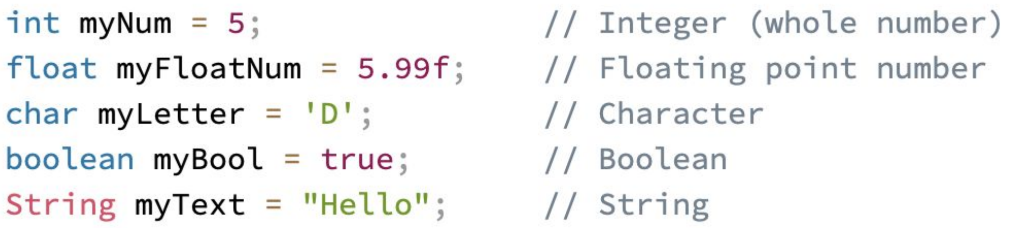
Java Bootcamp

Session #2

The DA Programming Club

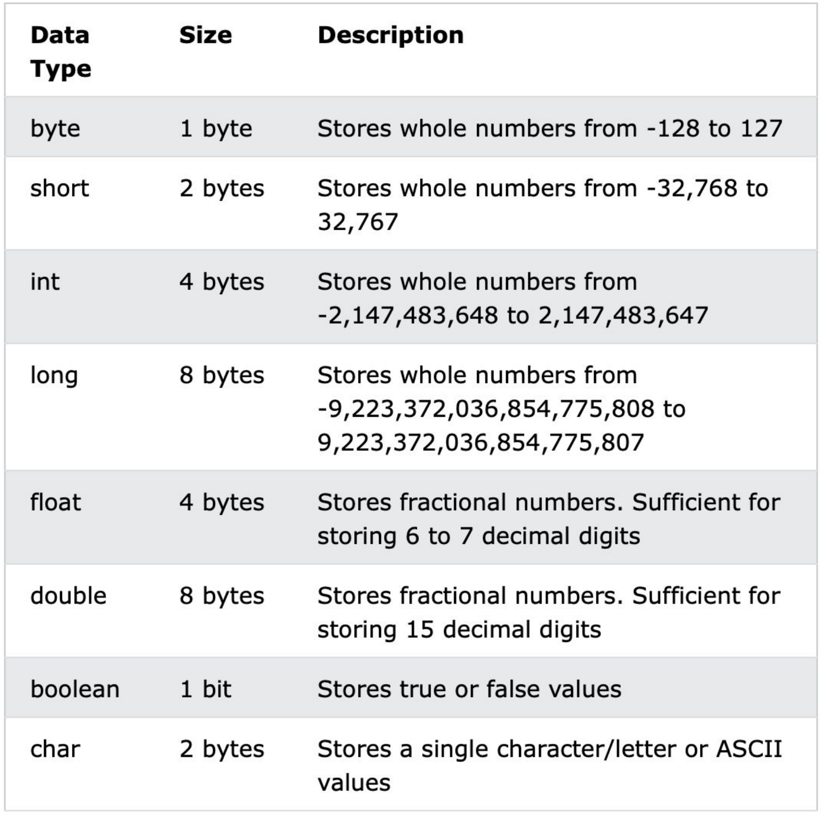
Quick Recap

- Java is a strong-typed, compiled language
- Common Java data types include:



```
int myNum = 5;               // Integer (whole number)
float myFloatNum = 5.99f;    // Floating point number
char myLetter = 'D';         // Character
boolean myBool = true;       // Boolean
String myText = "Hello";     // String
```

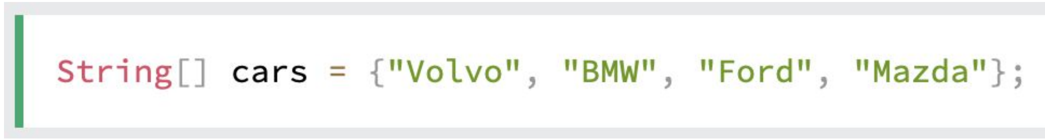
- There are eight primitive data types in Java —>



