3.2

A class diagram is static and only changes if the class files are changed.

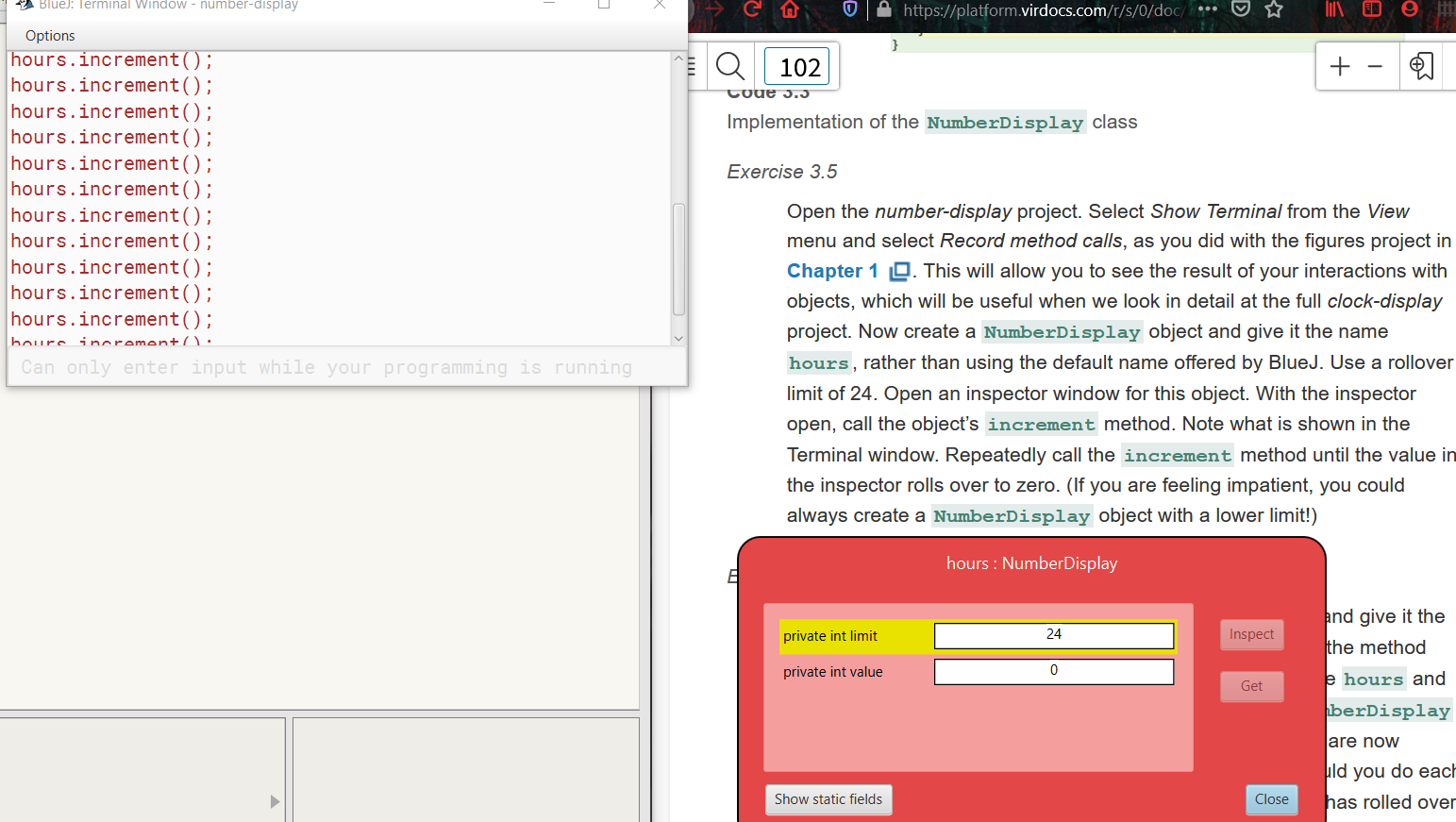
3.3

An object diagram is a representation of objects and their relationships at **one** moment in time during execution of a program. So, an object diagram would change depending upon which part of the program is currently executing or being examined.

