Refactoring Activity 2

Here we are continuing the work we started in refactoring activity 1. We will spend some time working on the NumberPrinter class and breaking it in various ways.

Step 4: Working on the NumberPrinter class

Let's see if we can do some cleanup of the NumberPrinter class:

- The parameter to the print method is called primes, but it clearly can be any array of numbers. So perform a Rename refactoring to change the parameter to numbers.

- The other parameter is the number of numbers to print. We don't really need that any more technically, as we can simply use the length of the numbers array instead. To do this in a stepwise fashion, do the following:

  - Find the first use of the number-of-primes variable, and use "Extract Variable" to create a local variable equal to it, and named numberOfNumbers. Make sure to replace all 3 occurences.

  - Change the numberOfNumbers = ... assignment to instead be numberOfNumbers = numbers.length (The numbers in the array actually start at index 1) Run your tests to make sure they still work.

  - Now the parameter in the print method should be grayed out as it is not being used. Use the "Safe Delete" intention on it, and run your tests again.

- Let's continue with refactoring the print method into smaller parts. We start with the double for loop: Perform the Extract Method refactoring to it, to obtain a method printNumbersOnPage.

- It seems that the first group of System.out calls prints a header, so go ahead and extract the method printHeader from it.

- The remaining part of the while loops seems to update the counters as we advance through each page. It also seems that the methods we use all need those values. So we should elevate them to fields so that they can be more easily shared. Perform an "extract field" refactoring on the page number, page offset and number of numbers variables.

- Extract a method from the last three lines of the while loop, call it moveToNextPage.

- Now that we are using fields, the page number and number of number parameters in the printHeader method are no longer needed. Perform an "Inline" refactoring to them, and remove the superfluous this. parts that are added. Run your tests.

- Do the same for the printNumbersOnPage method, leaving just the numbers parameter to it.

- It feels that numbers should also be set to a field. Extract the field from the first occurence of numbers, in the initialization of numberOfNumbers. Make sure to tell it to replace all occurences.

- Run your tests and they should now fail. That's because IntelliJ did not actually add a statement to initialize this.numbers to equal the parameter numbers. Do so in the print method, and check that your tests are back in order.

- The numbers parameter in printNumbersOnPage should now be redundant. Perform the "Safe delete" intention on it.

- The first four lines of the print method are all about initializing fields. Extract them to a method initialize.

- The test in the while loop is determining if there are still more numbers to print. Extract it into a method needToPrintMore (keep the original signature when asked). This make it more clear what that test does.

- Let's shift our attention to the printNumbersOnPage method. Look at the conditional at the innermost level of the loop. Let's extract it to a method printNumberAt.

- Notice that this new method takes two parameters but really only uses one, namely their combination as "row offset plus column offset times rows per page". Extract parameter from that expression, and tell it to replace both occurences. It should also remove the other two parameters in this case.

- Let's look at the row-offset variable of the outer loop. It seems to be initialized as page offset, then stop at a boundary similarly depending on page offset. It is only used in the index computation. Change it so that it instead starts at 0 and ends at rows-per-page minus 1, and change the index computation to include an additional page-offset. Run your tests to make sure they still pass. Maybe also perform a Rename refactoring to now call the variable row.

- Looking at the stopping tests in our for loops, we are more used to seeing them with a less than comparison, rather with a "less than or equal to the number minus 1" comparison. So fix those up, and run our tests to make sure they still pass.

Step 5: Reducing the number of fields

It seems there are still too many fields in the NumberPrinter class. A primary example is the page offset. It should be simple enough to compute the page offset from the page number, and it is not really used in too many places. We would like to replace it with a computation. In order to achieve this, we will do the following:

- Use the "Encapsulate Fields" refactoring to encapsulate the get access for page offset. This will replace all accesses to this page offset by a call to getPageOffset(). Run your tests to make sure we didn't break anything yet.

- Now, we can edit the getPageOffset method to instead compute the offset from the page number and return it. The computation should be "page number minus 1 then times rows per page and times columns per page, then add one to the result." Run our tests to make sure this change did not break anything.