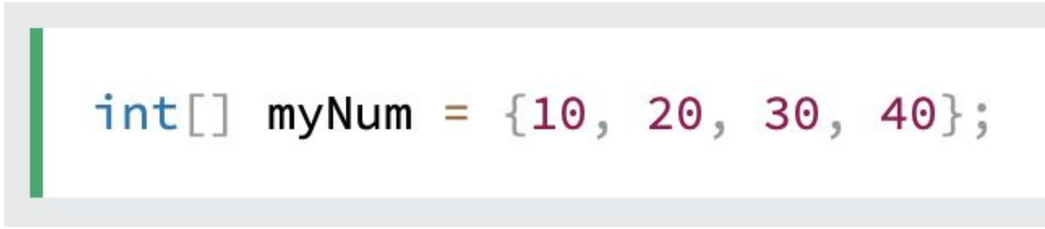
Data Type	Size	Description
byte	1 byte	Stores whole numbers from -128 to 127
short	2 bytes	Stores whole numbers from -32,768 to 32,767
int	4 bytes	Stores whole numbers from -2,147,483,648 to 2,147,483,647
long	8 bytes	Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	4 bytes	Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits
double	8 bytes	Stores fractional numbers. Sufficient for storing 15 decimal digits
boolean	1 bit	Stores true or false values
char	2 bytes	Stores a single character/letter or ASCII values

Quick Recap: Strings and Arrays

```java
// Declare String without using new operator
String s = "GeeksforGeeks";

// Prints the String.
System.out.println("String s = " + s);

// Declare String using new operator
String s1 = new String("GeeksforGeeks");

// Prints the String.
System.out.println("String s1 = " + s1);
```



```java
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
```

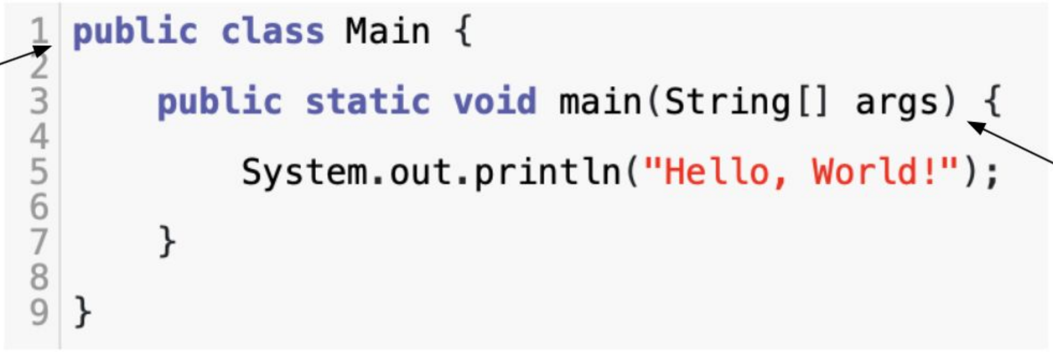


```java
int[] myNum = {10, 20, 30, 40};
```

Quick Recap

- A basic Java class that can run code looks like:

In Java, every line of code that can actually run needs to be inside a class.

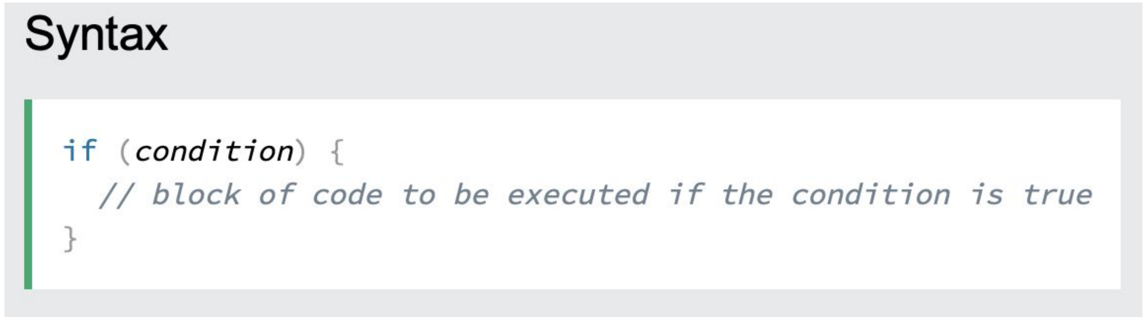


```java
public class Main {

    public static void main(String[] args) {

        System.out.println("Hello, World!");

    }

}
```

This is the entry point of our Java program. the main method has to have this exact signature in order to be able to run our program.