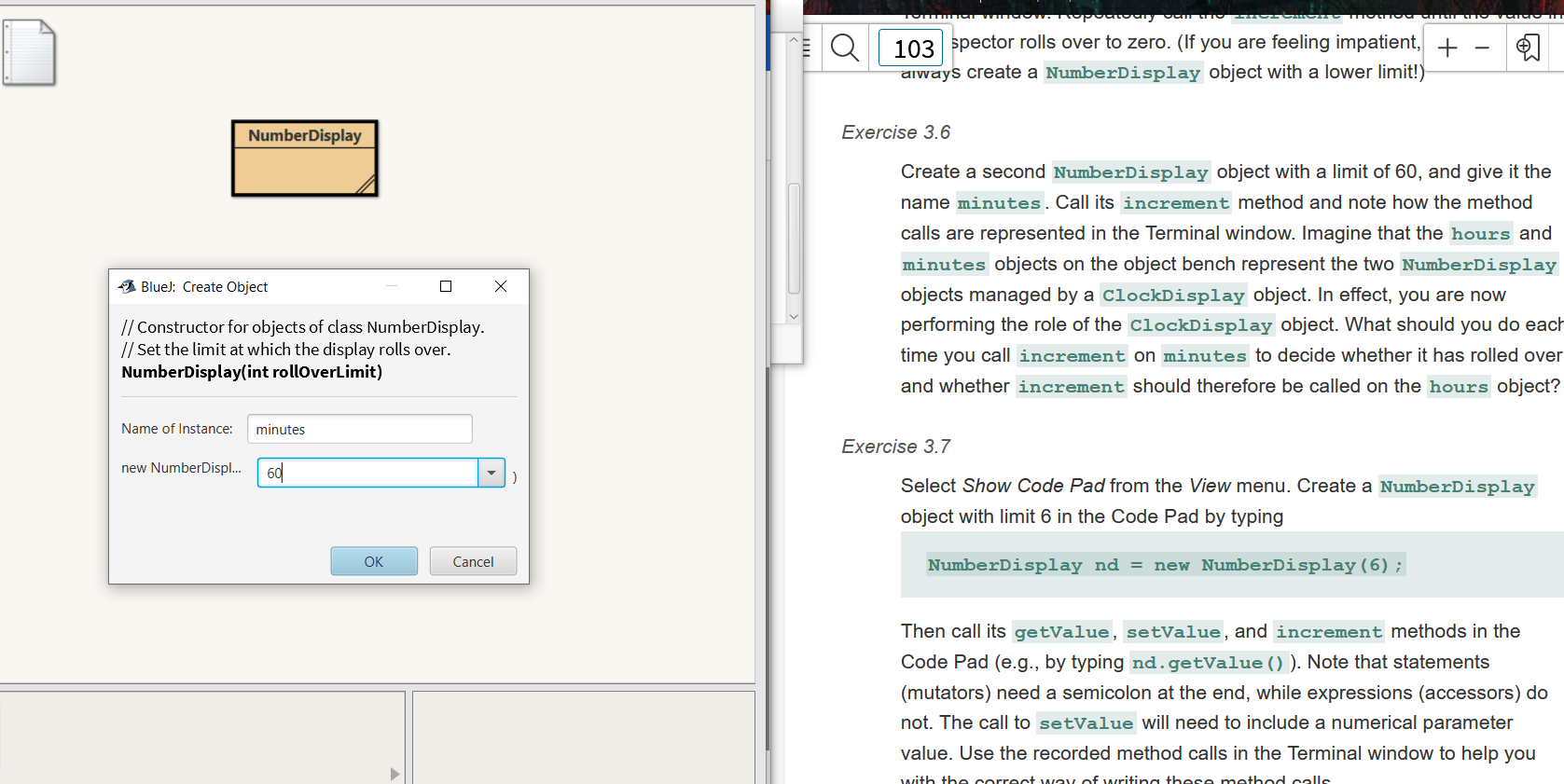
3.4

**private Instructor tutor;**

3.5

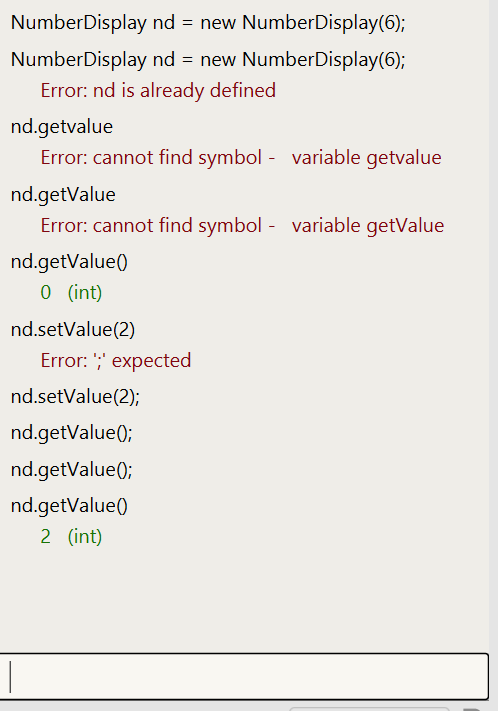


3.6

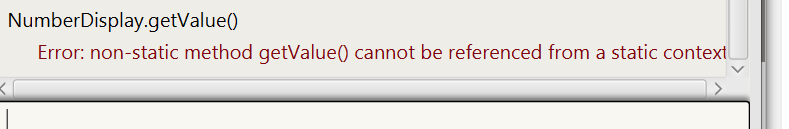


You should have the hours increment one every sixty times the minutes is incremented.

3.7

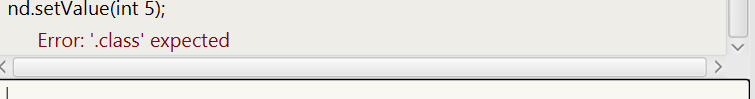


3.8



This is because the class name is being called not the variable name nb.

3.9



I’m guessing this is happening because the parameter type does not need to be specified here.

3.10

Nothing happens when an illegal replacement value is entered. This is an ok solution it might be better to have an ouput to the user to tell them why the value was not changed.

3.11

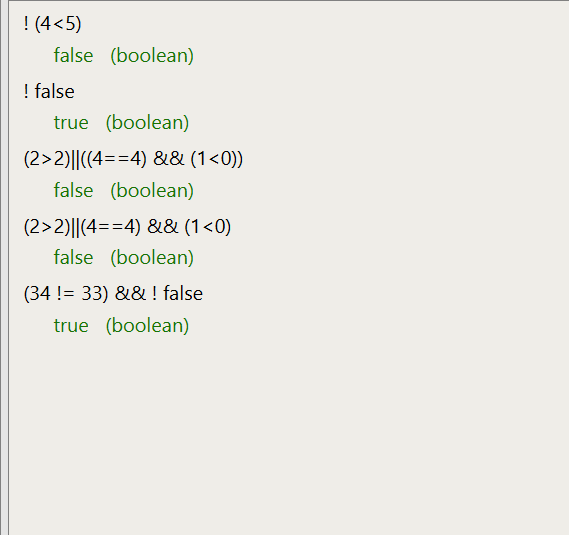
This would make it so that a ‘0’ would be an illegal value to use as a replacement. This is not what we would want since ‘0’ would be fine as a replacement value.

3.12

This would end up allowing numbers that are negative to be legal because they only must adhere to one of the two conditions with || instead of &&. This is not what we want here.

3.13

The second one, “! False “, would return true. Also, the last one, “(34 != 33) && ! False”, would return true since both sides of the ‘&&’ are true.



3.14

A == B

3.15

A == ! B

3.16

A == B

3.17

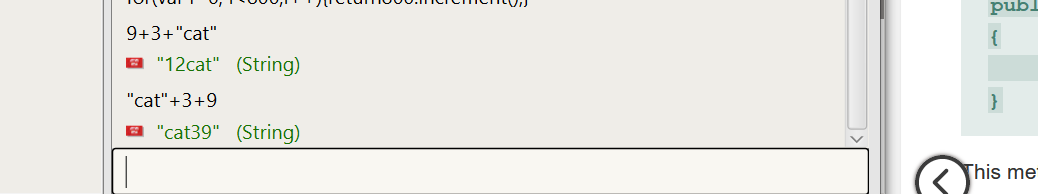
No, it’s not so good with three digit numbers but does work alright for the clock.

3.18

No, I do not think there is a difference I tried it in the editor both ways and they both work alright.

3.19

Yes, so, to my surprise they do come out different. When the integer type comes first it does add them as integers until it encounters a string type then at that time it changes the whole thing to a string. After it has already added the integers.



The first one adds 9 and 3 first then changes to a string and concatenates to string, “cat”. Whereas in the second one the string, “cat”, comes first and starts concatenation turning the first integer it encounters to produce a string before the integers are added.

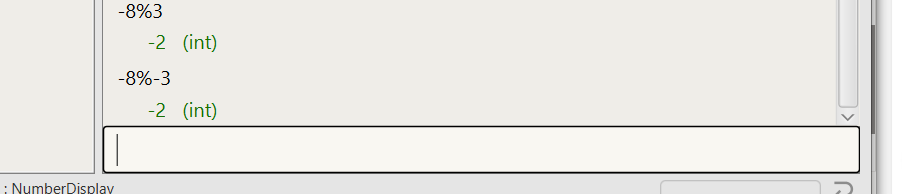
3.20

The modulus operator returns the remainder of the division of two numbers.

3.21



3.22



It seems like negatives come out with a return of a negative as well.

3.23

0-n

3.24

All positive integer results are possible results.

3.25

The increment method sets value = value + 1. That is simple to understand. But, then it does a % limit after that. This is how we make sure that when the limit is reached, we roll back over to ‘0’.

3.26

Public void increment()

{

If(value == limit - 1)

{

value = 0;

}else

{

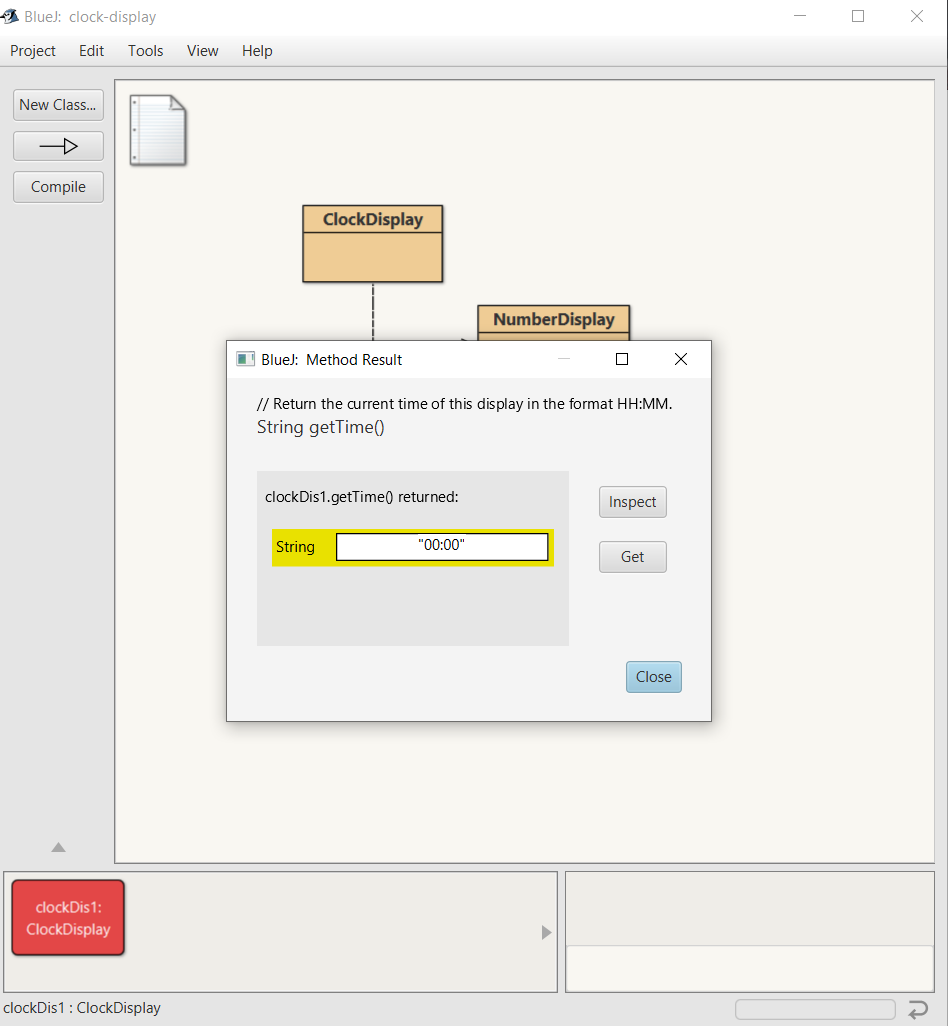
Value = value + 1;

