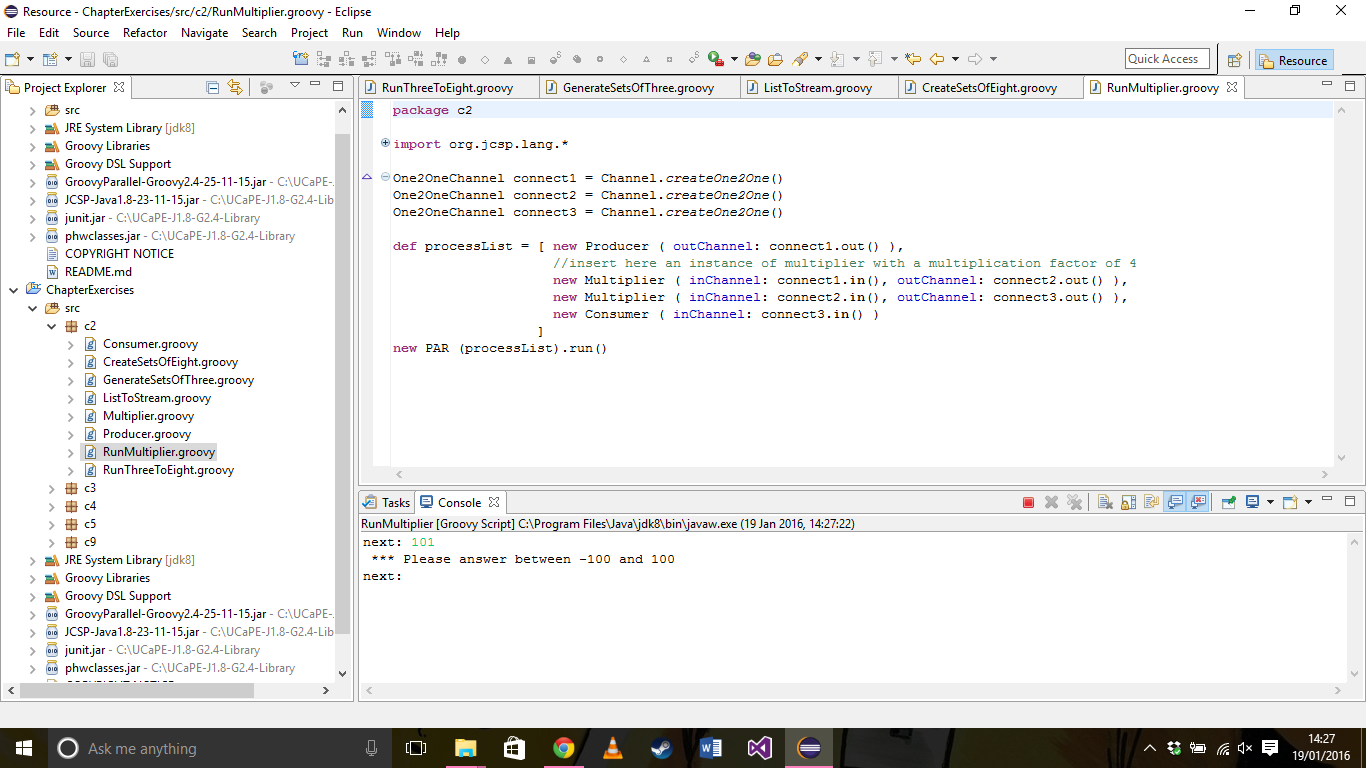
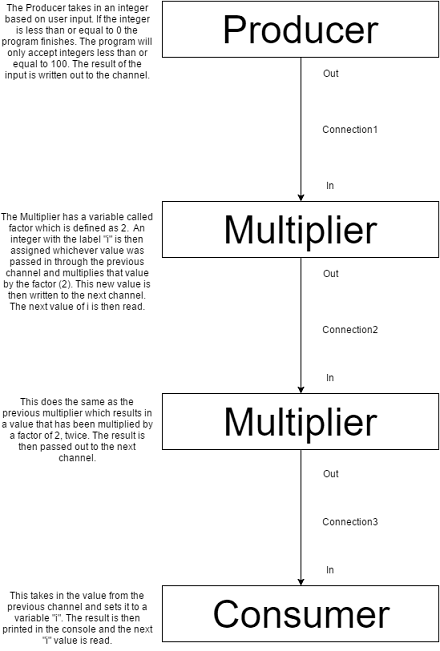
Fundamentals of Parallel Systems Lab Book