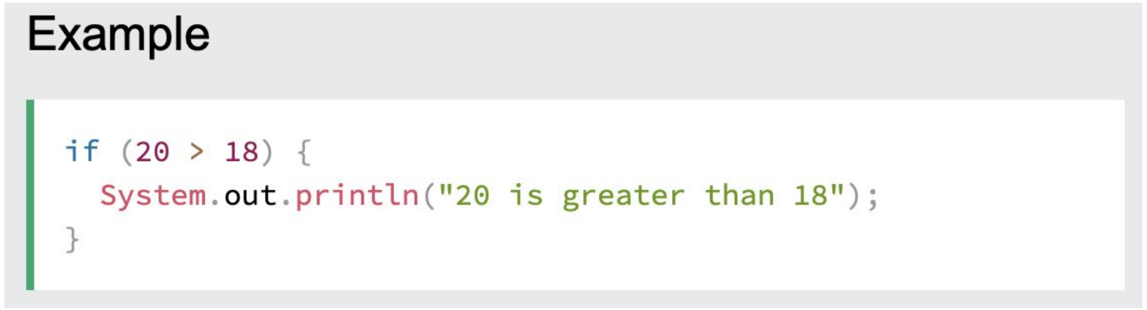
Conditionals

- A conditional statement consists of a condition and a task.



Syntax

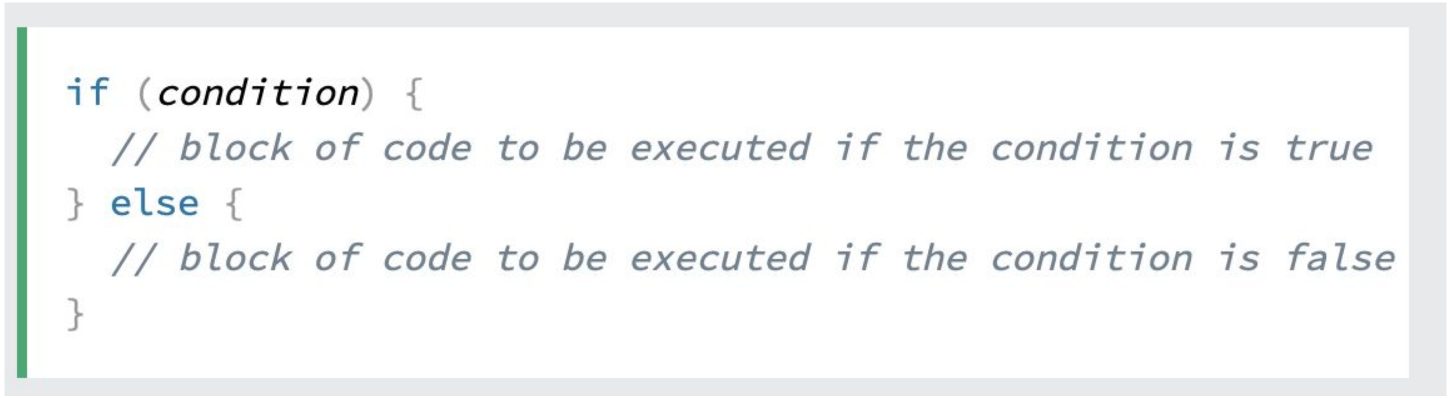
```
if (condition) {
  // block of code to be executed if the condition is true
}
```



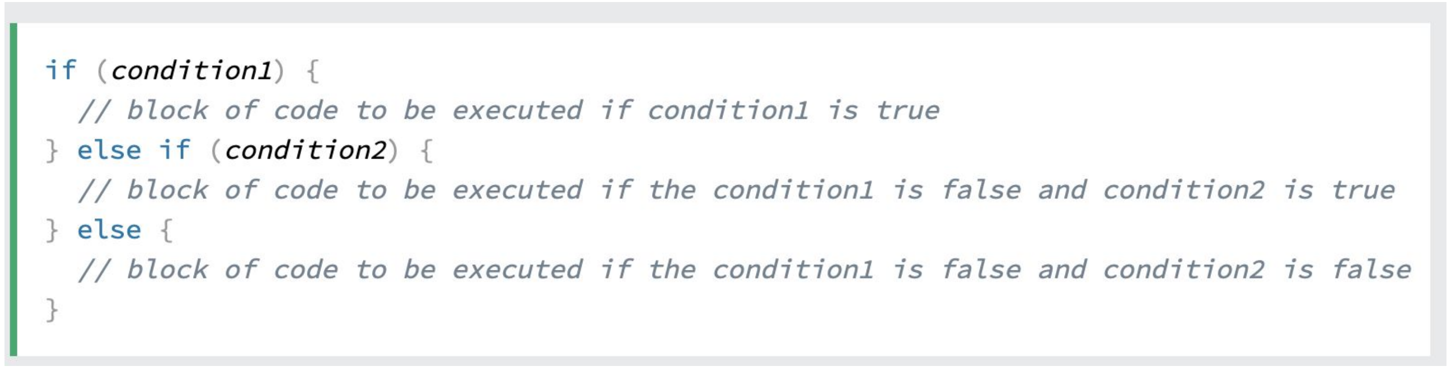
Example

```
if (20 > 18) {
  System.out.println("20 is greater than 18");
}
```