}

}

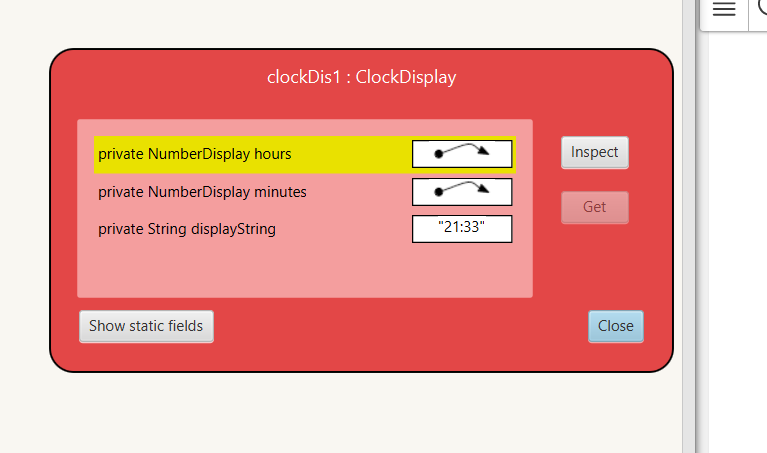
So, the if statement is more obvious when you look at it to see what's going on. Although, I really like how much less typing is needed when using the modulo operator. I think using the modulo is much cleaner looking and less coding, but it is kind of tricky and not obvious at first when reading it.

3.27



I think it starts at midnight here because I entered 24 for the hours and 60 for the minutes when I had set it up. I did another one with different inputs and got the time I had entered.

3.28



3.29

If it started at 00:00 you would have to click it 60 times to reach 01:00. You could call the setTime() method and enter 1 for the hour and 0 for the minutes to get this result.

3.30

Private Rectangle window;

window = new Rectangle(10, 5);

3.31

With the second, overloaded constructor for ClockDisplay the setTime() method is called and the time is set to whatever parameters are given. So, the tame can be set to something other than 00:00 to start.

3.32

The differences are that the first constructor calls the updateDisplay() method at the end whereas the second constructor calls the setTime() method. This is because the second one sets the time instead of defaulting to 00:00. It then does not need to call updateDisplay since the setTime() method will already do this.

3.33

p1.print(coolFile.doc, true);

p1.print(alrightFile.txt, false);

p1.getStatus(3);

p1.getStatus(5);

3.34

In chapter one in the house project the class Picture’s constructor Picture() used four object types: Square, Triangle, Circle, and Person.

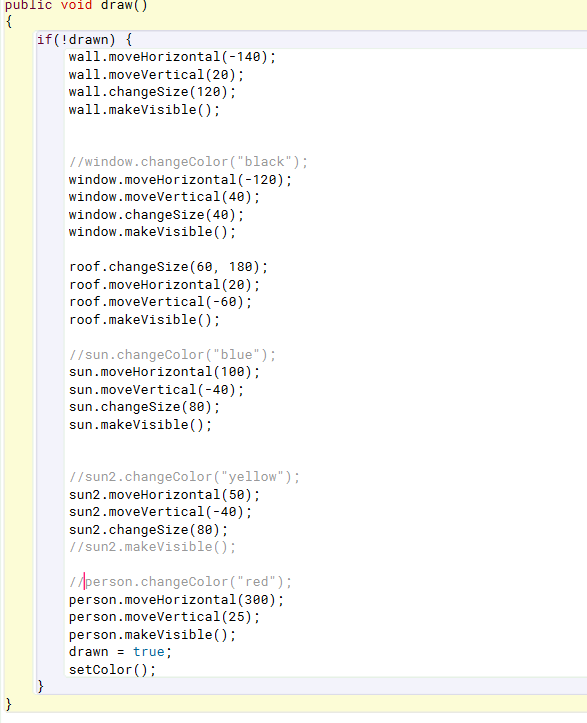
3.35

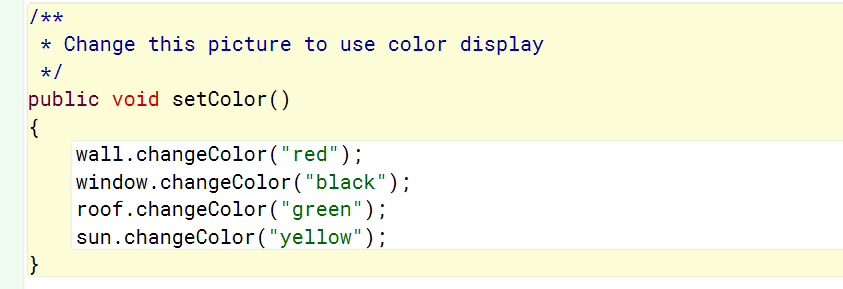
The Triangle object called roof makes four external method calls in the method draw(). They are: changeSize(), moveHorizontal(), moveVertical(), and makeVisible().

3.36

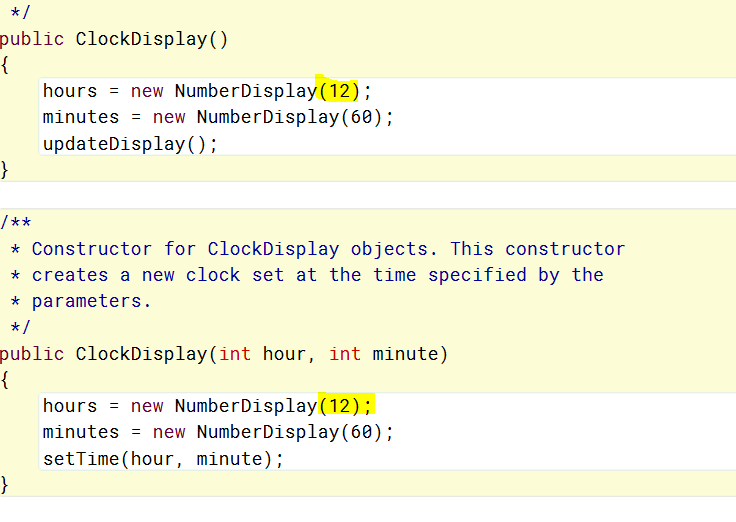
No, it looks like the Picture class only make external method calls.

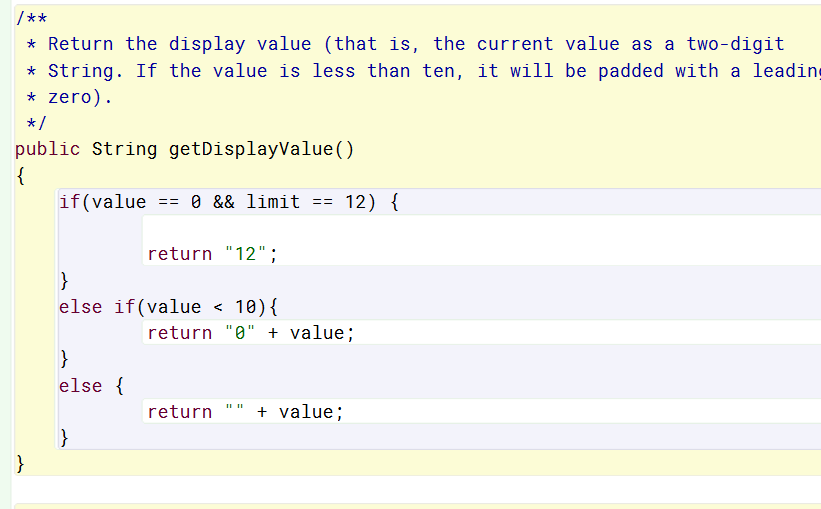
3.37





3.38





3.39

Out of both ways it is easier as well as more useful to change the display string of the clock in order to display as a 12 hour display. This is because the clock is really a 24 hour thing if you want to have am and pm. So, it makes sense to build it this wa and just change the way it is displayed. You can then have the same inner working run all types of displays.

