Fundamentals of Parallel systems

Lab Book

Conner Weatherston (40167111)

Exercise 2-1

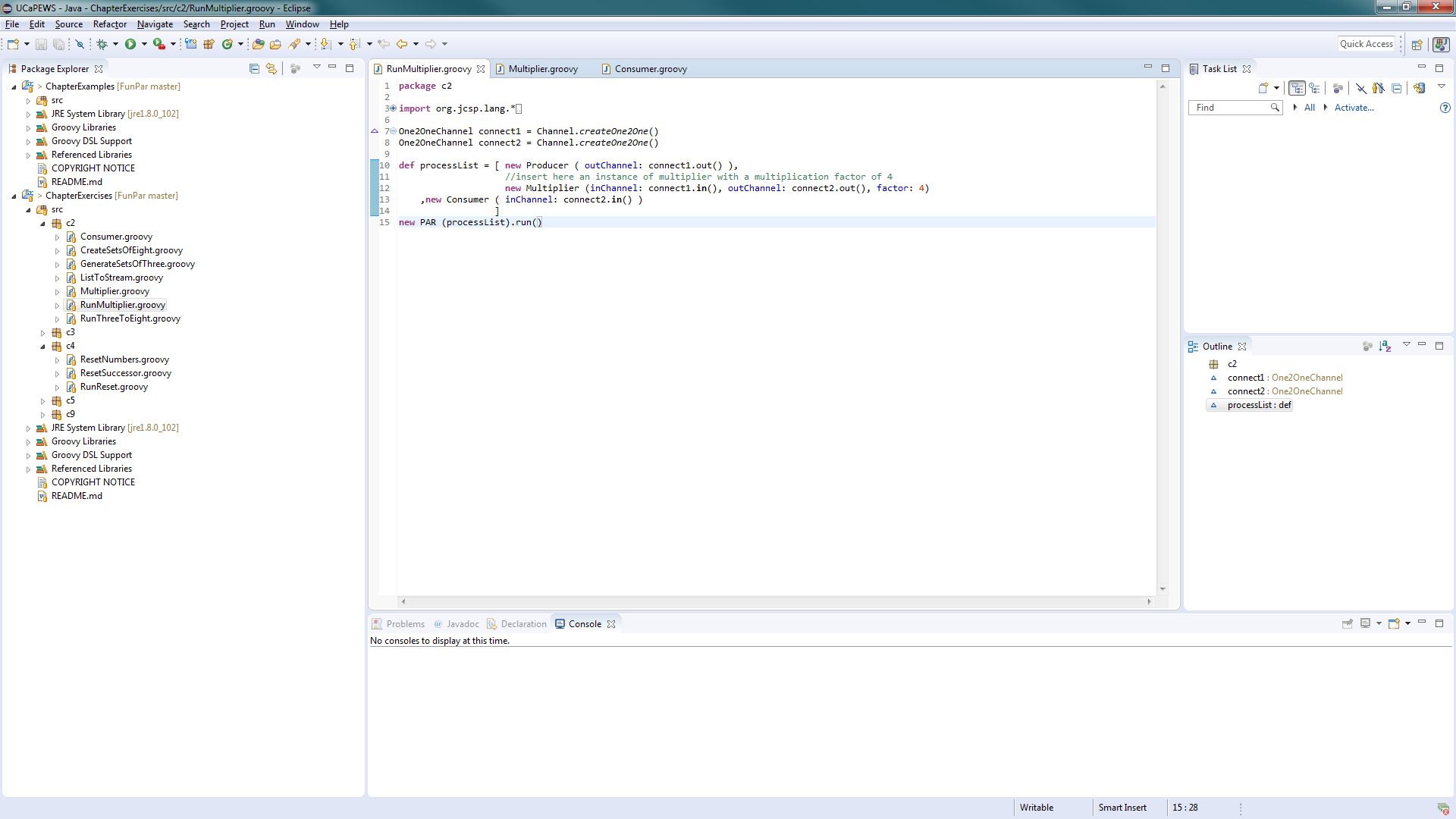
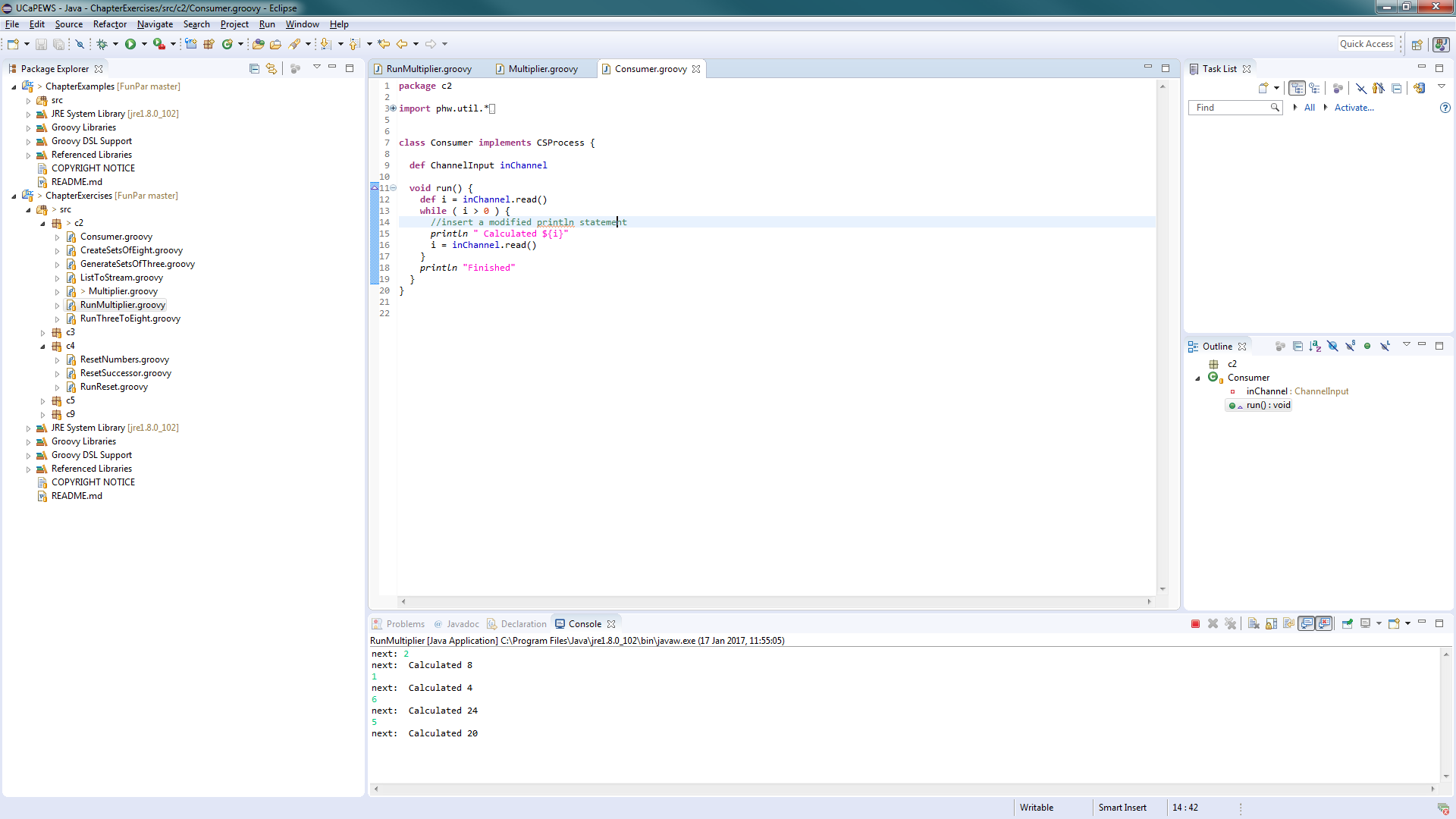
Producer