Conditionals



```
if (condition) {
  // block of code to be executed if the condition is true
} else {
  // block of code to be executed if the condition is false
}
```



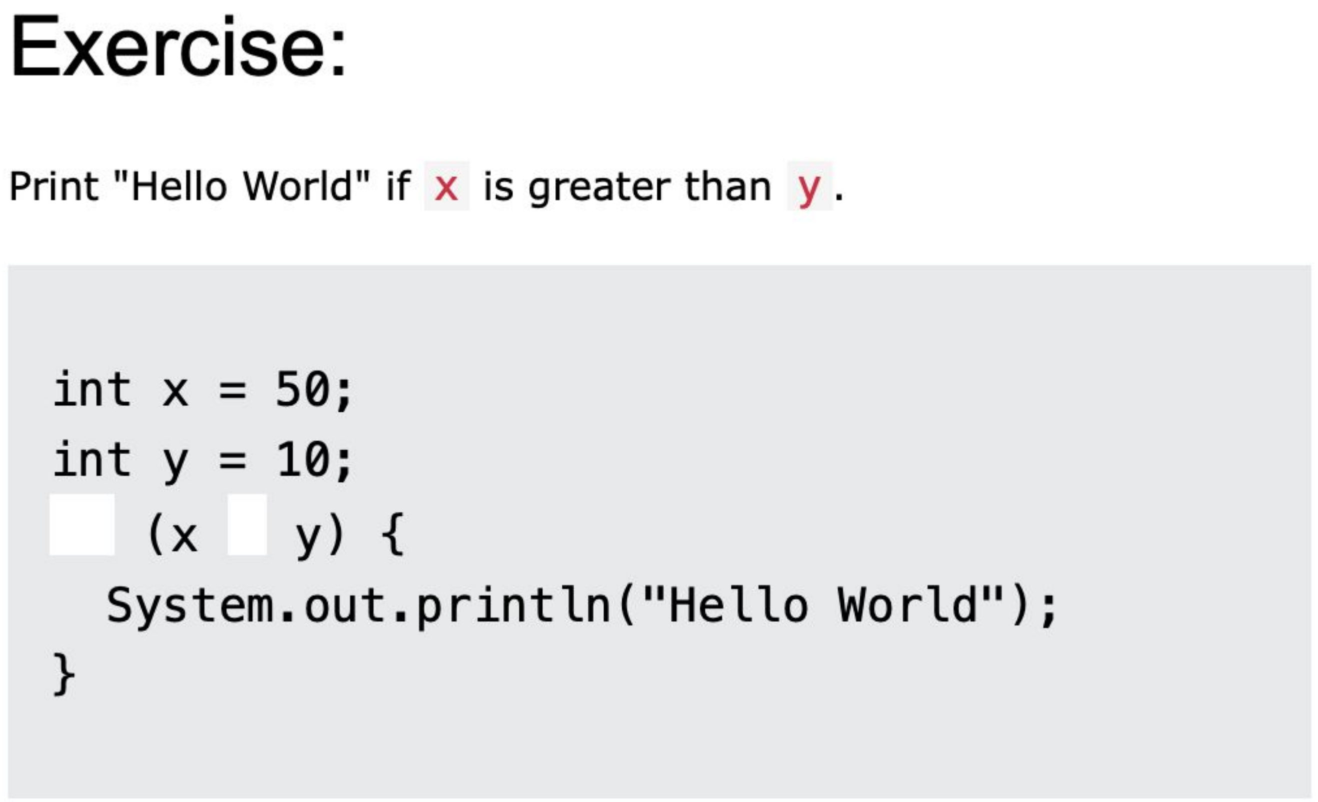
```
if (condition1) {
  // block of code to be executed if condition1 is true
} else if (condition2) {
  // block of code to be executed if the condition1 is false and condition2 is true
} else {
  // block of code to be executed if the condition1 is false and condition2 is false
}
```

Logical Conditions

Java supports the usual logical conditions from mathematics:

- Less than: `a < b`
- Less than or equal to: `a <= b`
- Greater than: `a > b`
- Greater than or equal to: `a >= b`
- Equal to `a == b`
- Not Equal to: `a != b`

Try it out!