3.40/3.41

public class Tree

{

private Triangle leaves;

private Rectangle trunk;

/\*\*

\* Constructor for objects of class Tree.

\* Set trunk and leaves shapes.

\*/

public Tree()

{

trunk = new Rectangle(5, 5);

leaves = new Triangle(7);

setUp();

}

/\*\*

\* Method to set up the Tree.

\*

\*/

SetUp()

{

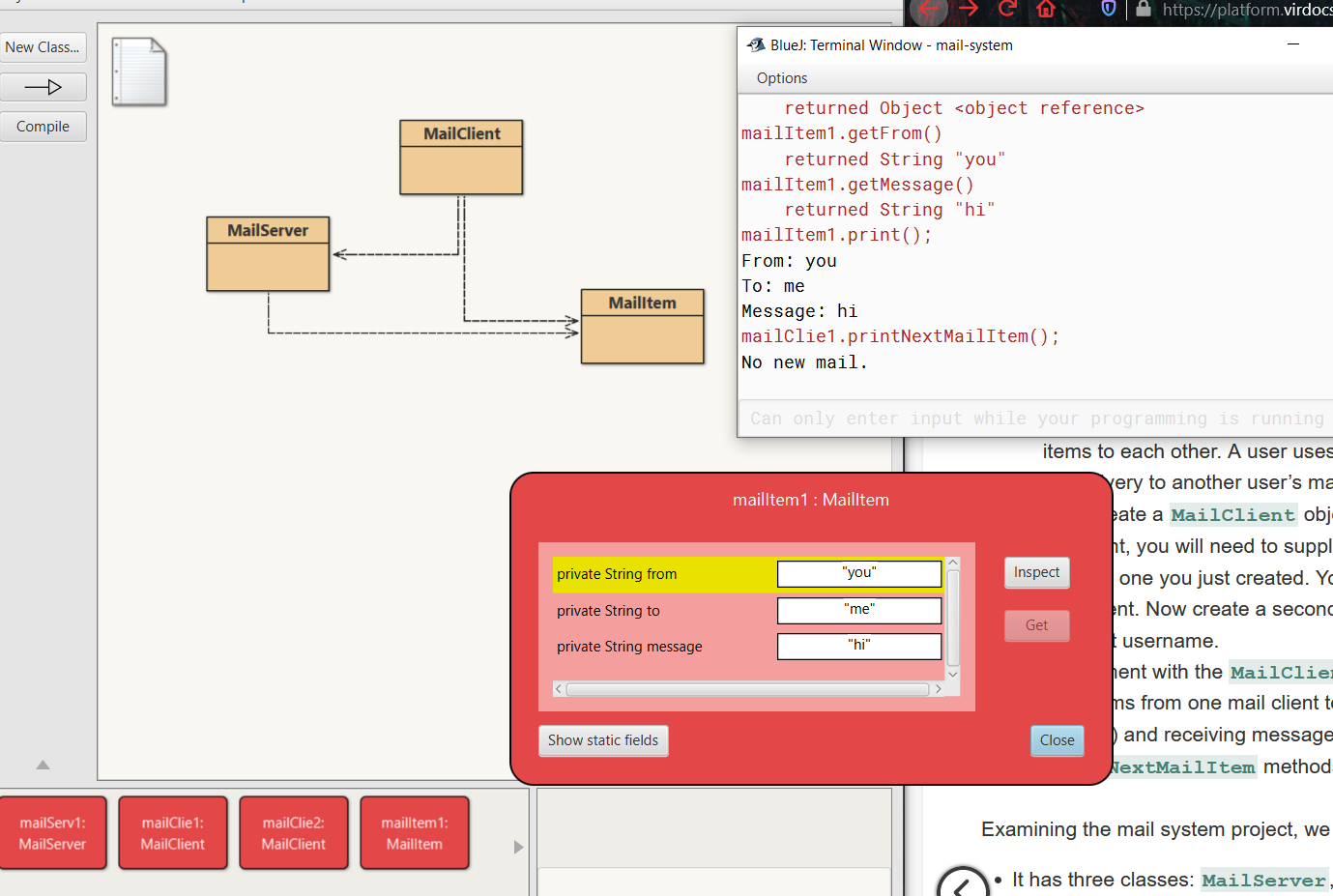
leaves.changeSize(10);

leaves.moveVertical(10);