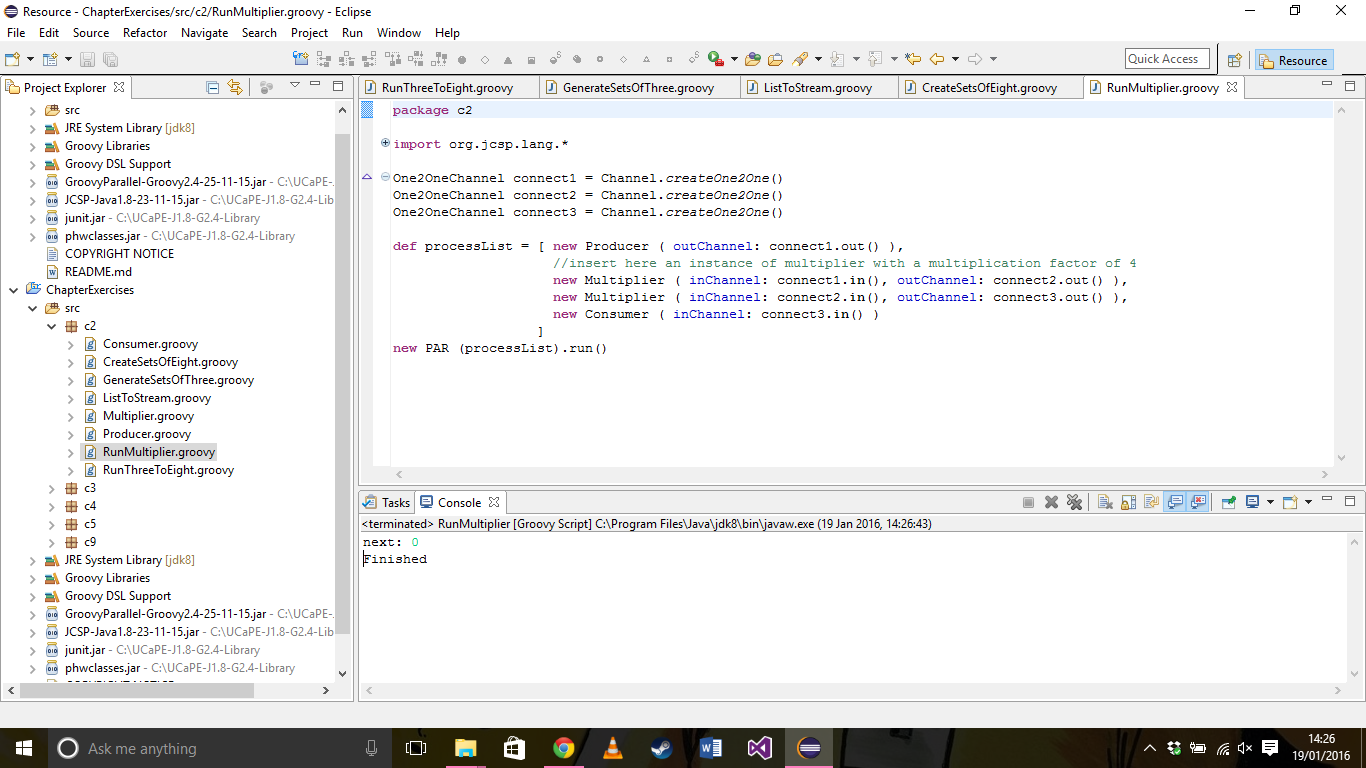
Jordan Stephano Gray

40087220

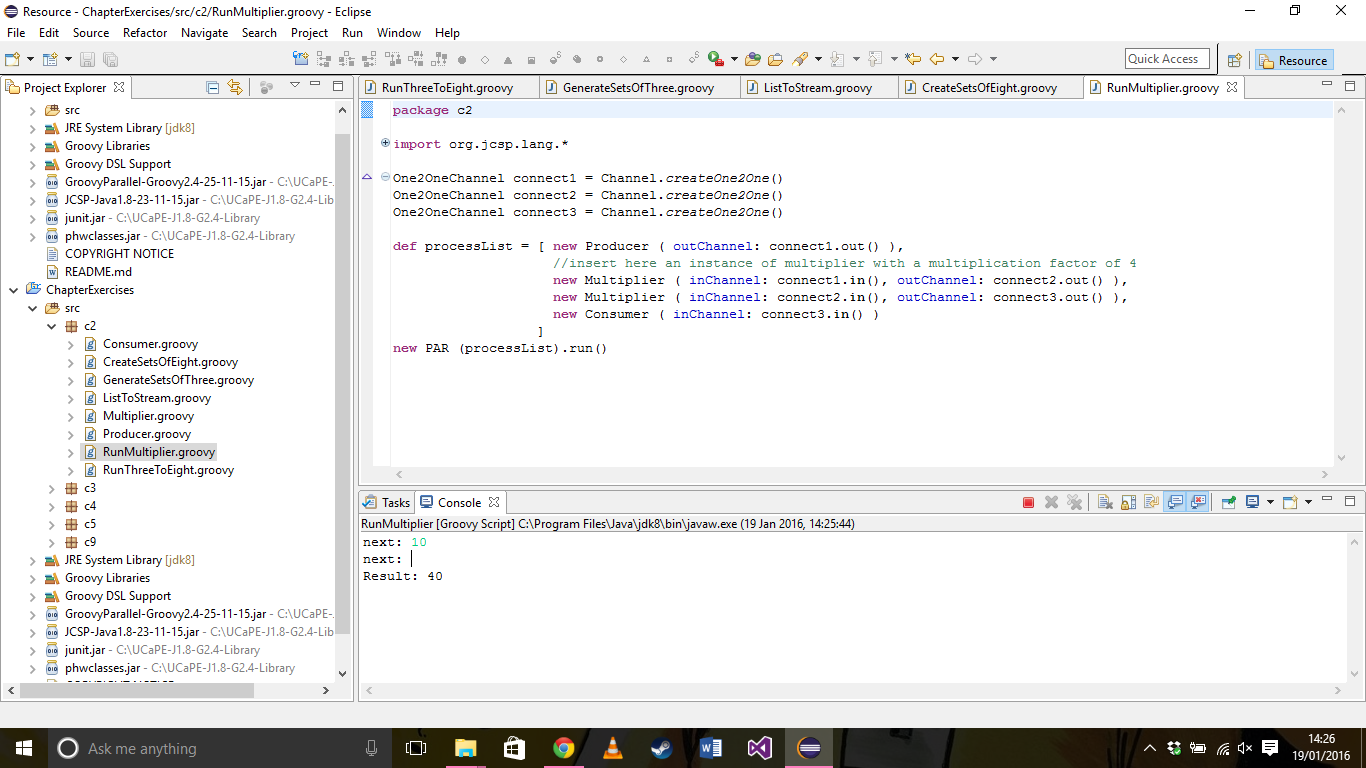
Exercise 2.1



This screenshot demonstrates the programs output when a number outside of the range is used as an input (-100 – 100).



This screenshot demonstrates the termination of the program when 0 is taken in as an input.



This screenshot demonstrates a working example of the program, this takes in the number 10 and is multiplied by 4 therefore producing the output 40.

Exercise 2.2

1. What change is required to output objects containing six integers?

Modify the for loop used in “CreateSetsOfEight” to loop for every index from 0 to 5 rather than 0 to 7.

1. How could you parameterise this in the system to output objects that contain any number of integers?

Create an integer variable to hold the list parameter and substitute it into the for loop in “CreateSetsOfEight”. The variable is then easily changed in “RunThreeToEight”.

1. What happens if number of integers required in the output stream is not a factor of total number of integers in input stream?

