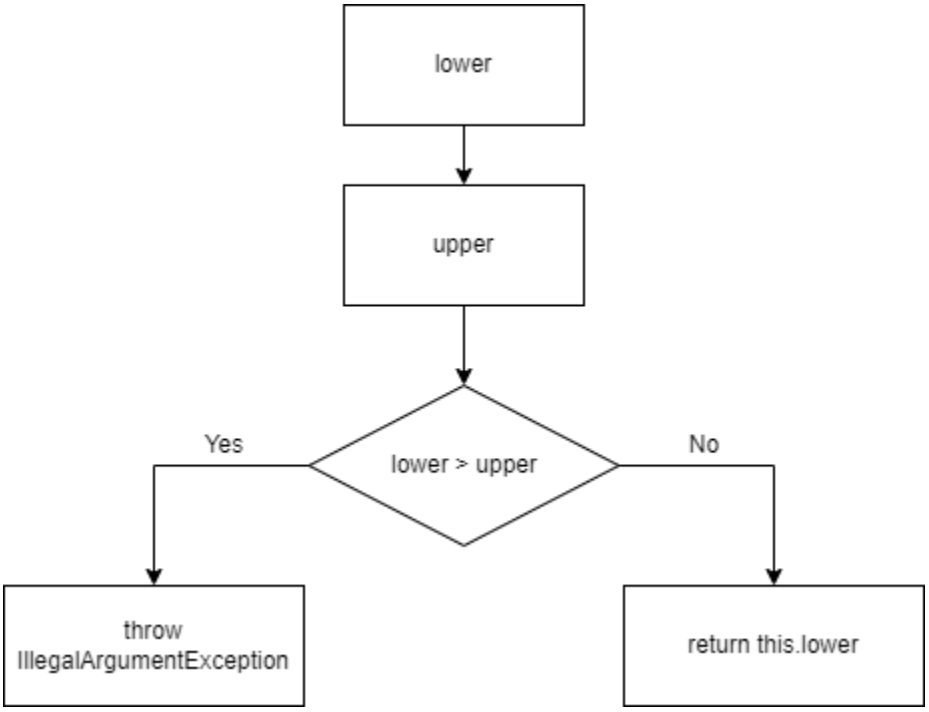
Data Flow Diagram:



The def-use sets per statement:

Statement:	Define	Use
If(lower > upper) {	None	lower, upper
String msg = "Range(double, double): require lower (" + lower + ") <= upper (" + upper + ").";	msg	lower, upper
throw new IllegalArgumentException(msg);	None	msg
return this.lower;	None	lower

List of all DU-pairs per variable:

lower:

Definition: defined in the constructor for the class.

Uses:

- Statement 1: if (lower > upper) {
- Statement 2: String msg = "Range(double, double): require lower (" + lower + ") <= upper (" + upper + ").";
- Statement 4: return this.lower;

upper:

Definition: defined in the constructor for the class.

Uses:

- Statement 1: if (lower > upper) {
- Statement 2: String msg = "Range(double, double): require lower (" + lower + ") <= upper (" + upper + ").";

msg:

Definition:

Statement 2: String msg = "Range(double, double): require lower (" + lower + ") <= upper (" + upper + ").";

Uses:

- Statement 3: throw new IllegalArgumentException(msg);

Test Cases:

The test cases are:

- getLowerBoundUnequalTest
- getLowerBoundEqualTest
- getLowerBoundNegativeTest

All three of the testcases create a Range object which defines the upper and lower. Then when the test cases are run, those 2 variables get used for the comparison in statement 1 which is true for all test cases and then lower is returned.

lower:

Definition: Defined in the test case when the constructor is called.

Uses:

- Statement 1: if (lower > upper) {
- Statement 4: return this.lower;

upper:

Definition: Defined in the test case when the constructor is called.

Uses:

- Statement 1: if (lower > upper) {

DU-Pair coverage:

Out of the 6 DU pairs, only 3 of them end up being covered by the tests. This is not a flaw in the test cases and the only way to increase this would be to rewrite the original method. The only way to define lower and upper is through the constructor. That constructor already if lower is greater than upper and throws an exception if it's true. As a result of this, it is impossible for the if statement to be true and the code is inside the if block is unreachable. This makes it so those last 3 DU pairs can never be tested and they can safely be ignored from the calculation. So the DU coverage ends up being 3/3 = 100%.