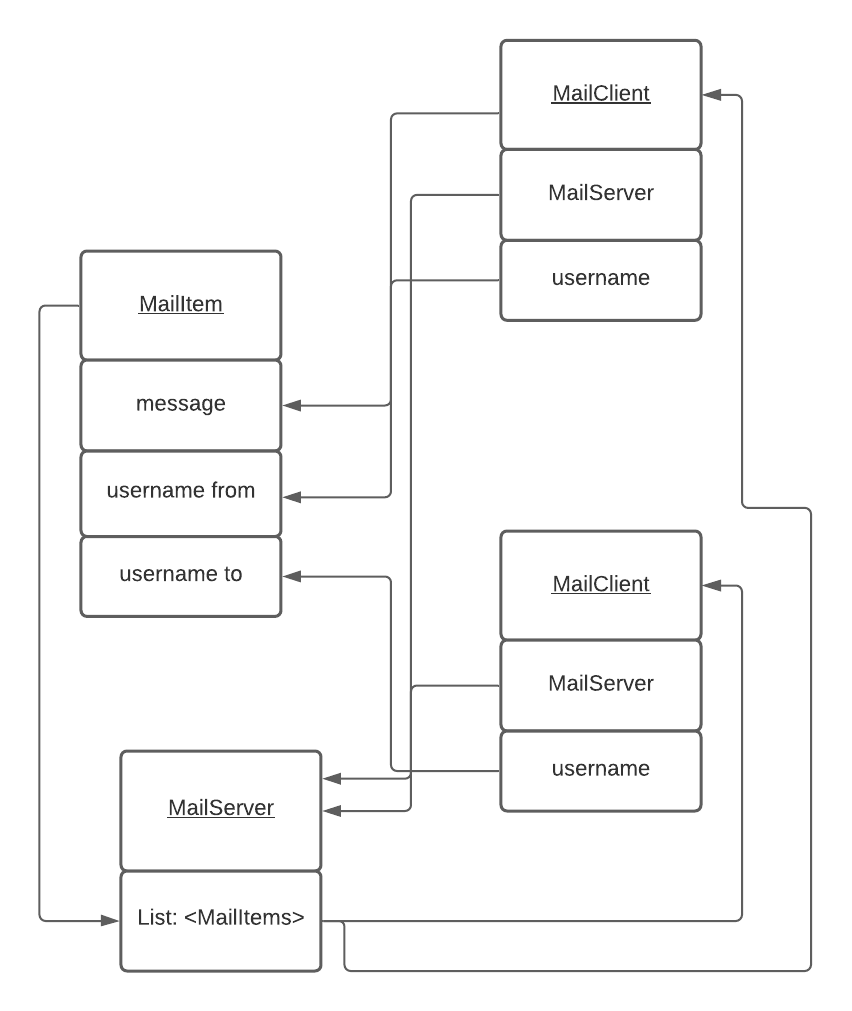
}

}

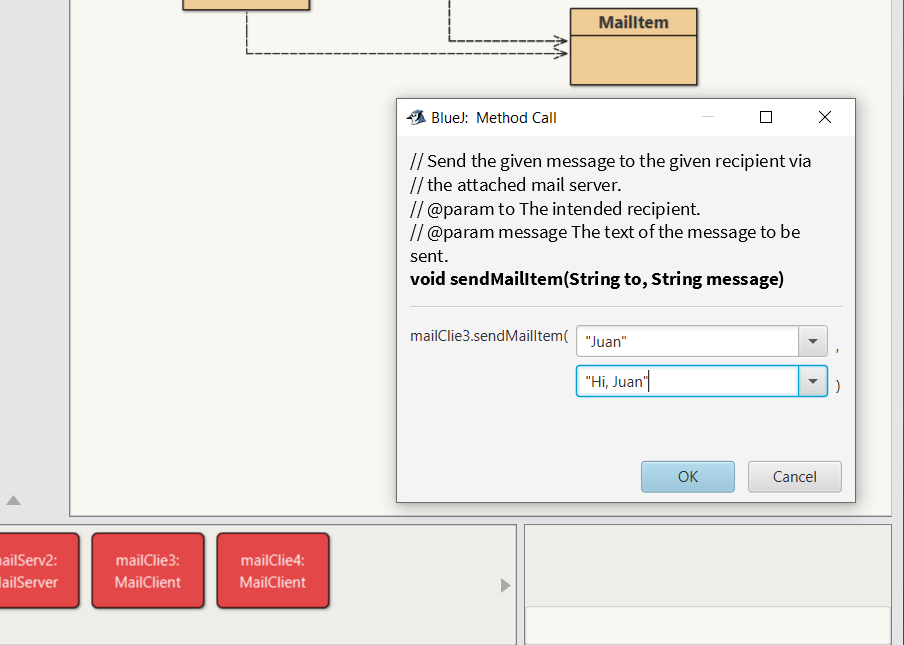
3.42



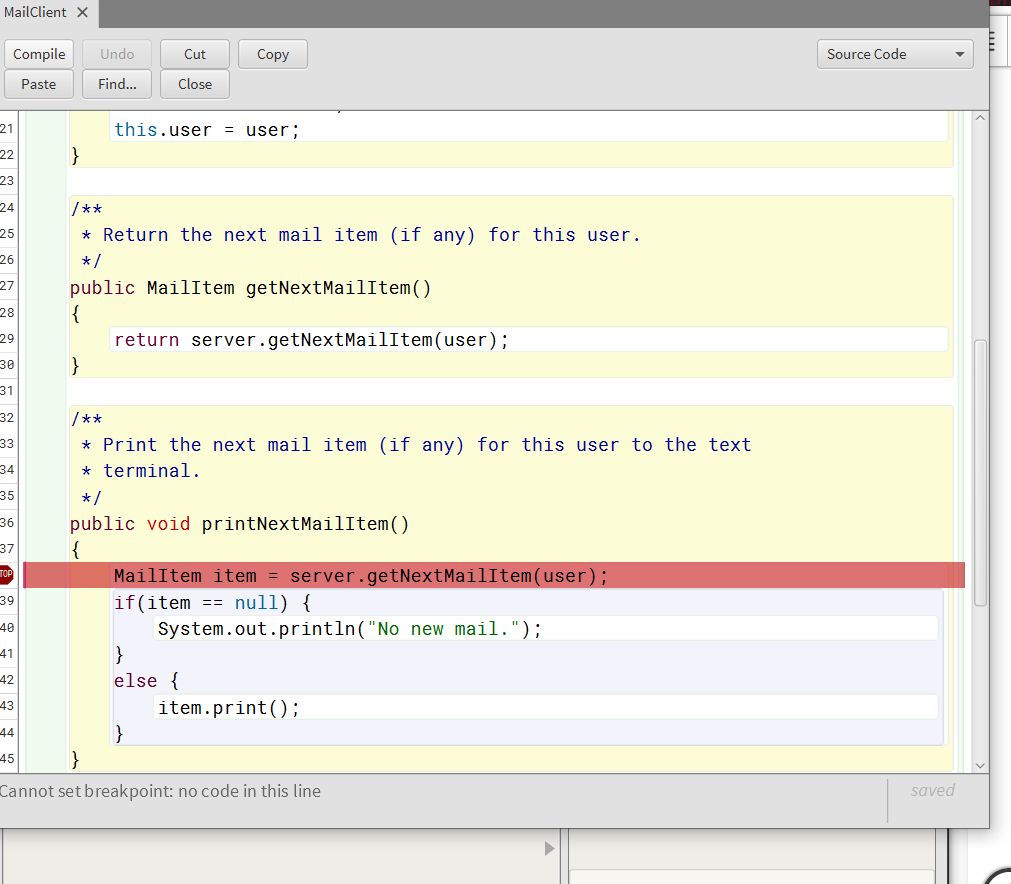
3.43



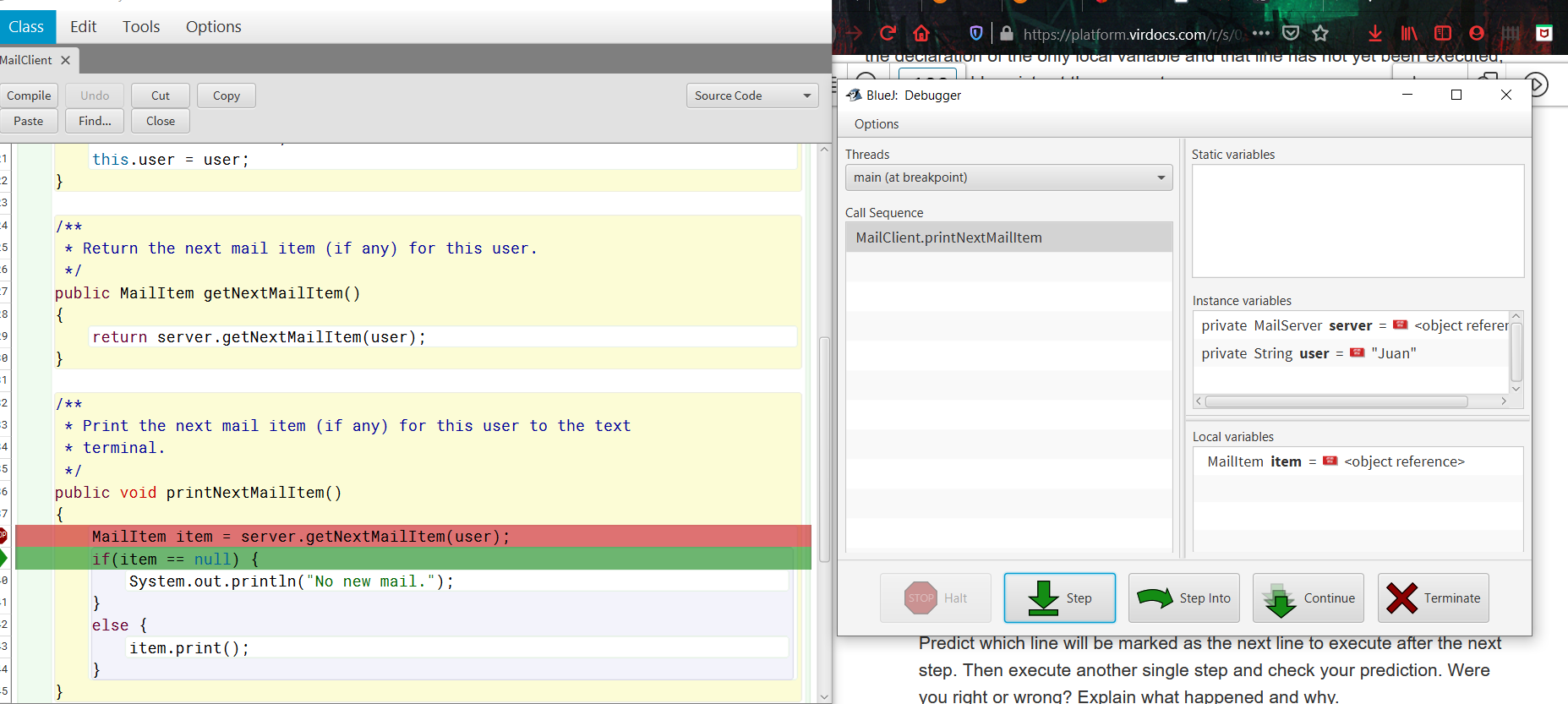
3.44



3.45

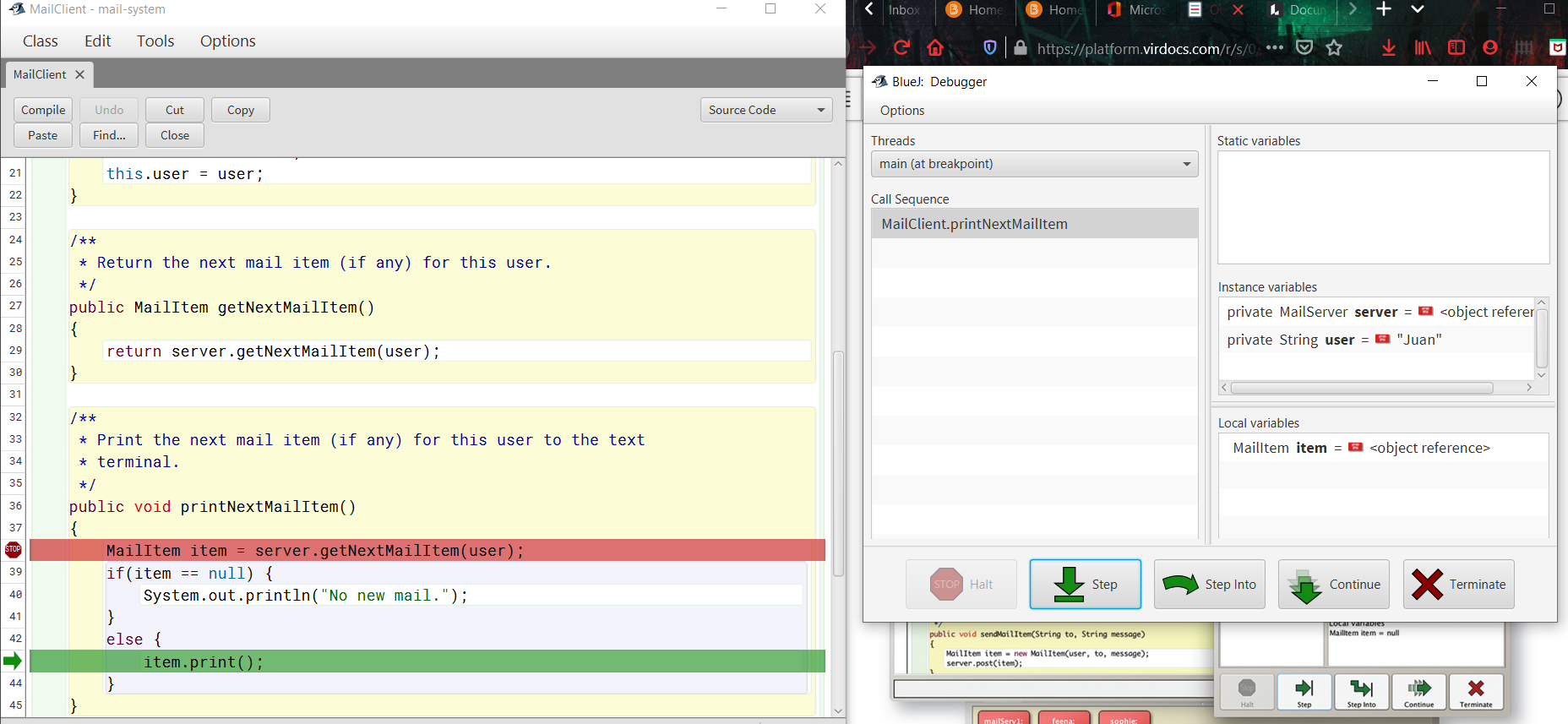


3.46



3.47

Line 43 because item is not null.



Yes, line 43! Because the result of the if was false, so we went to the else.

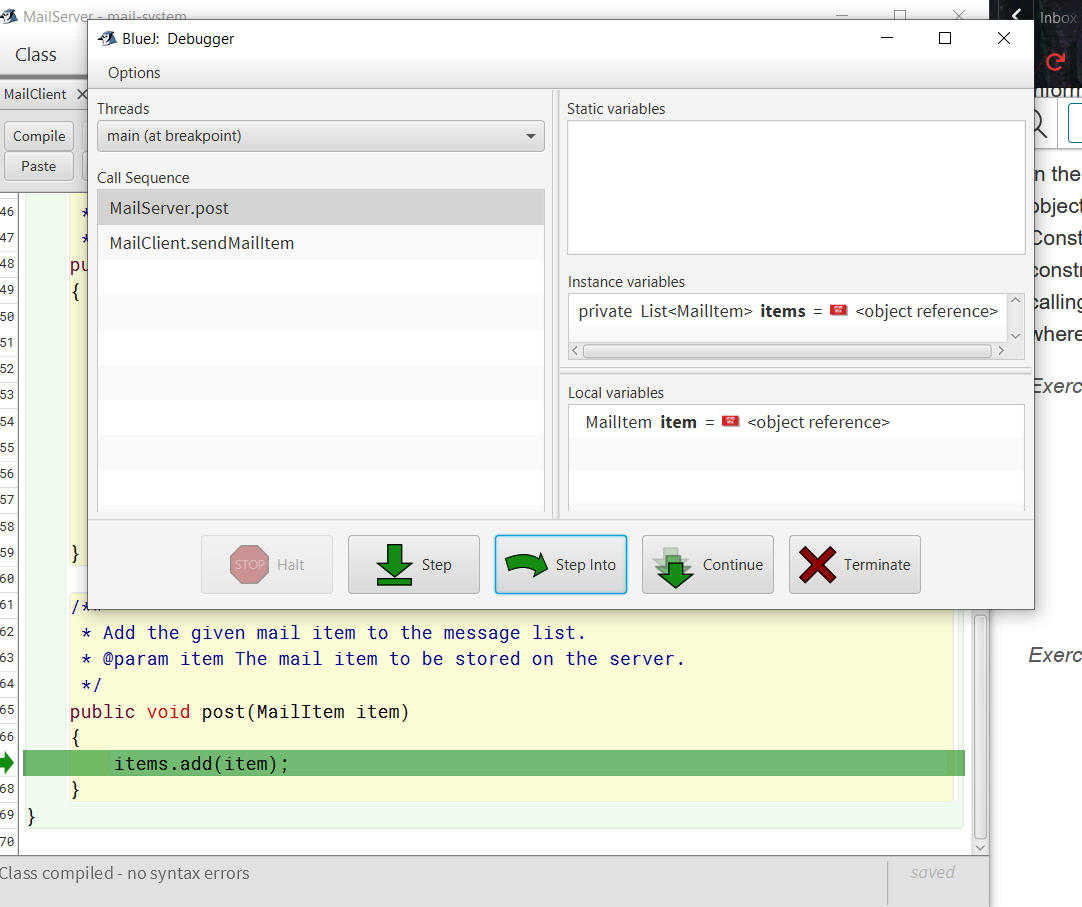