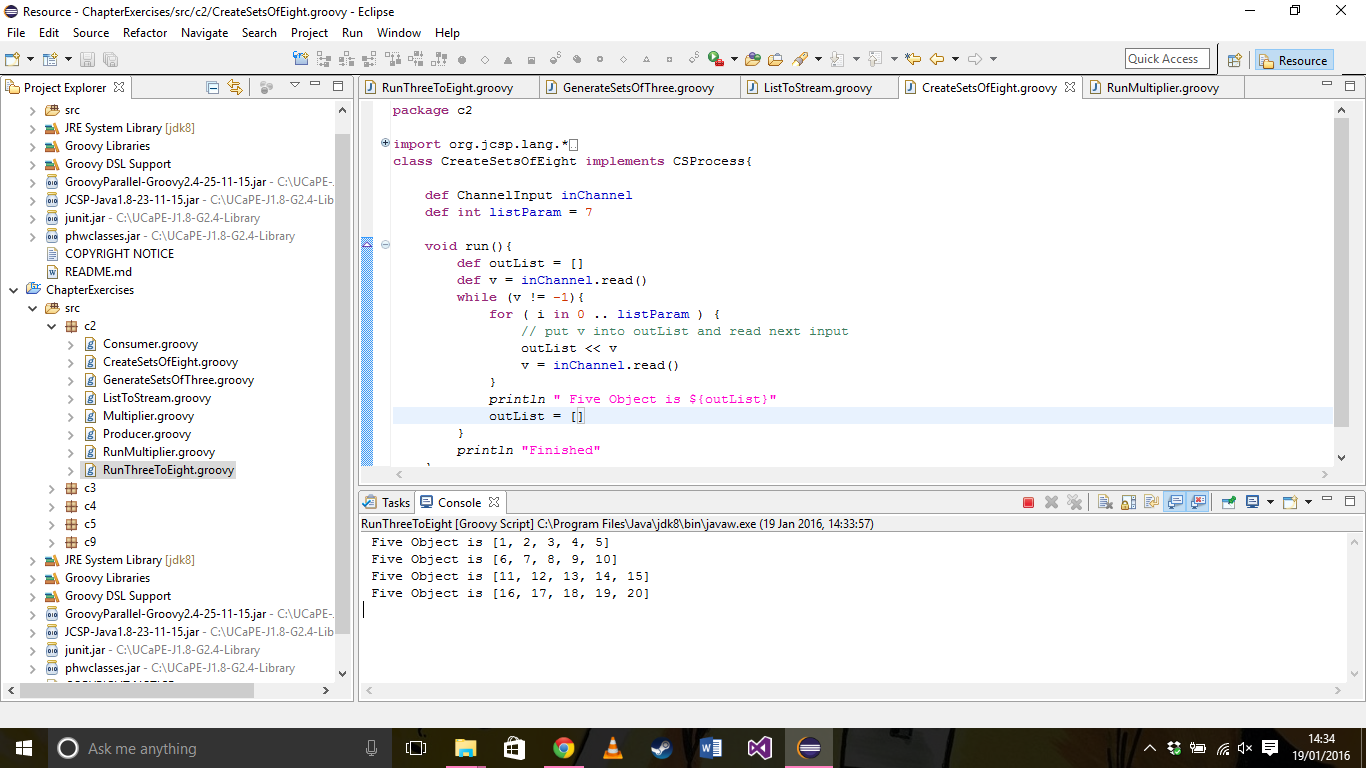
Any remaining numbers from input stream are left out of the output stream and the program can’t terminate.

