(2, 1)	(2, 2)	(2, 3)	(2, 4)	(2, 5)	(2, 6)	(2, 7)
(3, 0)	(3, 1)	(3, 2)	(3, 3)	(3, 4)	(3, 5)	(3, 6)	(3, 7)
(4, 0)	(4, 1)	(4, 2)	(4, 3)	(4, 4)	(4, 5)	(4, 6)	(4, 7)
(5, 0)	(5, 1)	(5, 2)	(5, 3)	(5, 4)	(5, 5)	(5, 6)	(5, 7)
(6, 0)	(6, 1)	(6, 2)	(6, 3)	(6, 4)	(6, 5)	(6, 6)	(6, 7)
(7, 0)	(7, 1)	(7, 2)	(7, 3)	(7, 4)	(7, 5)	(7, 6)	(7, 7)

Nested loops – Example 13

- Observe how the cells that are black are the ones where i + j is an even number (divisible by 2)
- Notice how the cells that are white are the ones where i + j is an odd number (not divisible by 2)

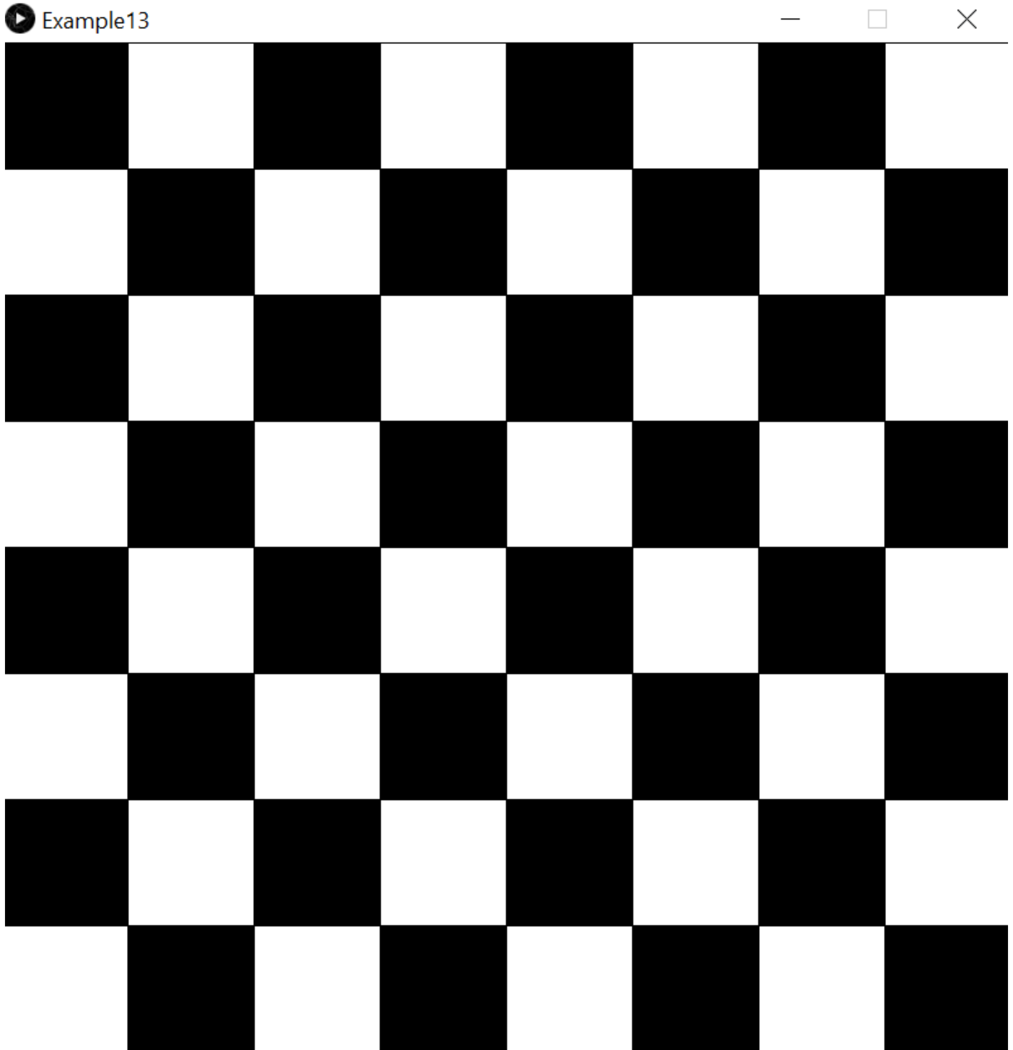


Nested loops – Example 13



```
//Example 13

size(800, 800); //Setting a background size of 800 by 800 units
noFill(); //Leaving the background blank, no fill
int num_square= 8; //The number of squares we want along each dimension
int square_size = 800/num_square; //The square size of 100 units

for(int i = 0; i < num_square; i++) //Outer loop
{
  for(int j = 0; j < num_square; j++) //Inner loop
  {
    if((i+j)%2==0) //If the sum of i and j is even
      fill(0); //Fill square with black colour
    else //If the sum of i and j is odd
      fill(255); //Fill square with white colour
    rect(i*square_size, j*square_size, square_size, square_size);
    //Draw the square in the appropriate position with the desired dimensions
  }
}