3.48

This time item was null, so the program printed the no mail prompt and hit the if statement (line 40) on next step.

3.49

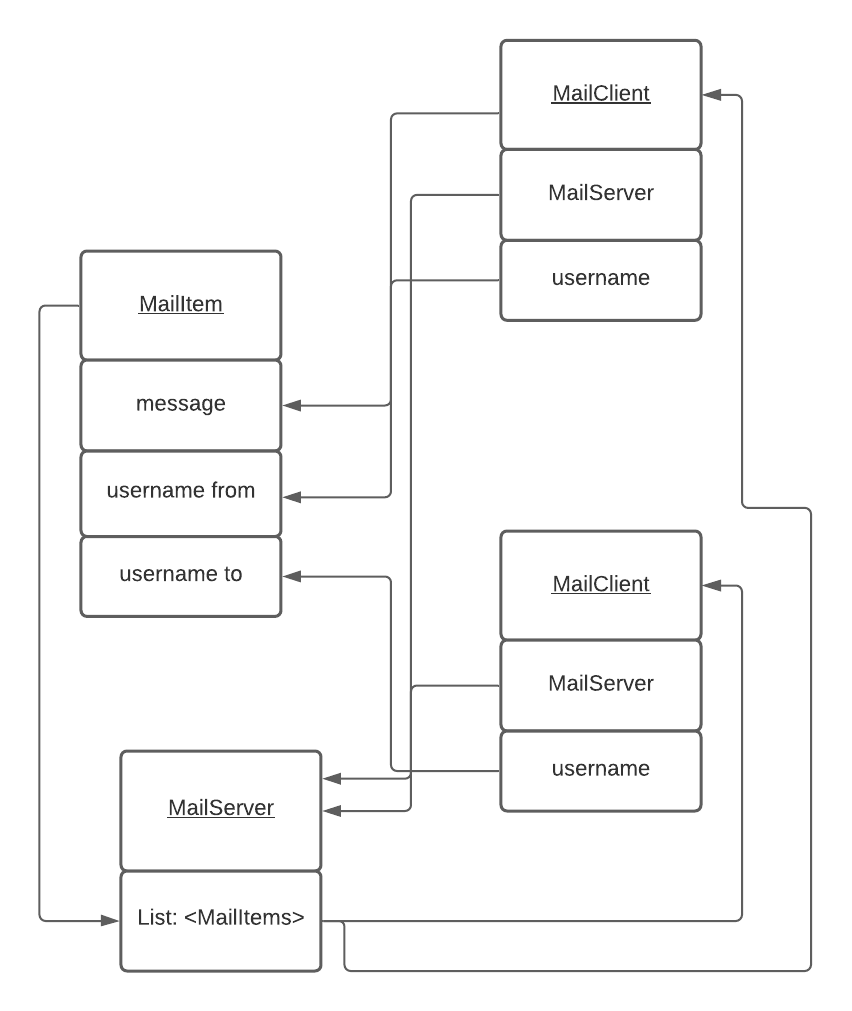
I see it print the to and from as well as the message step by step. This is because the step into command steps into the print method and goes step by step through the lines executing just one at a time.