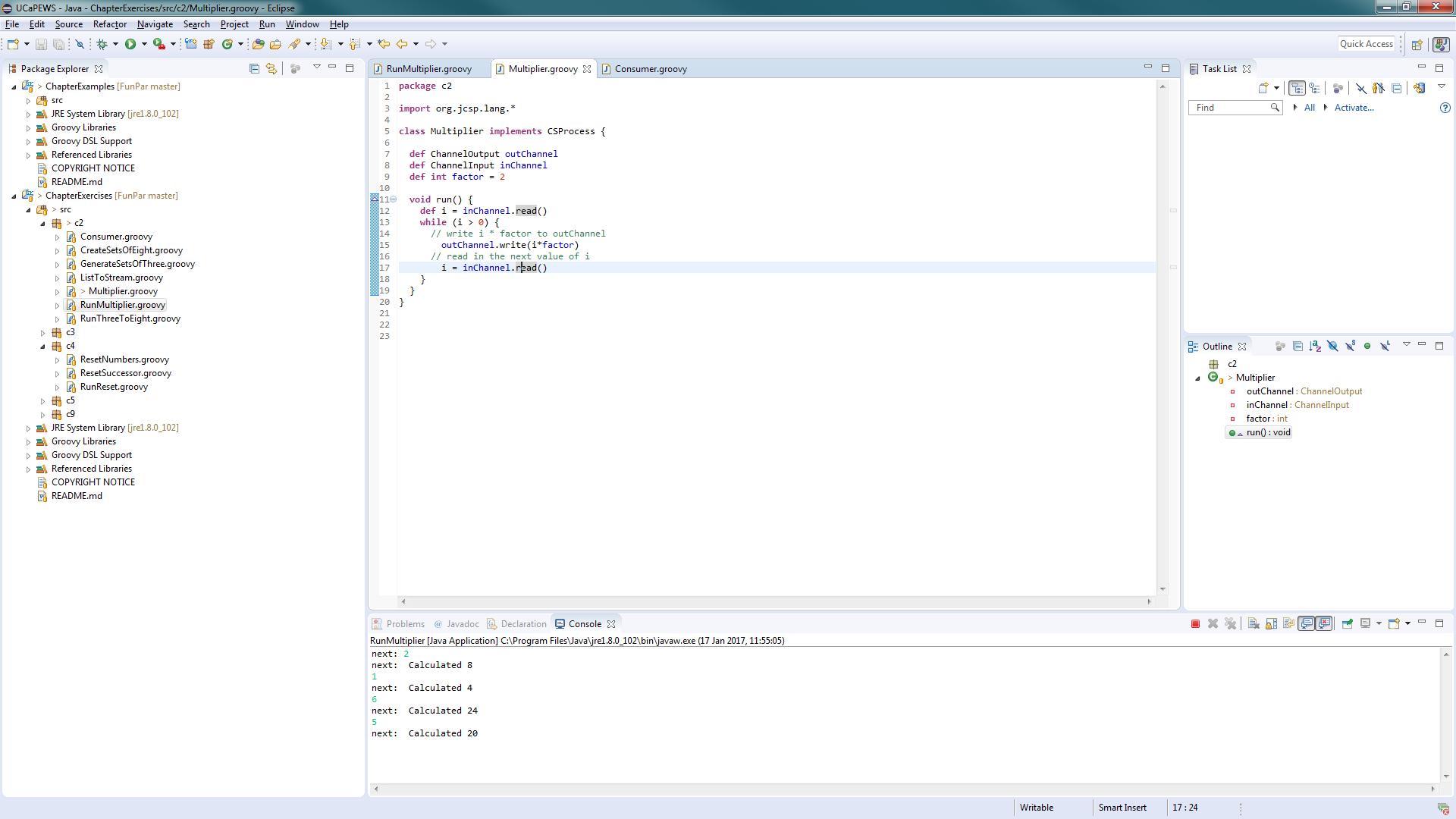
Multiplier

Consumer

