Access Methods with an Object

Example:

Create a Car object named myCar. Call the fullThrottle() and speed() methods on the myCar object, and run the program:

```
// Create a Main class
public class Main {

  // Create a fullThrottle() method
  public void fullThrottle() {
    System.out.println("The car is going as fast as it can!");
  }

  // Create a speed() method and add a parameter
  public void speed(int maxSpeed) {
    System.out.println("Max speed is: " + maxSpeed);
  }

  // Inside main, call the methods on the myCar object
  public static void main(String[] args) {
    Main myCar = new Main();     // Create a myCar object
    myCar.fullThrottle();      // Call the fullThrottle() method
    myCar.speed(200);          // Call the speed() method
  }
}
```

Output: The car is going fast as it can!

Max speed is: 200

Example explained

1) We created a custom Main class with the class keyword.

2) We created the fullThrottle() and speed() methods in the Main class.

3) The fullThrottle() method and the speed() method will print out some text, when they are called.

4) The speed() method accepts an int parameter called maxSpeed - we will use this in **8)**.

5) In order to use the Main class and its methods, we need to create an **object** of the Main Class.

6) Then, go to the main() method, which you know by now is a built-in Java method that runs your program (any code inside main is executed).

7) By using the new keyword we created an object with the name myCar.

8) Then, we call the fullThrottle() and speed() methods on the myCar object, and run the program using the name of the object (myCar), followed by a dot (.), followed by the name of the method (fullThrottle(); and speed(200);). Notice that we add an int parameter of **200** inside the speed() method.

Remember that..

The dot (`.`) is used to access the object's attributes and methods.

To call a method in Java, write the method name followed by a set of parentheses **()**, followed by a semicolon (`;`).

A class must have a matching filename (`Main` and **Main.java**).

Using Multiple Classes

Like we specified in the Classes chapter, it is a good practice to create an object of a class and access it in another class.

Remember that the name of the java file should match the class name. In this example, we have created two files in the same directory:

- Main.java
- Second.java

Main.java

```
public class Main {

  public void fullThrottle() {

    System.out.println("The car is going as fast as it can!");

  }


  public void speed(int maxSpeed) {

    System.out.println("Max speed is: " + maxSpeed);

  }

}
```

Second.java

```
class Second {

  public static void main(String[] args) {

    Main myCar = new Main();     // Create a myCar object

    myCar.fullThrottle();      // Call the fullThrottle() method

    myCar.speed(200);          // Call the speed() method

  }

}
```

When both files have been compiled:

```
C:\Users\Your Name>javac Main.java
C:\Users\Your Name>javac Second.java
```

Run the Second.java file:

```
C:\Users\Your Name>java Second
```

And the output will be:

```
The car is going as fast as it can!
Max speed is: 200
```

Reference:

https://www.w3schools.com/java/java_class_methods.asp