```

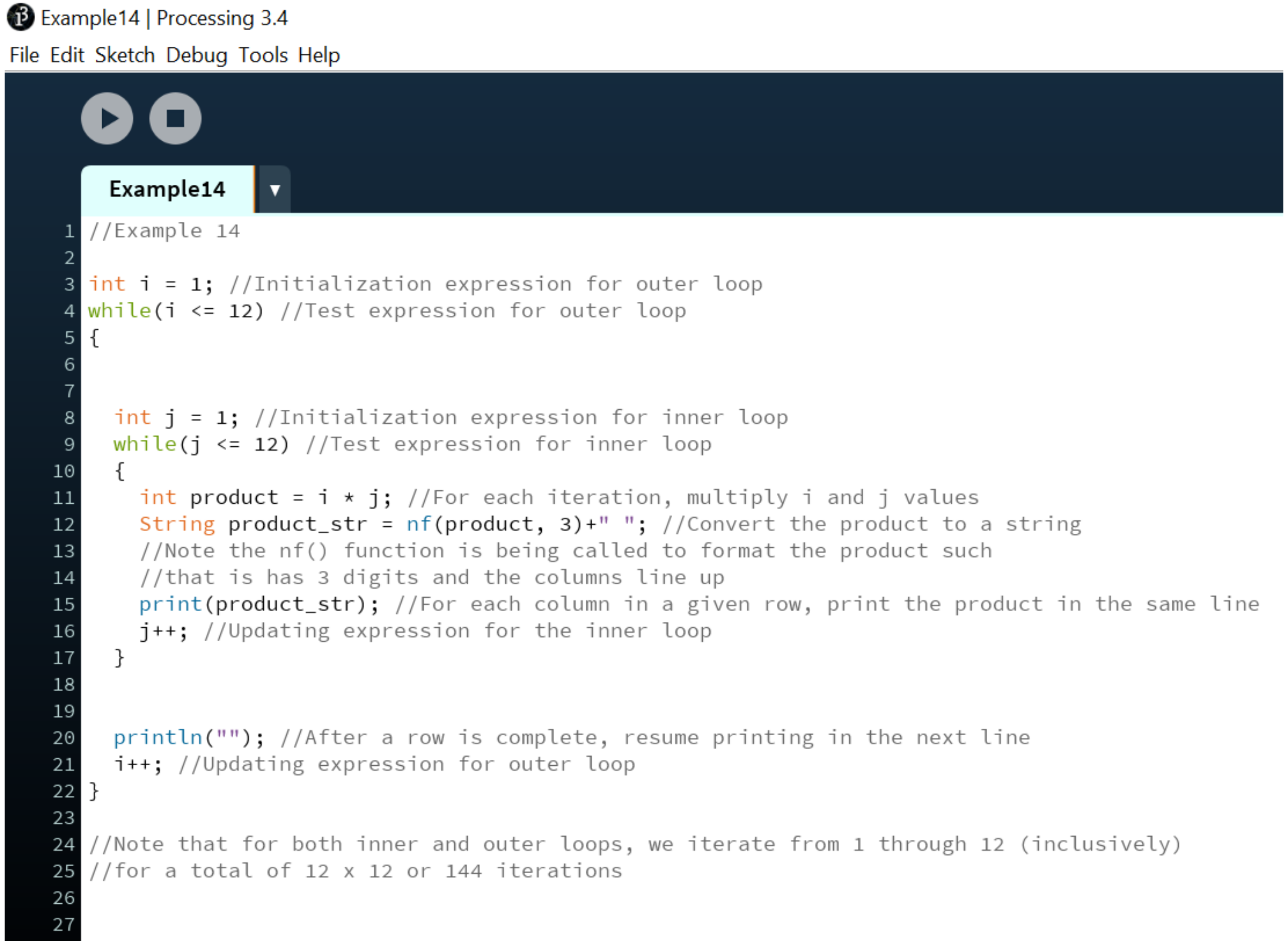
Nested loops – Example 13



Nested loops – Example 14

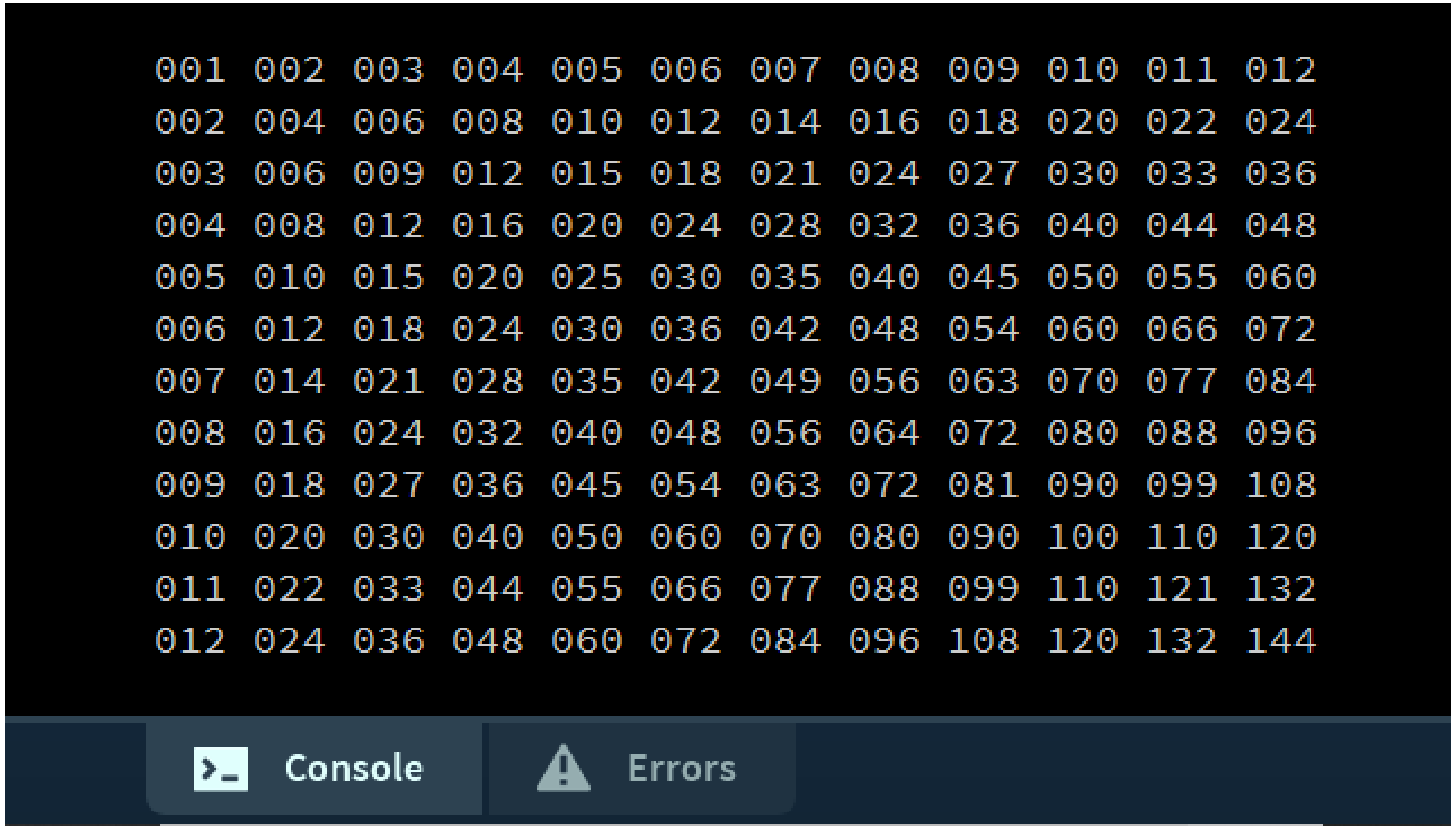
- Let’s do one more example
- Use nested loops to print out a 12 x 12 multiplication table like you’ve probably seen in elementary school
- This time, we’ll use nested while loops instead of nested for loops to change things up

Nested loops – Example 14



```
//Example 14

int i = 1; //Initialization expression for outer loop
while(i <= 12) //Test expression for outer loop
{


  int j = 1; //Initialization expression for inner loop
  while(j <= 12) //Test expression for inner loop
  {
    int product = i * j; //For each iteration, multiply i and j values
    String product_str = nf(product, 3)+" "; //Convert the product to a string
    //Note the nf() function is being called to format the product such
    //that is has 3 digits and the columns line up
    print(product_str); //For each column in a given row, print the product in the same line
    j++; //Updating expression for the inner loop
  }


  println(""); //After a row is complete, resume printing in the next line
  i++; //Updating expression for outer loop
}

//Note that for both inner and outer loops, we iterate from 1 through 12 (inclusively)
//for a total of 12 x 12 or 144 iterations
```

Nested loops – Example 14



```
001 002 003 004 005 006 007 008 009 010 011 012
002 004 006 008 010 012 014 016 018 020 022 024
003 006 009 012 015 018 021 024 027 030 033 036
004 008 012 016 020 024 028 032 036 040 044 048
005 010 015 020 025 030 035 040 045 050 055 060
006 012 018 024 030 036 042 048 054 060 066 072
007 014 021 028 035 042 049 056 063 070 077 084
008 016 024 032 040 048 056 064 072 080 088 096
009 018 027 036 045 054 063 072 081 090 099 108
010 020 030 040 050 060 070 080 090 100 110 120
011 022 033 044 055 066 077 088 099 110 121 132
012 024 036 048 060 072 084 096 108 120 132 144
```

Summary

- In this crash course, we learned about break statements and infinite loops
- Infinite loops go on forever because either there is no test expression or the condition being tested is never met
- Break statements can be used to exit loops completely and stop their execution
- Continue statements can be used to skip or jump over certain iterations within the loop
- We also explored nested loops that are used in higher dimension situations, e.g. in 2D where there is an inner loop and an outer loop