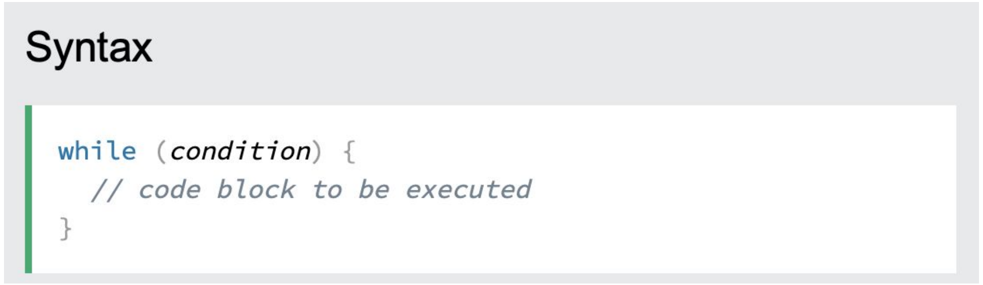
Exercise:

Print "Hello World" if `x` is greater than `y`.

```
int x = 50;
int y = 10;
     (x    y) {
  System.out.println("Hello World");
}
```

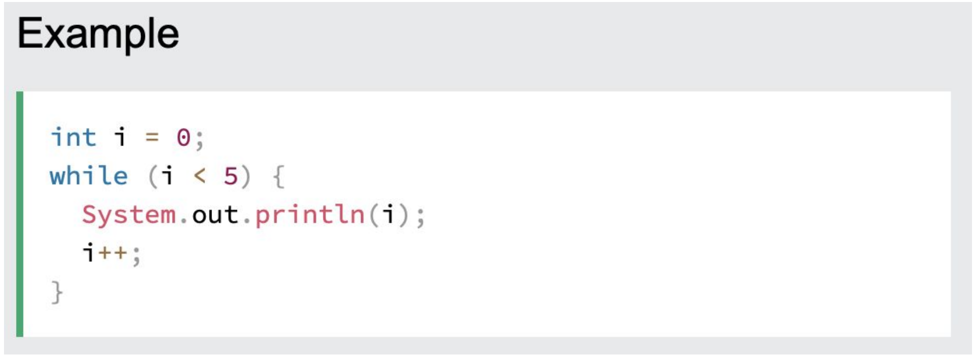
Iteration: While Loop

Loops can execute a block of code as long as a specified condition is reached. They save time, reduce errors, and make code more readable.



Syntax

```
while (condition) {
  // code block to be executed
}
```

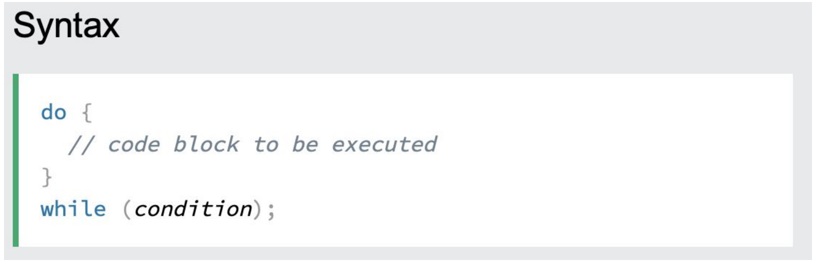


Example

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

Iteration: Do/While Loop

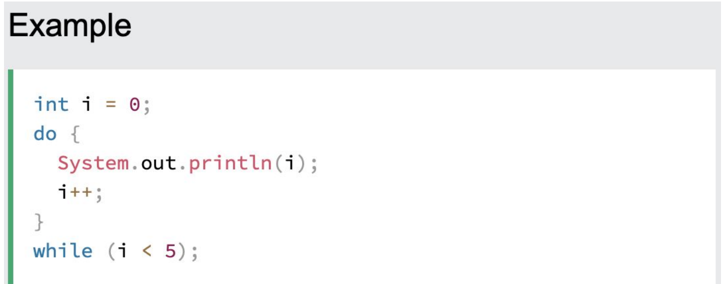
The `do/while` loop is a variant of the `while` loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.