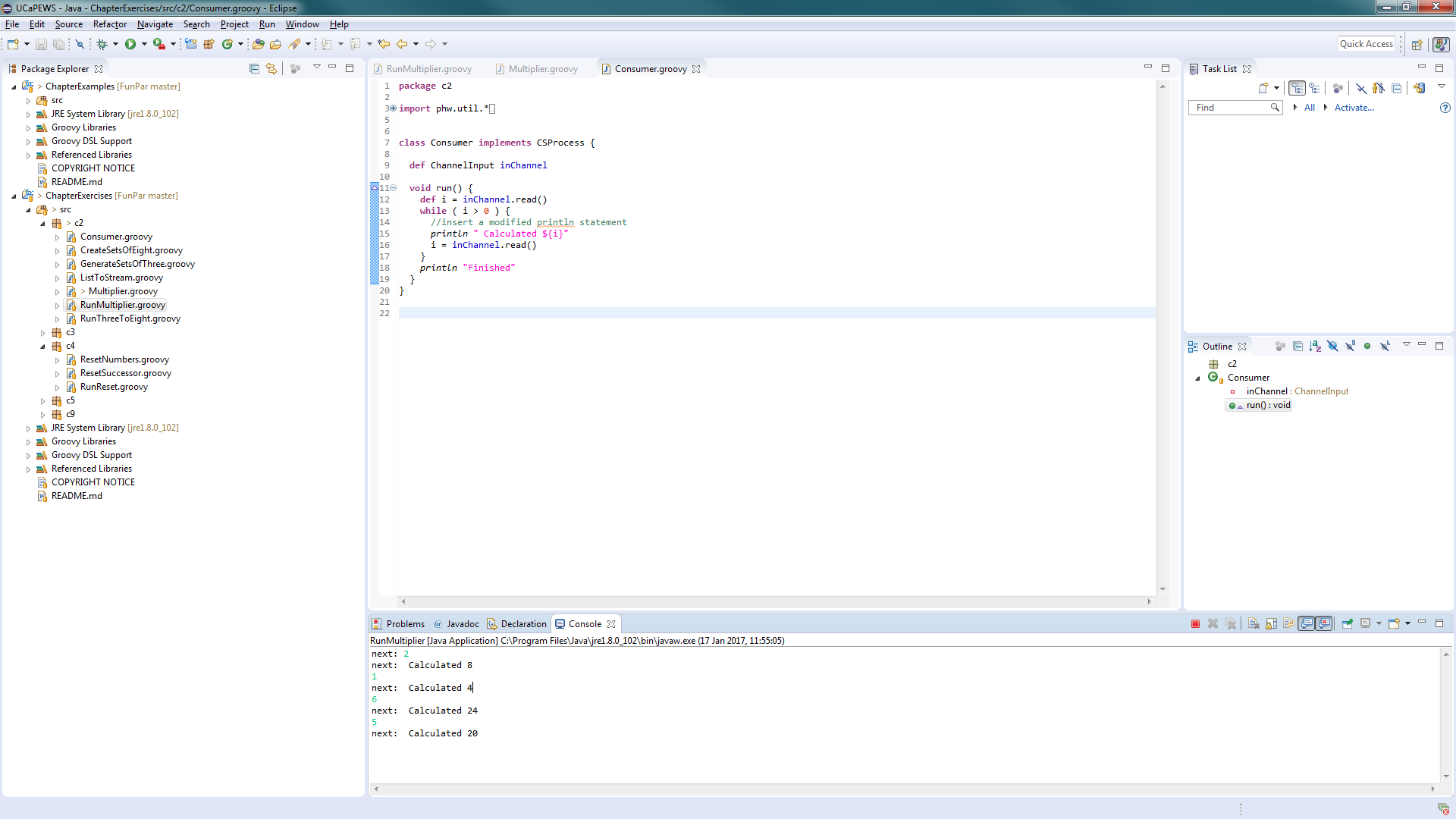
Channel 2

Channel 1





Output



Exercise 2-2

