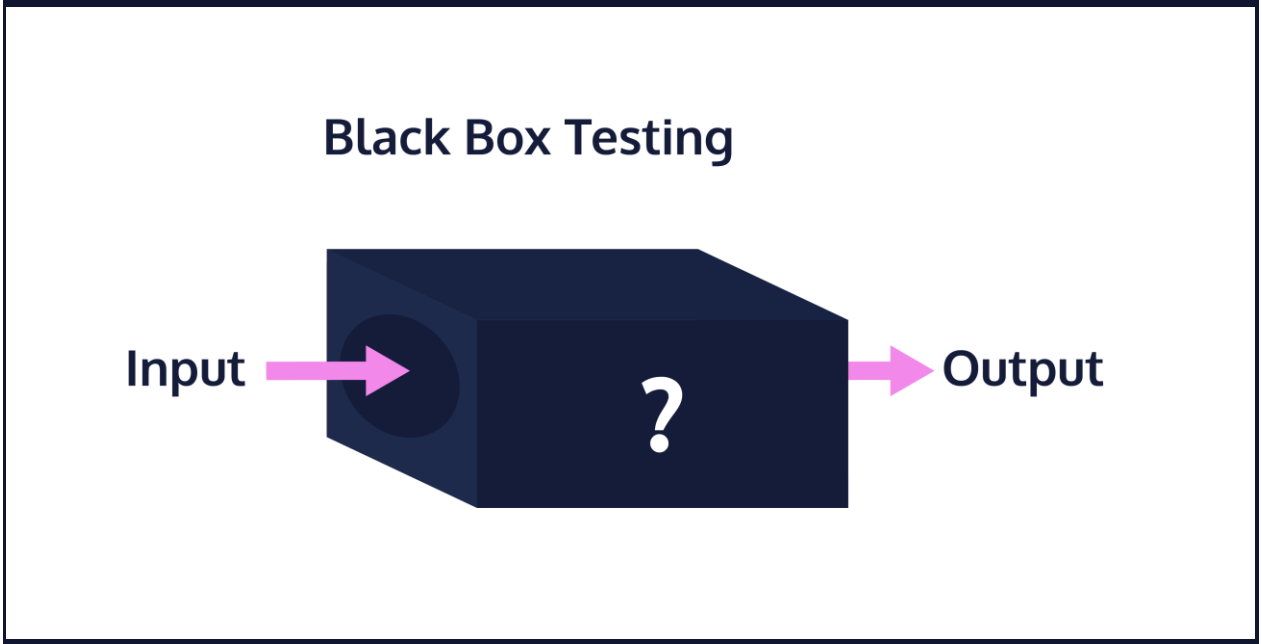
Black Box Programming

Learn some best practices for writing methods.

What Is A Black Box

As you begin to work with classes and objects, you may come across the term *black box* programming. Often times, programmers write methods as if they were black boxes — the programmer knows what is going on inside the box, but all of those implementation details are hidden from the user. The user should be aware of the input of the black box, and they should be able to predict an output given the input, but the details of how that input gets transformed into the output can be hidden from the users.



Most technology we use these days is a black box to some degree. Consider a television remote — we can give the remote some input, like pressing the "Volume Up" button, and we can expect some output, like the volume increasing. But beyond that, all of the inner workings of the process is hidden from us. We, as the television remote user, don't *really* know how the remote communicates with the TV or how the TV interfaces with its speakers. All we know is the format of the input and the expected output. Then we let the black box do the rest.

Preconditions and Postconditions

To relate this back to Java, it is important to think of *preconditions* and *postconditions* of a method. A precondition is a condition that must be true in order for a method to work as expected. These preconditions should be communicated to the user in some way.

For example, in order for your TV remote to work as expected, the TV might have to already be on. For a Java example, you might be writing a function that takes a number as an input. A potential precondition could be that number should be positive.

You could write your method that checks whether or not that precondition is true. For example, if your method requires a positive number, you could print an error message if the method receives a negative number. It is good practice to write methods that account for a variety of different situations but in reality, some methods will have preconditions that you will need to be aware of.

Similarly, it is important to think about the postcondition of a method. A postcondition is a condition that is guaranteed to be true after a method is called, given that the preconditions were met. Going back to our TV example, the postcondition of the "Turn Up Volume" is that the volume will increase by 1. This will only be true if the preconditions are met. The preconditions might be something like "the TV is on" and "the TV is not at maximum volume".

Just like with preconditions, the postconditions should be communicated to the user of the method. This is often done through documentation. Take a look at some examples of official Java documentation. Try to find where the documentation describes the precondition and postcondition of these methods. Some of these methods are from classes you haven't encountered yet:

- indexOf() from the String class
- pow() from the Math class
- nextDouble() from the Random class

As you take a look at this documentation, notice that some methods describe more implementation details than others. If you want to learn more about a black-box method, there are ways to dive deeper into the code or documentation. A black box doesn't have to be black forever — if you're truly curious about how a method works, you can always find more information. However, as you begin to write your own methods, remember that at a bare

minimum, you should make it clear what the preconditions and postconditions of your methods are.