Syntax

```
do {
  // code block to be executed
}
while (condition);
```

The example below uses a `do/while` loop. The loop will always be executed at least once, even if the condition is false, because the code block is executed before the condition is tested:

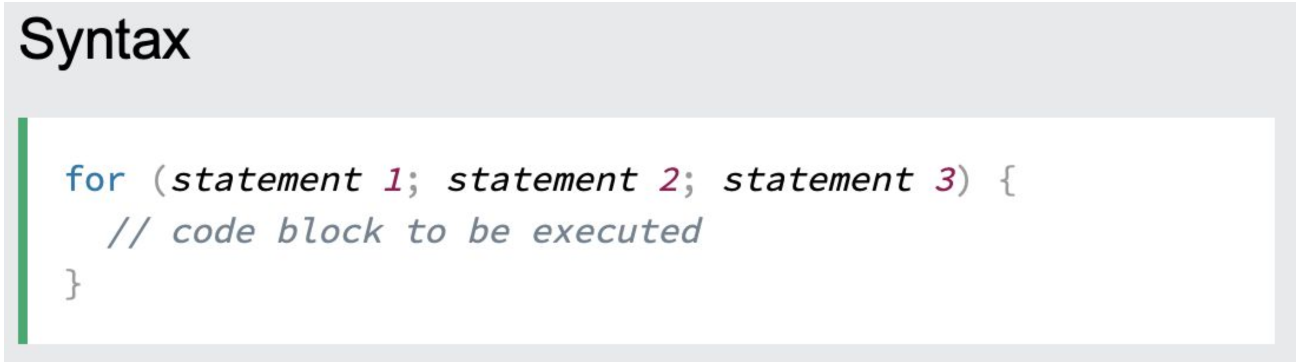


Example

```
int i = 0;
do {
  System.out.println(i);
  i++;
}
while (i < 5);
```

Iteration: For Loop

When you know exactly how many times you want to loop through a block of code, use the `for` loop instead of a `while` loop:


Syntax

```
for (statement 1; statement 2; statement 3) {
  // code block to be executed
}
```

Statement 1 is executed (one time) before the execution of the code block.

Statement 2 defines the condition for executing the code block.

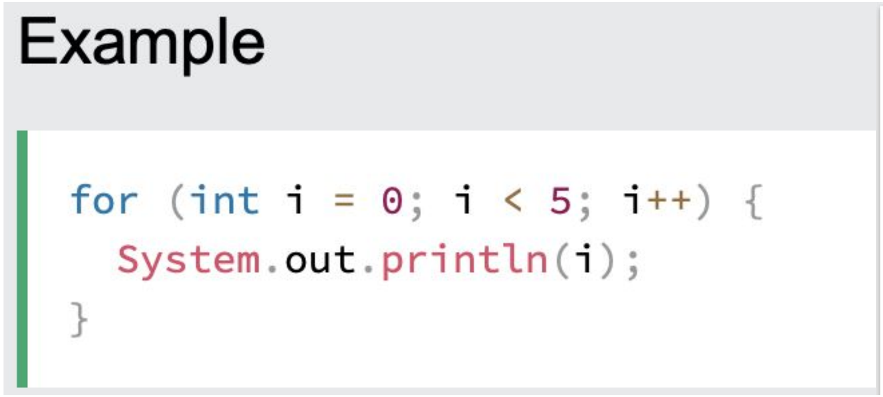
Statement 3 is executed (every time) after the code block has been executed.

Iteration: For Loop

Statement 1 sets a variable before the loop starts (int i = 0).

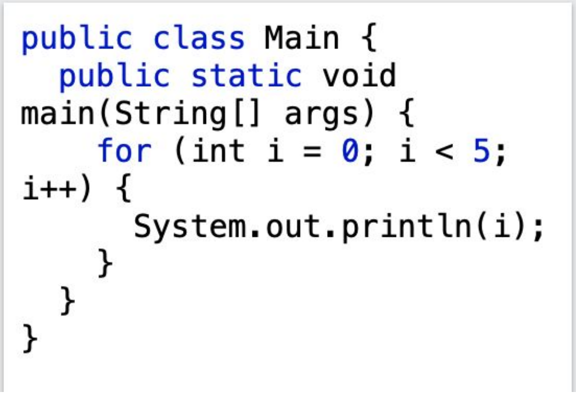
Statement 2 defines the condition for the loop to run (i must be less than 5). If the condition is true, the loop will start over again, if it is false, the loop will end.

Statement 3 increases a value (i++) each time the code block in the loop has been executed.