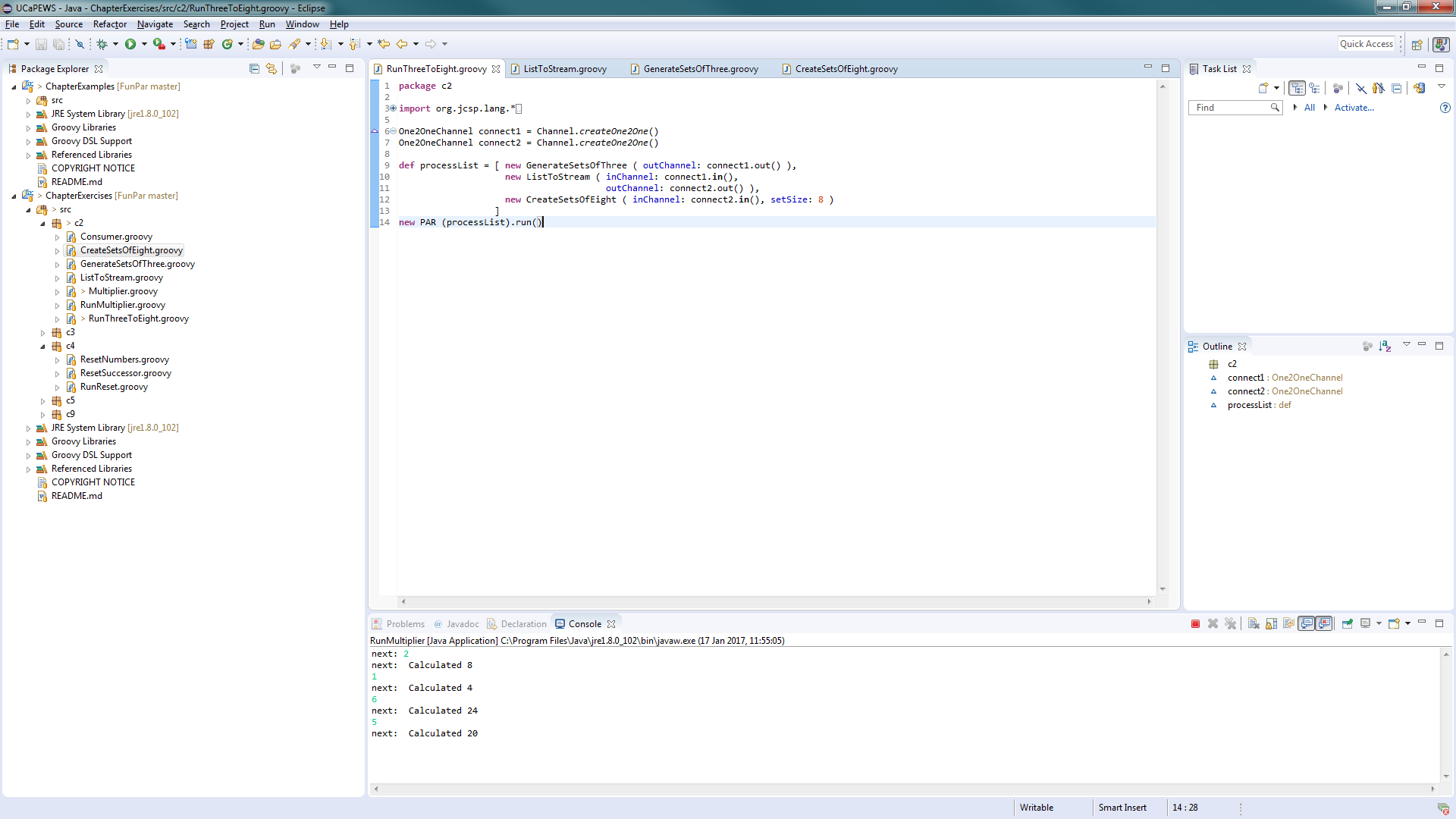
Channel 2