C:\Users\Jordan\Downloads\Chapter2.2Diagram.png

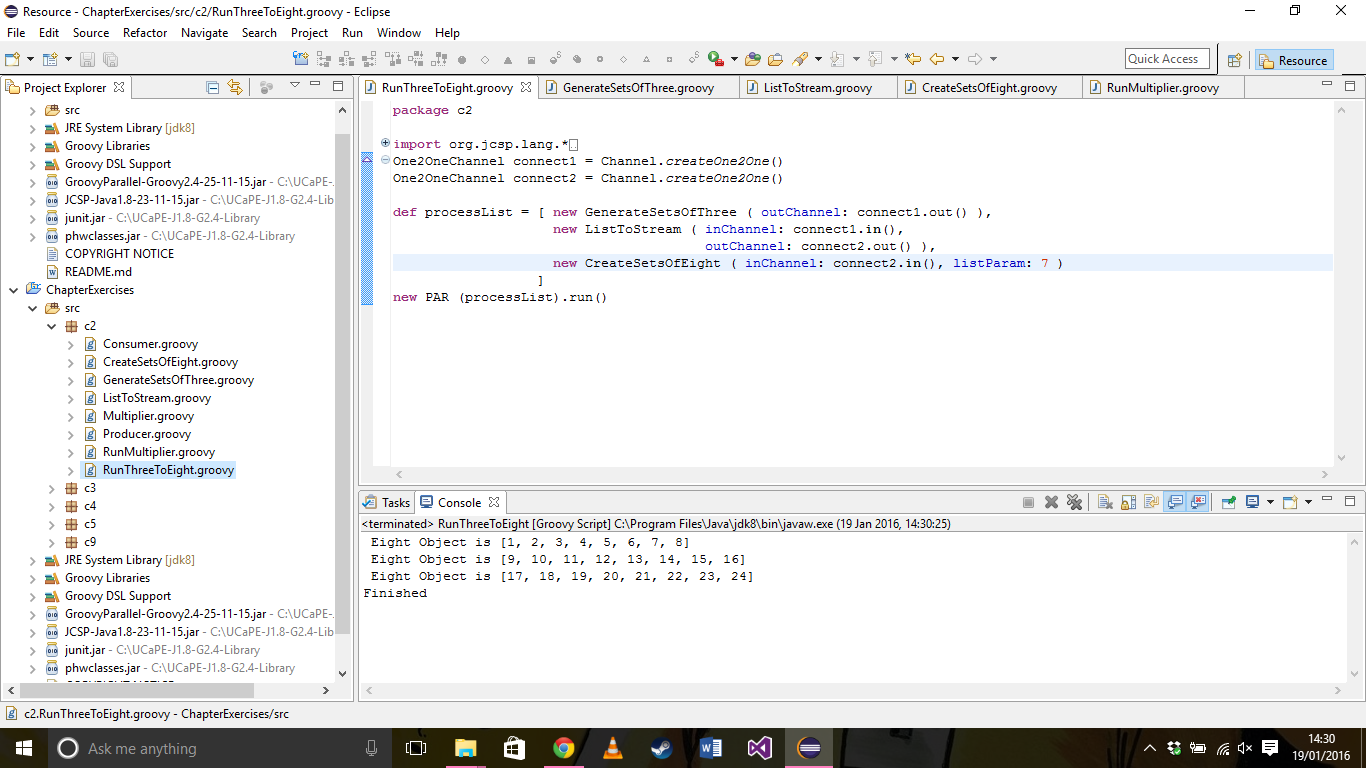
GenerateSetsOfThree defines a list with each index holding three numbers and writes this list to the outChannel.

ListToStream then takes this data through the inChannel and separates these groups of three into individual integers and streams them to the outChannel.

GenerateSetsOfEight then defines a new list to hold the new assortment of values, the listParameter is set to 7 which means the data will be split into sets of 8 (including index 0 – 7). After the values have been amended to the output list the list is printed of with a simple print statement.



This is an example of what happens when the number of integers required in the output stream is not a factor of the total number of integers in the input stream.



This demonstrates the working program outputting the data into sets of eight.

Exercise 3.1

I found that using the Minus method was more pleasing to me. Although using the Negator makes more sense by just inverting the value to a negative number although there is also an extra intermediate step.

Exercise 3.2

Exercise 3.3

Exercise 4.1

After this line has been copied out and the program is run it runs as usual counting upwards from the value of 1000. The change is apparent when an input is provided, what then happens is the program begins counting upwards from the new value whilst alternatively still counting upwards from the old. For example counting upwards from 1000 and 1 is provided as an input, the program will then count to 1001 then the next output will be 1 then 1002 then 2 and so on. When a third value is provided to the input field the program stops producing output altogether this results in livelock.

This happens because the program is cutting out processes and data can’t be passed on.



Exercise 4.2

This doesn’t solve the problem that was apparent in the previous exercise. What this does is allow the program to run through the processes one more time and produce two more outputs than previously possible before livelocking. This happens because…

Exercise 5.1

I tested whether the system ran expected of different delayed values demonstrated here in the table below.

|  |  |  |
| --- | --- | --- |
| QProducer Delay (ms) | Queue Delay (ms) | Total Time Taken (Seconds) |
| 0 | 0 | 1.85 |
| 500 | 0 | 26.77 |
| 1000 | 0 | 51.53 |
| 0 | 500 | 25.63 |
| 0 | 1000 | 47.49 |
| 500 | 500 | 26.69 |
| 1000 | 500 | 51.77 |
| 500 | 1000 | 51.53 |

These results show that the delay effects the program equally when it is imposed on either of the processes. This happens because the two processes loop until completion and the delays begin at the same time rather than one after the other so after the initial delay the following process is ready to take in the data and process it.

Exercise 5.2

Exercise 6.1

Exercise 7.1

Exercise 8.1

Exercise 9.1

Exercise 9.2

Exercise 9.3

Exercise 11.3