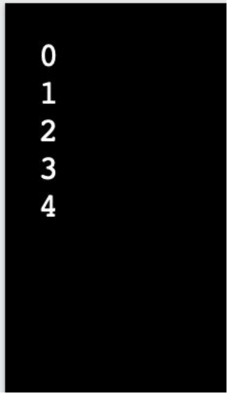


Example

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



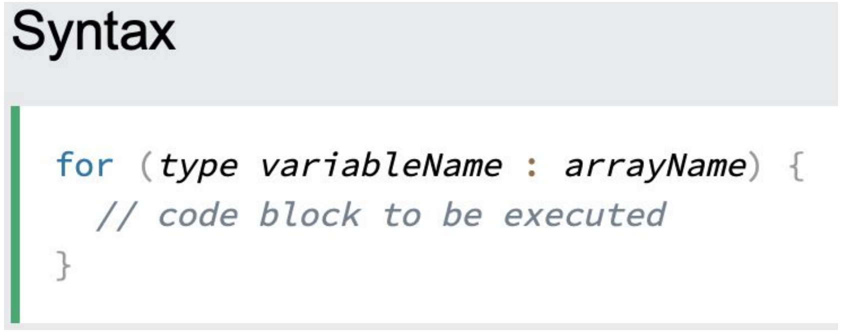
```java
public class Main {
  public static void
main(String[] args) {
    for (int i = 0; i < 5;
i++) {
      System.out.println(i);
    }
  }
}
```



```
0
1
2
3
4
```

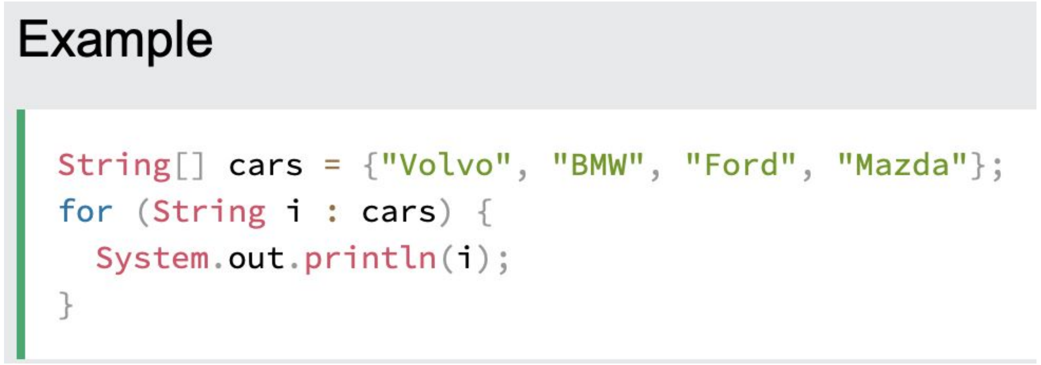
Iteration: For-Each Loop

There is also a "for-each" loop, which is used exclusively to loop through elements in an array:



Syntax

```
for (type variableName : arrayName) {
  // code block to be executed
}
```