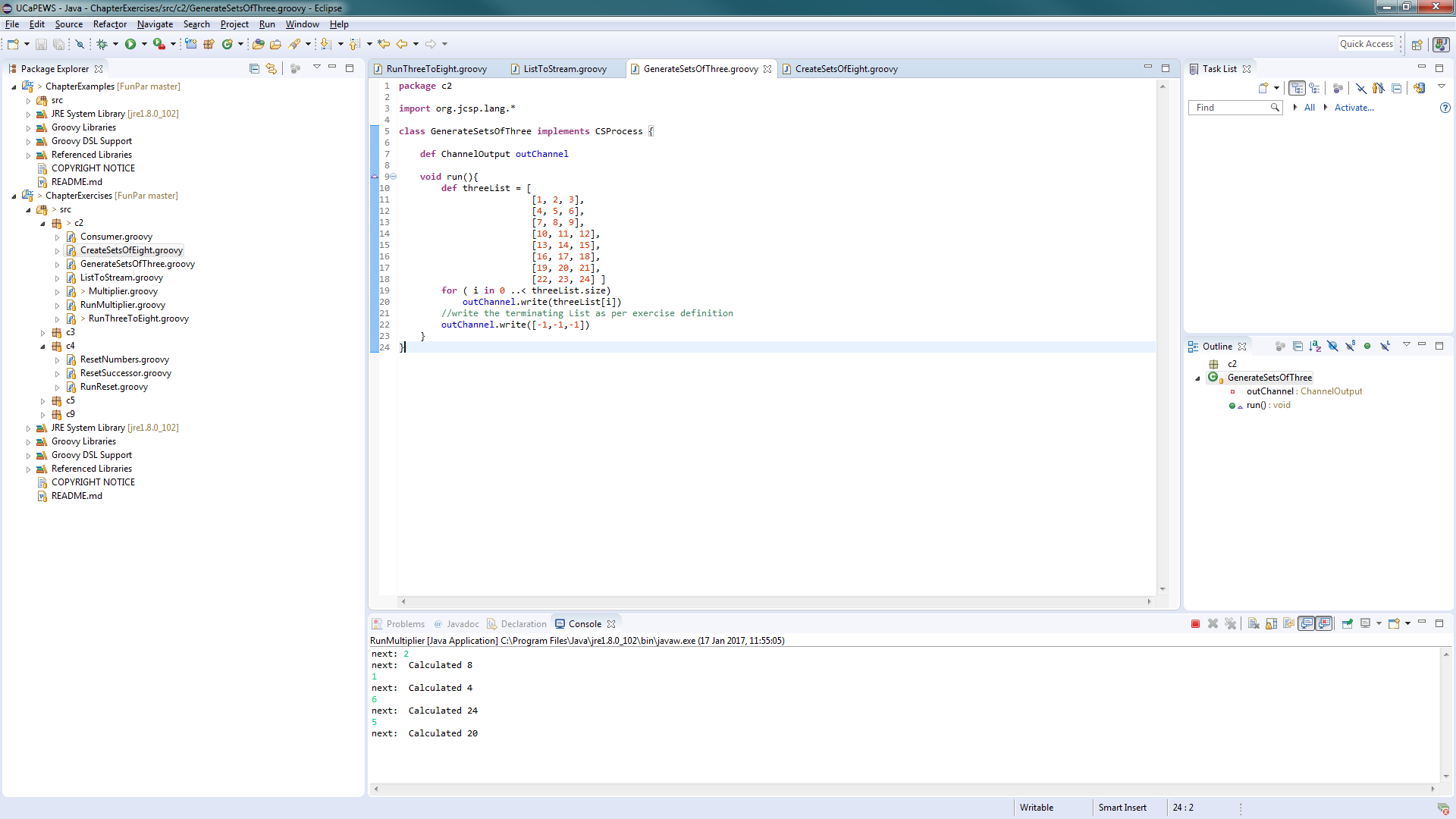
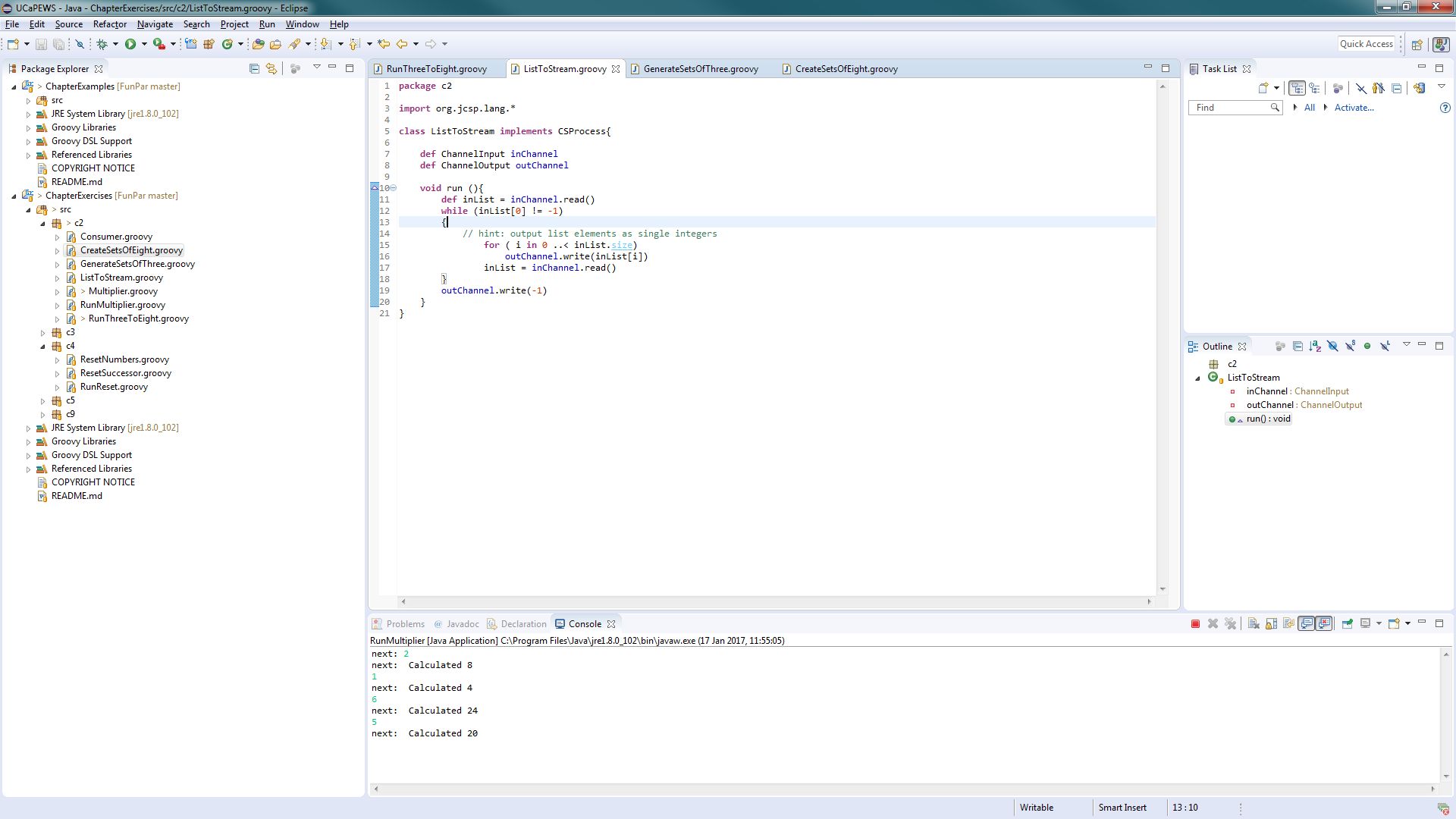
Channel 1