- Now, the page offset field should appear grayed out in its declaration point. Go ahead and do the "Safe delete" intention and run the tests again.

Next we have the number of numbers field. It's not used much, and we can just compute it from the numbers array instead. So let's do that:

- Use the "Encapsulate Fields" refactoring again, this time to encapsulate the getters of the number of numbers field.

- Replace the body of the new getter to instead return the numbers array length minus one. Run the tests to make sure they still run.

- Notice that the numberOfNumbers field appears grayed out now, and perform the "Safe delete" intention. Make sure the tests still run.

Step 6: Parameterizing the title

The printing of the numbers includes some header information. The first part of that information is the title, the other is the page numbers. We should probably make the title into a parameter that our creators provide, as we don't know what kinds of numbers they would want us to print. We'll keep the page number logic as part of our work.

This all will happen in the printHeader method, which currently is a series of System.out.print calls. Our first task would be to bring them together.

- Step-by-step merge each of the first two calls into the next one, by prepending its string to the front of the argument. For example after the first step the first two line should have become one call, with argument "The First " + Integer.toString(getNumberOfNumbe Run your tests after each step.

- Do the same to bring the last two calls together.

- Eliminate the Integer.toString parts (leaving their arguments intact, letting the plus operator worry about adding strings to integers.

- The " --- Page " part belongs with the second statement, not the first, so move it over and make sure your tests still pass.

- You should now have two System.out... statements, the first one setting the document title, the other setting the page number. We now want to turn the document title into a parameter. Select it and perform "Extract Parameter", name the parameter title.

- Going up to the print method, the title shows up there instead. Do another Extract Parameter to lift it to a parameter of the print method. Your tests should now be failing. Go to the PrimePrinter method and change the call to numberPrinter.getNumberOfNumbers into a reference to the number of primes method instead.

- Back in our printHeader method, put the two String.out statements into one, then go anywhere in the string and use the Replace "+" with String.format intention.

Step 7: Make the main loop clearer

There is something bothering us about the current structure of the main loop: It is supposed to be printing the next page every time, yet somehow its current structure doesn't allow for that. Part of the problem is that the page number is currently a field value, and getting updated in mysterious intervals: It is initialized in the initialize method, though nothing about the name of that method suggests that, then is updated at the end of the while loop, which feels a bit backwards. Ideally our loop, and print function, should say:

```
while there is a next page:
    print the next page
```

Even better, we should be able to simply say:

```
for page in pages:
    print page
```

Or in Java syntax:

```
for (int page : getPages())
    printPage(page)
```

In order to achieve this, we need to have an iterator. But before that, we need to have the page number as a parameter to the methods that form our for loop. The page number is used in a number of places:

- printHeader uses it to print the page number on the header.

- printNumbersOnPage uses it in its getOffset calculation.

- moveToNextPage actually increments it, which complicates matters considerably.

Let's work through this refactoring:

- We start with printHeader. Find the use of pageNumber in printHeader and perform Extract Parameter on it. Check that your tests still pass.

- Then, inline the moveToNextPage method and remove it. It won't really be doing much after we move around the page increment, so we'll just find a better place for the System.out call later. Do the same for the initialize method.

- Then look at the getPageOffset method, and perform "Extract Parameter" on the pageNumber variable there. Then move to the printNumbersOnPage method and perform "Extract Parameter" on the pageNumber variable from there.

- Finally, the needToPrintMore method also uses it via getPageOffset, so perform an Extract Parameter from there as well.

- Now we hopefully have isolated all the changes of the pageNumber field to the print method. It is set to 1 at the beginning of the print method, then incremented later on. You can confirm that the field is not used elsewhere by moving your cursor over the field declaration and using the "Navigate -> Declaration menu item". It should show you all the usages.

- Now with the cursor on the field declaration, choose the "Convert to local" intention. Then run our tests again to make sure everything works fine.

Thinking through the problem more, it almost feels like we need a separate class to capture the idea of the individual *pages*. Then that class can incorporate the logic about computing indices and knowing when it's done, for example. Perhaps we can call this new class a Page.

TODO