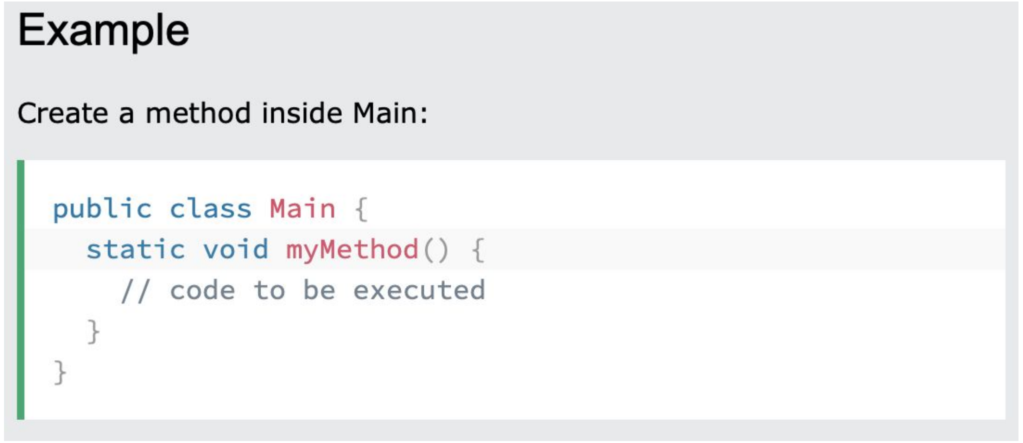


Example

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
for (String i : cars) {
  System.out.println(i);
}
```

Java Methods (aka Functions)

A method is a block of code which only runs when it is called. You can pass data, known as parameters, into a method.

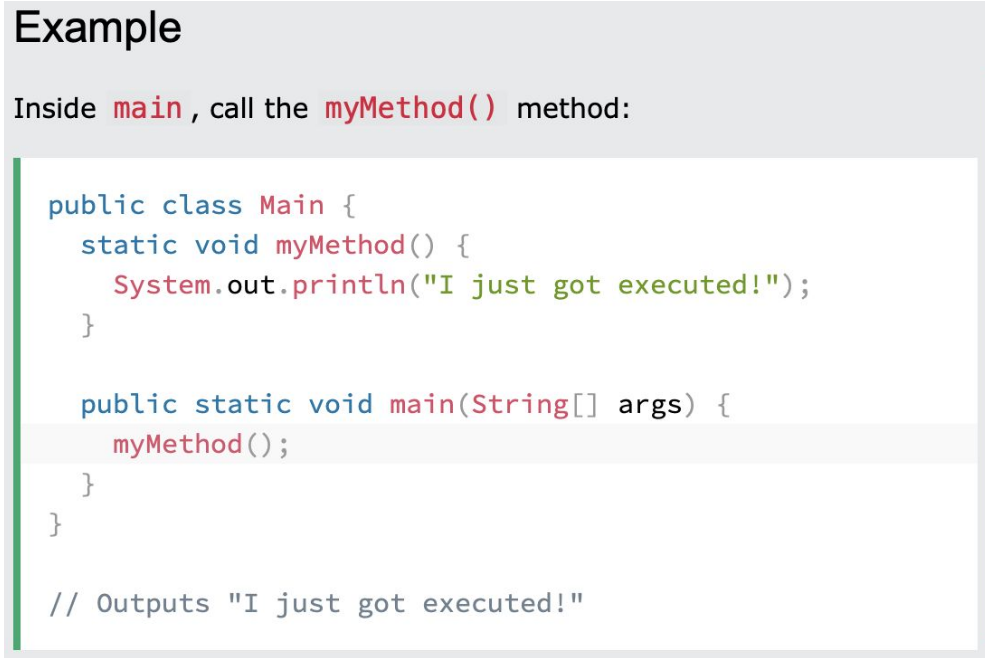

Example

Create a method inside Main:

```
public class Main {
  static void myMethod() {
    // code to be executed
  }
}
```

- `myMethod()` is the name of the method
- `static` means that the method belongs to the Main class and not an object of the Main class.
- `void` means that this method does not have a return value.

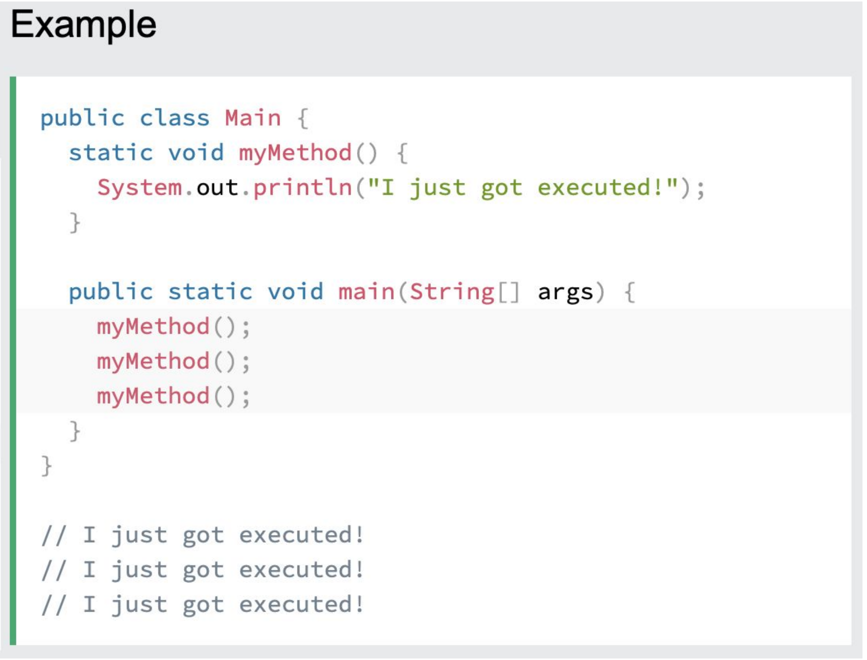
Java Methods (aka Functions)



Example

Inside `main`, call the `myMethod()` method:

```java
public class Main {
  static void myMethod() {
    System.out.println("I just got executed!");
  }

  public static void main(String[] args) {
    myMethod();
  }
}

// Outputs "I just got executed!"
```



Example

```java
public class Main {
  static void myMethod() {
    System.out.println("I just got executed!");
  }

  public static void main(String[] args) {
    myMethod();
    myMethod();
    myMethod();
  }
}

// I just got executed!
// I just got executed!
// I just got executed!
```

Write your own program!

- Create a method that uses a conditional and loop to determine what to output to the terminal!
- Here's some ideas to get you started:
  - Assess whether an array of numbers is odd or even and output the results
  - Assess whether two strings match or have the same number of characters