ListToStream

CreateSetsOfN

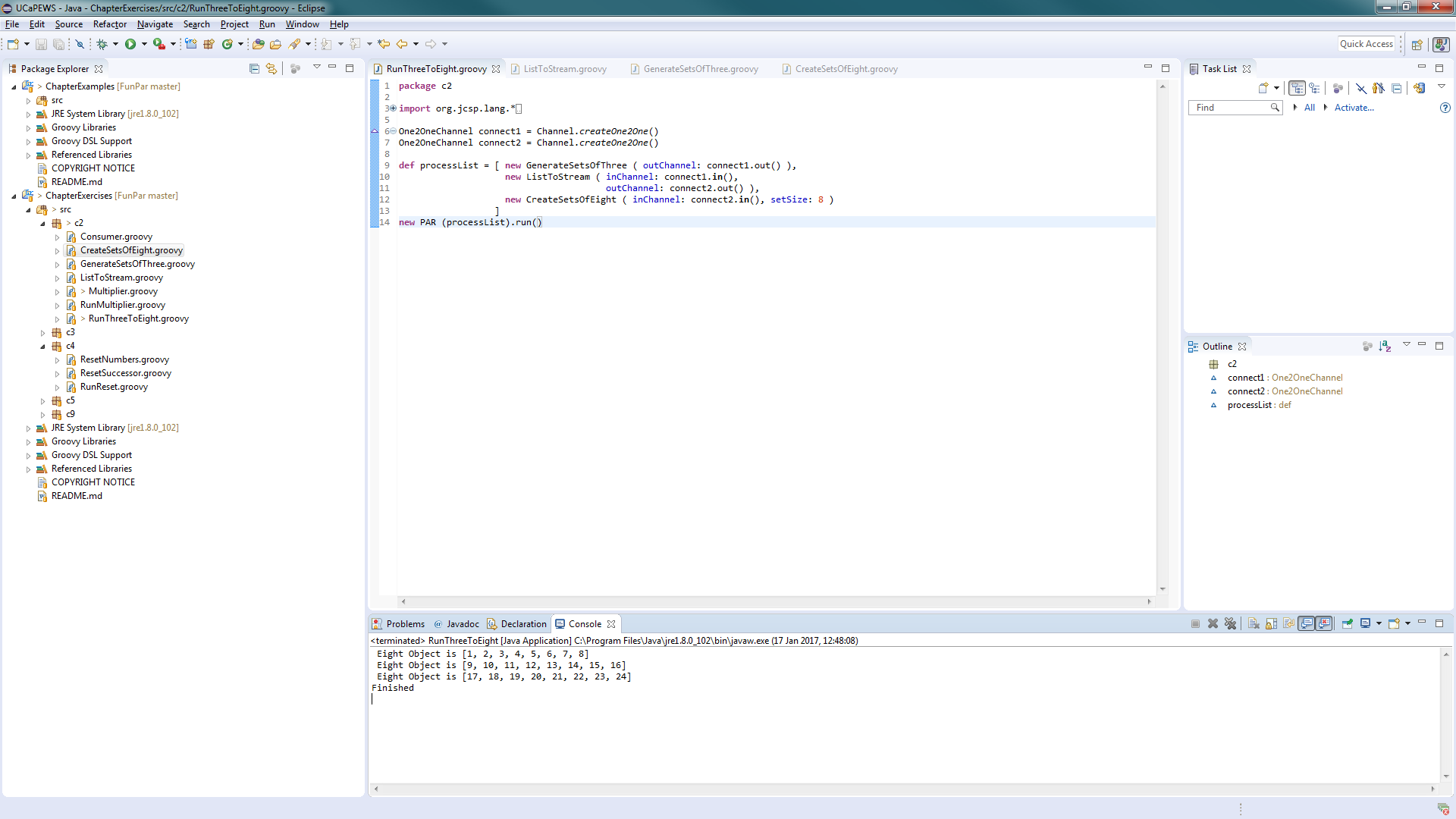
GenerateSetsOf  
Three







Output.



What change is required to output objects containing six integers?

