Methods

A byte size lesson in Java programming.

What is a method?

- A block of code that performs a specific task
  - User-defined methods – ones we create.
  - Standard Library methods – built-in and available to use

Here is an example of a standard library method from the Math class

```
// using the sqrt() method
System.out.print("Square root of 4 is: " + Math.sqrt(4));
```

User-defined Methods

- We declare a method using the following syntax:

```
returnType methodName() {
  // method body
}
```

- The **return type** is a type of value that the method comes back with.
- The **method name** is the identifier we will use to call the method.
- The **method body** is where we put the code for what the method does.

Let’s test your understanding!

- Answer the questions about this method

```
String sayHello(){
   return “Hello”;
}
```

What name will we use to call this method?

What data type will the method return?

Using method parameters

- In most cases, we need to pass information to the method to work with. Say we are creating a method that adds two numbers:

```
int addNumbers(int num1, int num2) {
  int addition = num1 + num2;
  return addition;
}
```

In this case we specify that the method **addNumbers()** needs two parameters **num1** and **num2**

Calling methods

Defining methods is all well and good, but how do we call them?

```
int addNumbers(int num1, int num2) {
  int addition = num1 + num2;
  return addition;
}
```

Here's how we call the above method:

```
int myAddition = addNumbers(2,5);
```

Let’s test your understanding!

Complete the following method definition:

```
int subtractNumbers(int num1, int num2) {
  int subtraction = ____________;
  return __________ ;
}
```

And this call to subtract 7 from 10:

```
int mySubtraction = ____________;
```

Methods that do not return a value

Sometimes a method does what it needs to without returning a value.

```
void sayHello(){
  System.out.println(“Hello”);
}
```

We use the **void** keyword instead of the type

Static methods

If we use the static keyword, it can be accessed without creating objects:

```
class Greeting {
  static void sayHello(){
   System.out.println(“Hello”);
}
```

In this case we call the method on the class

```
Greeting.sayHello();
```

```
class Greeting {
  void sayHello(){
   System.out.println(“Hello”);
}
```

In this case we need an object instance

```
Greeting g = new Greeting()
g.sayHello();
```

We will see this later in the course!

Let’s test your understanding!

Complete the following method definition of a method that

- says hello followed by the name passed in as a parameter
- does not return anything
- is able to be called on the class itself without needing an object instance.

```
class Greeting{
                 sayHello(               ) {
      System.out.println(                 );
  }
}
```