Exercises 7.1-7.5

The functionality tested here should all work correctly.

Exercise 7.6

It is possible to give a rating of zero.

Exercise 7.7

It is possible to down-vote an item to a negative number of votes.

Exercise 7.8

The functionality tested here should work correctly, with the exception of the problem exposed in Exercise 7.9.

Exercise 7.9

The `findMostHelpfulComment()` method fails if there are no comments. There error is a `NoSuchElementException`.

Exercise 7.10

The `downVote()` method should protect its field from going negative:

```
    public void downvote()
    {
        if(votes > 0) {
            votes--;
        }
    }
```

The `ratingInvalid()` method should correctly test its boundaries:

```
    private boolean ratingInvalid(int rating)
    {
        return rating < 1 || rating > 5;
    }
```

The `findMostHelpfulComment()` method should not assume that there is at least one comment:

```
    /**
     * Return the most helpful comment. The most useful comment is the
one with the highest vote
     * balance. If there are multiple comments with equal highest
balance, return any one of
     * them.
     * Return null if there are no comments.
     */
    public Comment findMostHelpfulComment()
    {
        Comment best = null;
        if(comment.size() > 0) {
```

```
            Iterator<Comment> it = comments.iterator();
            best = it.next();
            while(it.hasNext()) {
                Comment current = it.next();
                if(current.getVoteCount() > best.getVoteCount()) {
                    best = current;
                }
            }
        }
        return best;
    }
```

Any change to the code could have knock-on effects on existing code, or introduce additional errors, so it is not safe to assume that previous tests will still be passed.

Exercise 7.11

Test	Positive or Negative
Exercise 7.1	positive
Exercise 7.2	positive
Exercise 7.3	positive
Exercise 7.4	negative
Exercise 7.5	positive
Exercise 7.6	negative
Exercise 7.7	positive and negative (downvoting below zero)
Exercise 7.8	positive
Exercise 7.9	negative

Exercise 7.14

The methods `setUp()` and `tearDown()` are created automatically.

Exercise 7.15

- Start recording.
- Create a `SalesItem` object.
- Call the `addComment()` method to add a comment. Assert that the result should be `true`.
- Call the `addComment()` method again to add a comment – use the same author

name as in the previous comment. Assert that the result should be `false`.

- Select the *End* button to finish the recording.

Exercise 7.17

An error diagnostic appears in the lower panel of the *Test Results* window. The first line is:

```
expected:<false> but was:<true>
```

This is followed by a *stack-trace* which is not particularly informative, but selecting the *Show Source* button highlights which assertion has failed in the source of the test class.

Exercise 7.21

The terminal window shows:

```
Testing the addition operation.
The result is: 7
Testing the subtraction operation.
The result is: 5
All tests passed.
```

There is no way to know (just from the test output) that the tests passed, because we only get the test result value. So we should not trust it.

Exercise 7.22

Calling `testPlus()` returns the value 1.

It is not the same result as was returned by `testAll()`.

Calling `testPlus()` one more time returns 7.

It should always give the same answer.

According to the source code it should return 7.

Exercise 7.23

No.

Exercise 7.24

The `testMinus()` method is similar in structure to the `testPlus()` method. Making a walk through of `testMinus()` obviously makes us want to take a closer look at the `minus()` method. Furthermore we could make a note to check that negative values are handled correctly.

Exercise 7.25

Method called	displayValue	leftOperand	previousOperator

initial state	0	0	''
`clear()`	0	0	''
`numberPressed(3)`	3	0	''
`plus()`	0	3	'+'
`numberPressed(4)`	4	3	'+'
`equals()`	7	0	'+'

Exercise 7.26

No. The `equals()` method does not does not explicitly check the value of the `previousOperator` when it is '-'. Therefore anything other than a '+' sign will result in a subtraction.

Exercise 7.27

Method called	**displayValue**	**leftOperand**	**previousOperator**
initial state	0	0	''
`clear()`	0	0	''
`numberPressed(3)`	3	0	''
`plus()`	0	3	'+'
`numberPressed(4)`	4	3	'+'
`equals()`	7	0	'+'
`clear()`	0	0	'+'

It is not in the same state as the previous `clear()` (the `previousOperator` differs).

This could have a big impact on subsequent calls. This is most likely the source of the inconsistent result we saw in Exercise 7.22.

Exercise 7.28

Two things has come to our attention while doing the walk through:

1. `equals()` does not explicitly test for the '-' value of `previousOperator`.
2. `clear()` does not clear all the fields.

The `clear()` method should be changed to this:

```
public void clear()
{
    displayValue = 0;
    leftOperand = 0;
    previousOperator = ' ';
}
```

The `equals()` method should be changed to this:

```
    public void equals()
    {
        if(previousOperator == '+') {
            displayValue = leftOperand + displayValue;
        }
        if(previousOperator == '-') {
            displayValue = leftOperand - displayValue;
        }
        else {
            //do nothing
        }
        leftOperand = 0;
    }
```

Exercise 7.29

Method called	displayValue	leftOperand	previousOperator
initial state	0	0	''
`clear()`	0	0	''
`numberPressed(9)`	9	0	''
`plus()`	0	9	'+'
`numberPressed(1)`	1	9	'+'
`minus()`	0	10	'-'
`numberPressed(4)`	4	10	'-'
`equals()`	6	0	'-'

The result should be 6, which is indeed what is in the `displayValue` at the end of the walk through.

Exercise 7.31

Yes. The output shows the same information as we collected in the table in the previous exercises.

Exercise 7.33

Debug statements have to be removed again.

Debug statements might be less error prone than the manual walk through.

Debug statements might be easier/faster to do (this is probably subjective)

Debug statements are easier to use when you need to verify a correction.

The activity of doing the manual walk through might give a better understanding of the program.

Exercise 7.35

See the project from the cd: `calculator-full-solution`

Exercise 7.37

The bugs in the *bricks* project are:

class `Brick`:

`getSurfaceArea()`: `side1` *should be* `side2` *in* `sum()`
`getWeight()`: `1000` *should be* `1000.0`

class `Pallet`:

`getHeight()`: `%` *should be* `*`
`getWeight()`: `+ baseWeight` *is missing*