* Change the for loop in the CreateSetsOfN to print out in a list of 6 instead of a list of 8.

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

* Create a variable to store what the size of the list is to be outputted.

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

* The program omits the remainder of the set that would be printed and cannot terminate.

Exercise 3-1

D2P

I2D

N2I

GPrint

GIntegrate

Differentiate

GNumbers

Differentiate

a

Minus

In

Out

b

c

GPrefix

GPCopy

DifferentiateNeg

a

d

In

GPlus

Negator

GPrefix

GPCopy

out

c

b

The original differentiate approach is better as it provides a more sensible solution to the problem. Also creating a minus process means it can be reused in the future. If we used the DifferentiateNeg approach we would have to ensure that the negator process occurs because the GPlus process in order for a minus to occur.

Exercise 3-2

GSPairsA

C

B

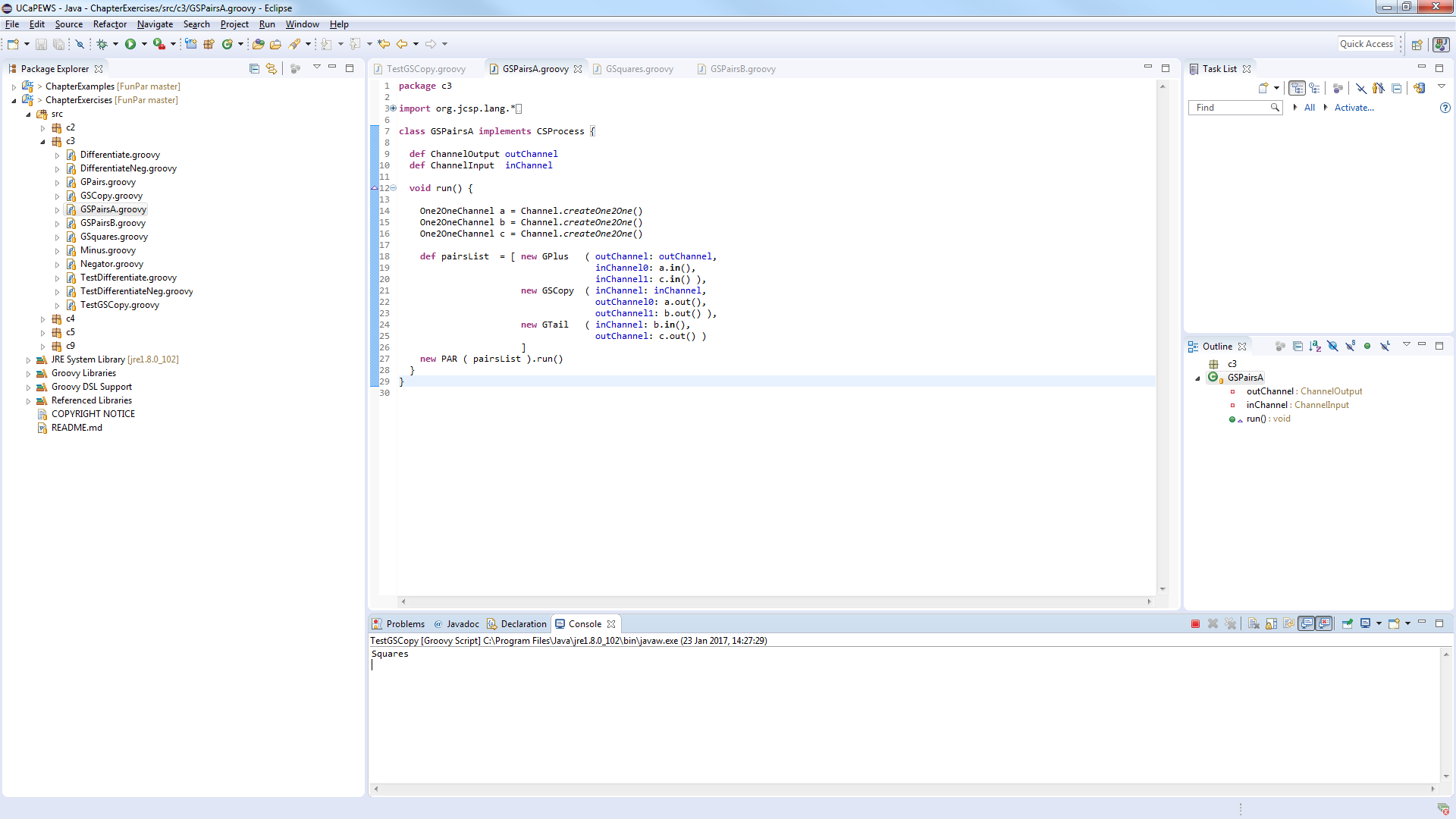
GTail

GPlus

GSCopy

A (this comes first)

