Exercise 3.1

The class diagram contains only 2 elements: LabClass and Student. The LabClass class is linked to the Student class.

The object diagram contains 4 elements. 1 LabClass object and 3 Student objects. The LabClass object contains links to the three Student objects.

Exercise 3.2

A class diagram changes when you modify the source code. That can be by changing the relations between classes or creating/deleting classes.

Exercise 3.3

An object diagram changes when the program is running. It can be changed by creating new objects, calling methods, and making assignments involving object references.

Exercise 3.4

private Instructor tutor;

Exercise 3.6

You should call getValue and if it returns 0 you should call increment on the hours object.

Exercise 3.8

Error: non-static method getValue() cannot be referenced from a static context

Exercise 3.9

Error: '.class' expected

Exercise 3.10

Nothing happens.
No.
It should print out an error message.

Exercise 3.11

It would not allow a value of zero to be used as a replacement.

Exercise 3.12

The test will be true if one of the conditions is true
It will always be true if the limit is larger than or equal to 0. It would also be true if the replacement value is negative, for instance.

Exercise 3.13

false
true
false
false
true

Exercise 3.14

The long version of this would be:

```
(a == true && b == true) || (a == false && b == false)
```

This can be simplified to:

```
(a && b) || (!a && !b)
```

But since both must have identical values, the simplest form is:

```
a==b
```

Exercise 3.15

As in the previous exercise, the long version would be:

```
(a == true && b == false) || (a == false && b == true)
```

This can be simplied to:

```
(a && !b) || (!a && b)
```

Or:

```
(a || b) && (a != b)
```

Or even:

```
a != b
```

Exercise 3.16

```
!(!a || !b)
```

Exercise 3.17

No.

The method assumes that the value will only contain two digits.

Exercise 3.18

No.

Exercise 3.19

“12cat” and “cat39”

Exercise 3.20

The exact definition can be found in The Java Language Specification Third Edition in section 15.17.3
https://docs.oracle.com/javase/specs/jls/se8/jls8.pdf

The correct answer may not be intuitive when negative values are involved.

The key is that the modulus operator works so that

```
(a / b) * b + (a % b) == a
```

rearranging we get:

```
a %  b == a - (a / b) * b
```

which works even when negative values are involved.

Exercise 3.21

2

Exercise 3.23

-4, -3, -2, -1, 0, 1, 2, 3, 4

Exercise 3.24

Values in the range from -(m-1) to (m-1)

Exercise 3.25

As long as value is smaller than limit, it gets incremented by 1 (the modulo can be ignored)
When value reaches the limit, the modulo operation will result in value being set to 0.
So, the increment method increments value by one, until the limit is reached, at which point it will start over at 0.

Exercise 3.26

Many students get this one wrong on the boundaries. A common *wrong* solution, which results in a value equal to the limit is:

```
public void increment()
{
    if(value < limit) {
        value = value + 1;
    }
    else {
         value = 0;
    }
}
```

A correct, but difficult to visually verify version, which is often the result of fixing the above is:

```
public void increment()
{
    if(value < limit - 1) {
        value = value + 1;
    }
    else {
         value = 0;
    }
}
```

Best, because it is easy to see that it is correct is:

```
public void increment()
{
    value = value + 1;
    if(value >= limit) {
         value = 0;
    }
}
```

Both ways of incrementing are equally good. Once the modulus operator is understood, its use is obviously more succinct.

Exercise 3.27

The time is initialized to 00.00.
The constructor creates two new NumberDisplay which are initialized to 0 (in the constructor for NumberDisplay)

Exercise 3.29

It needs 60 clicks
Using the method setTime() on the object.

Exercise 3.30

```
Rectangle window = new Rectangle(3,6);
```

Exercise 3.31

It initializes the time to the values passed to the method.
It uses the method setTime to set the time to the initial value.

Exercise 3.32

Both constructors creates two new NumberDisplays.
The first constructor calls updateDisplay and the second calls setTime(hour, minute).
In the second constructor there is no call to updateDisplay because this will be done in the method setTime.