3.50



3.51

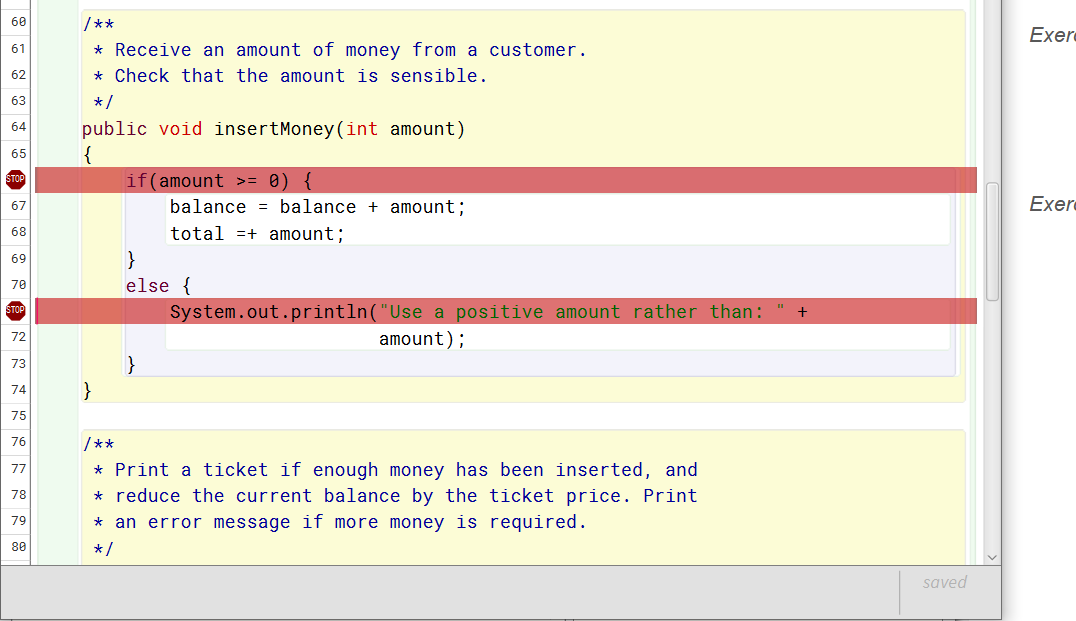
The mail client creates a mail item, and the mail item gets stored on the mail server. The mail client then can retrieve a mail item from the mail server.