Exercise 3.33

```
p1.print("file1.txt", true);

p1.print("file2.txt", false);

int status;

status = p1.getStatus(12);

status = p1.getStatus(34);
```

Exercise 3.34

Two Square objects, a Triangle and a Circle.

Exercise 3.35

changeSize, moveHorizontal, moveVertical and makeVisible.

Exercise 3.36

No.

Exercise 3.38 and 3.39

1) change the updateDisplay in ClockDisplay as this:

```
/**
 * Update the internal string that represents the display.
 */
private void updateDisplay()
{
    int hour = hours.getValue();
    String suffix = "am";
    if(hour >= 12) {
        hour = hour - 12;
        suffix = "pm";
    }
    if(hour == 0) {
        hour = 12;
```

```
    }
    displayString = hour + "." + minutes.getDisplayValue() +
suffix;
}
```

2)

```
public ClockDisplay()
{
    hours = new NumberDisplay(12); //changed
    minutes = new NumberDisplay(60);
    updateDisplay();
}

public ClockDisplay(int hour, int minute)
{
    hours = new NumberDisplay(12); //changed
    minutes = new NumberDisplay(60);
    setTime(hour, minute);
}

private void updateDisplay()
{
    int hour = hours.getValue();
    if(hour == 0) {
        hour = 12;
    }
    displayString = hour + "." + minutes.getDisplayValue();
}
```

Exercise 3.40 and 3.41

```
public class Tree
{
    // The tree's trunk.
    private Square trunk;
    // The tree's leaves.
    private Triangle leaves;

    /**
     * Constructor for objects of class Tree
     */
    public Tree()
    {
        trunk = new Square();
        leaves = new Triangle();
        setup();
    }

    /**
     * Set up the trunk and the leaves of the tree.
     */
    public void setup()
    {
        leaves.changeSize(120, 140);
        leaves.moveVertical(-140);
        leaves.moveHorizontal(128);
        trunk.makeVisible();
        leaves.makeVisible();
```

```
    }
}
```

Exercise 3.47

Since the debugger shows that:
Mailitem item = <object reference>
This indicates that item is not null, and the next line that will be marked is item.print()

Exercise 3.48

This time the item is null. The if-statement will then execute the line:
System.out.println("No new mail.");

Exercise 3.49

After pressing Step Into and then pressing Step it prints:
From: Sophie
After the next Step it prints:
To: Juan
And after another Step:
Message: Hi Juan!

Step Into goes into the method print(). From there on, each line of print is executed by a pressing Step.
This results in the lines being printed one at a time instead of just printing all 3 lines as before.

Exercise 3.54

A subject field must be added to the MailItem class and set via the constructor. An accessor is also added and the print() method should include it.

The MailClient's sendMailItem() method requires a parameter to receive the subject.

See **03-54-mail-system**.

Exercise 3.55

```
Screen screen = new Screen(1024, 768);
if(screen.numberOfPixels() > 2000000) {
    screen.clear(true);
}
```

Exercise 3.56

A third NumberDisplay object would need to be created in both constructors. The setTime method would add a third parameter. The updateDisplay method would

need to build the displayString from the three values. The logic of the timeTick method becomes more complicated. The seconds must be incremented and if they roll over then the minutes must be incremented. The hours are incremented as currently.

Exercise 3.57

```
    public void timeTick()
    {
        seconds.increment();
        if(seconds.getValue() == 0) {
            // it just rolled over!
            minutes.increment();
            if(minutes.getValue() == 0) {
                hours.increment();
            }
        }
        updateDisplay();
    }
```

Exercise 3.58

In theory it could, but the logic would be unwieldy.

Exercise 3.59

This is likely to be very challenging at this stage. The NumberDisplay class would have an additional field such as:

```
private NumberDisplay nextTimeUnit;
```

This would need be set from a parameter passed in by the ClockDisplay. For instance,

```
public ClockDisplay()
{
    hours = new NumberDisplay(24, null);
    minutes = new NumberDisplay(60, hours);
    seconds = new NumberDisplay(60, minutes);
}
```

The increment method in NumberDisplay would be:

```
public void increment()
{
    value = (value + 1) % limit;
    if(value == 0) {
        if(nextTimeUnit != null) {
            nextTimeUnit.increment();
        }
    }
}
```