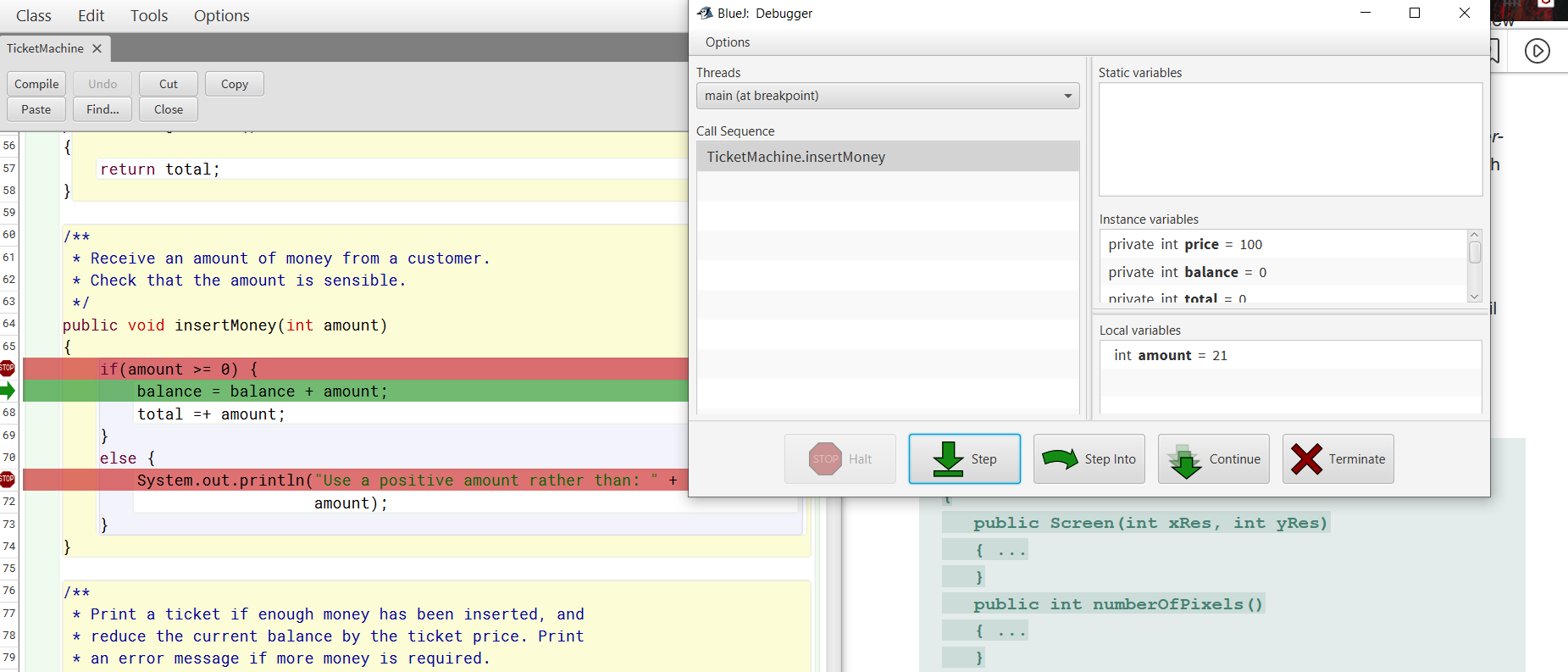


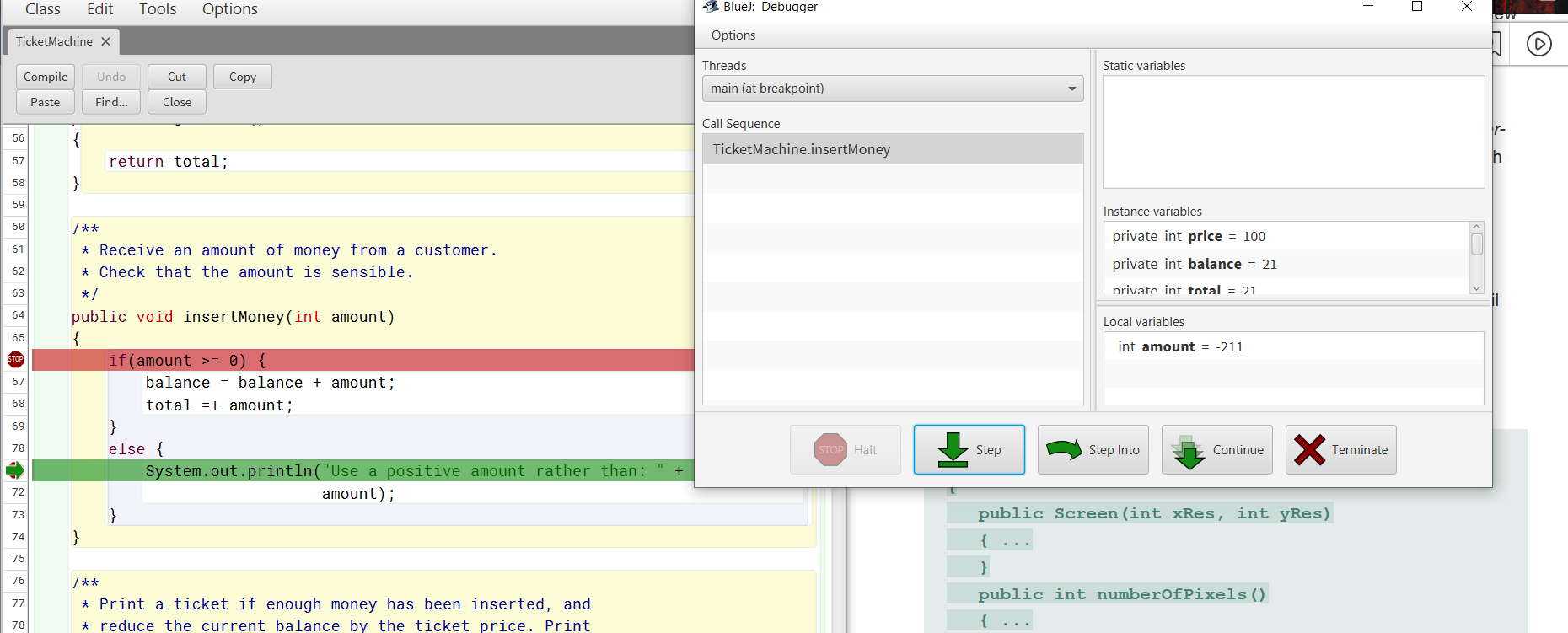
3.52

It seemed to do mostly what I would expect. One thing I didn’t realize before is just how many steps it takes to do one time tic even.

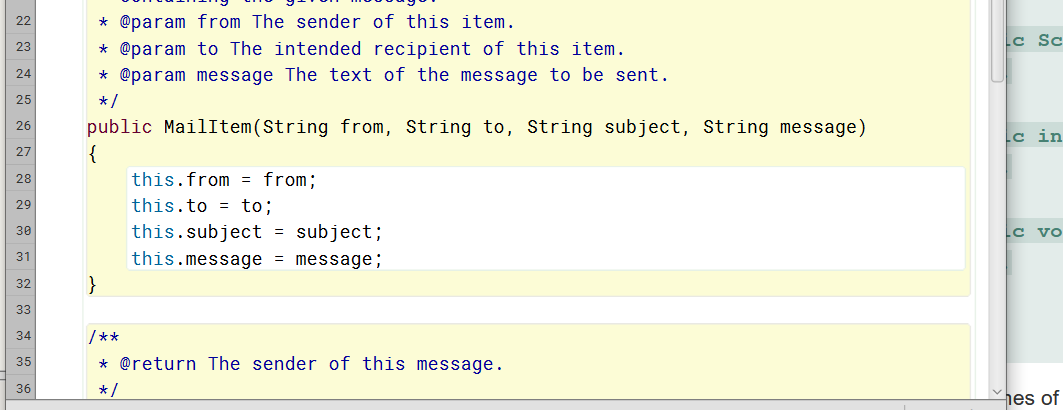
3.53

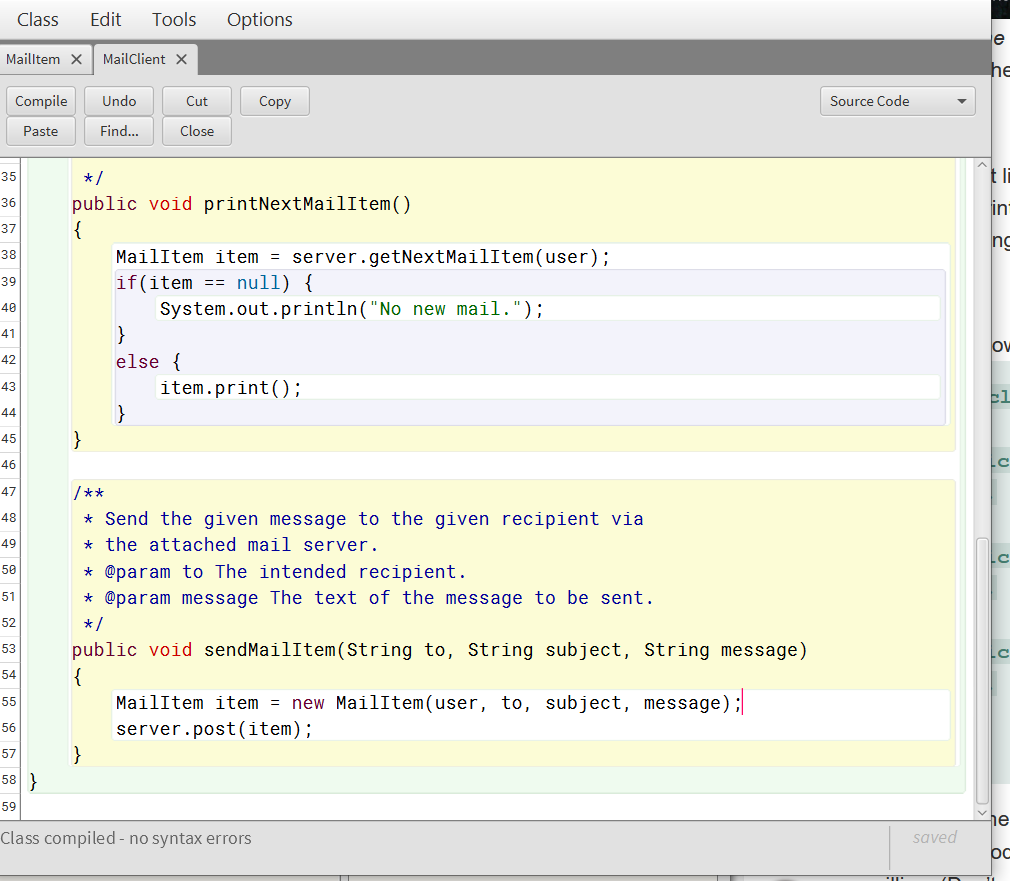


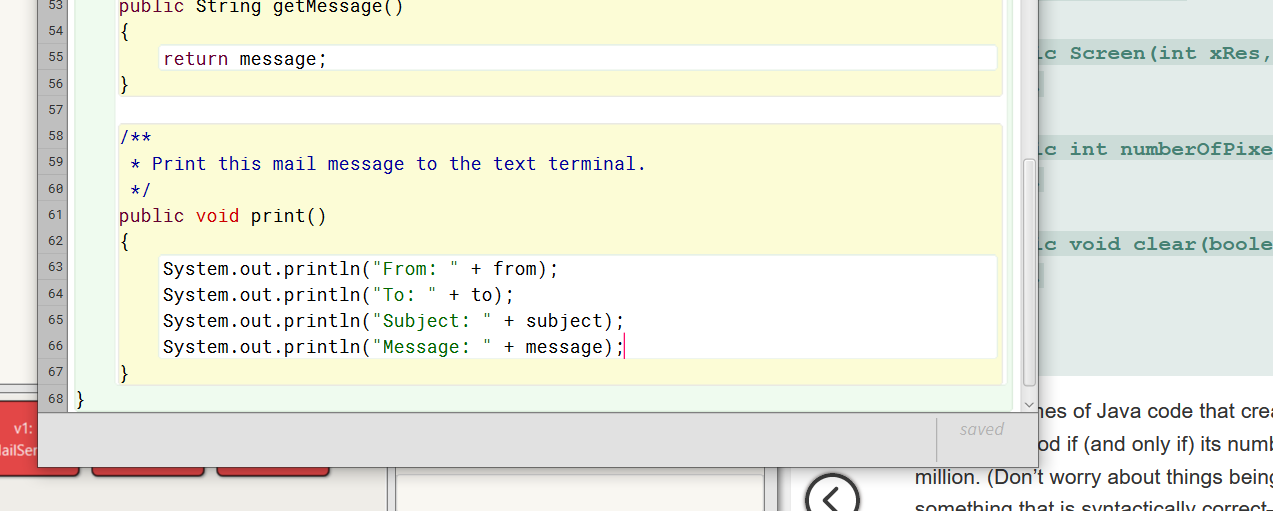




3.54







3.55