Output



GSPairsB

C

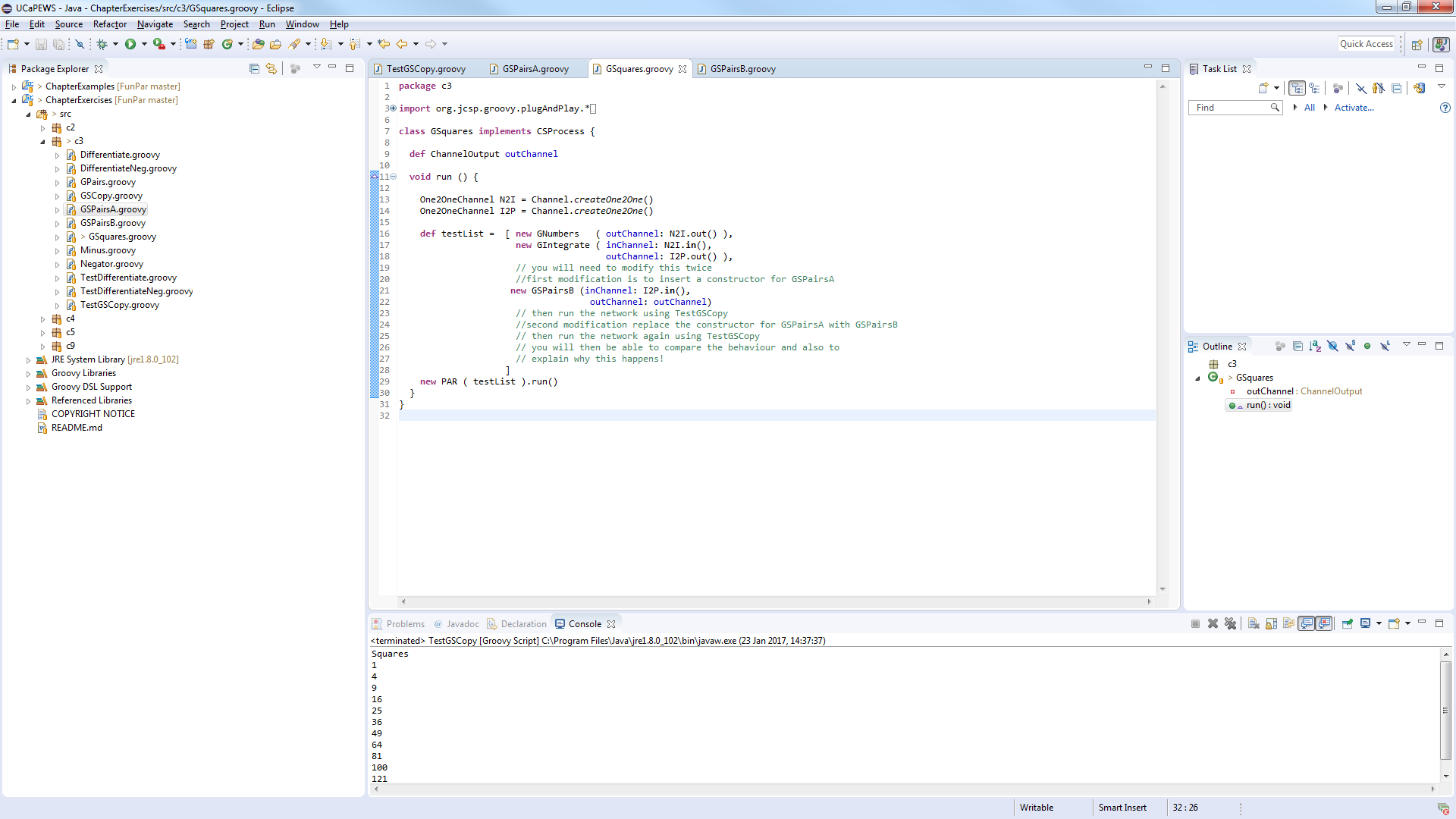
B

GTail

GPlus

GSCopy

A (B occurs first)

Output snippet

The results will keep going.

GSPairsA does not work as the process GPlus will keep asking for another value from the C channel. This input cannot be provided as C needs an value outputted from GTail. This process also cannot get a result as GSCopy cannot execute as there is still a value on the A Channel which cannot be read, thus the entire system fails.

GSPairsB can run as the order of the channels being written too sequentially is correct. As it writes to A first then goes into GTail. After that the value is put onto channel B which GPlus will then ask ask for a value from the C channel. Once this happens the next value from GSCopy can then be put onto the ‘a’ channel, as it is outputting sequentially and not in parallel. This value will go through the GTail process and then will allow the GPlus process to execute. Once it is executed then GSCopy can output onto channel B and the cycle continues.

Exercise 3-3

It reduces the number of processes required in order to print a series of results. This makes the system less complex and easier to understand.

Exercise 4-1

ResetNumbers

out

a

GPCopy

GPrefix

c

ResetSuccessor

b

reset

The system will eventually deadlock. This is due to processes not being able to finish executing thus the values are stuck in the process not allowing the cycle to continue.

Output

Exercise 4-2

Moving the reset channel to the reset successor process will not fix the issue that arose in exercise 4-1. The issue is there a value from another process trying to be written to the process where the reset channel is. This needs to be resolved or the issue in 4-1 will persist.

Exercise 5-1

Q2QC

QP2Q

QConsumer

Queue

QProducer

QC2C

Regardless of what delays that are put on the QProducer and QConsumer the output will always be the same. Which is printing out in the correct order (1 .. 50). The reason for this is the Queue process manages the information exchange thus it ensures that the data is processed in the correct order.

Print results with different delays here to show that I am correct.

Exercise 5-2

Controller

injector

suspend

factor

GFixedDelay

data

Scale

GNumbers

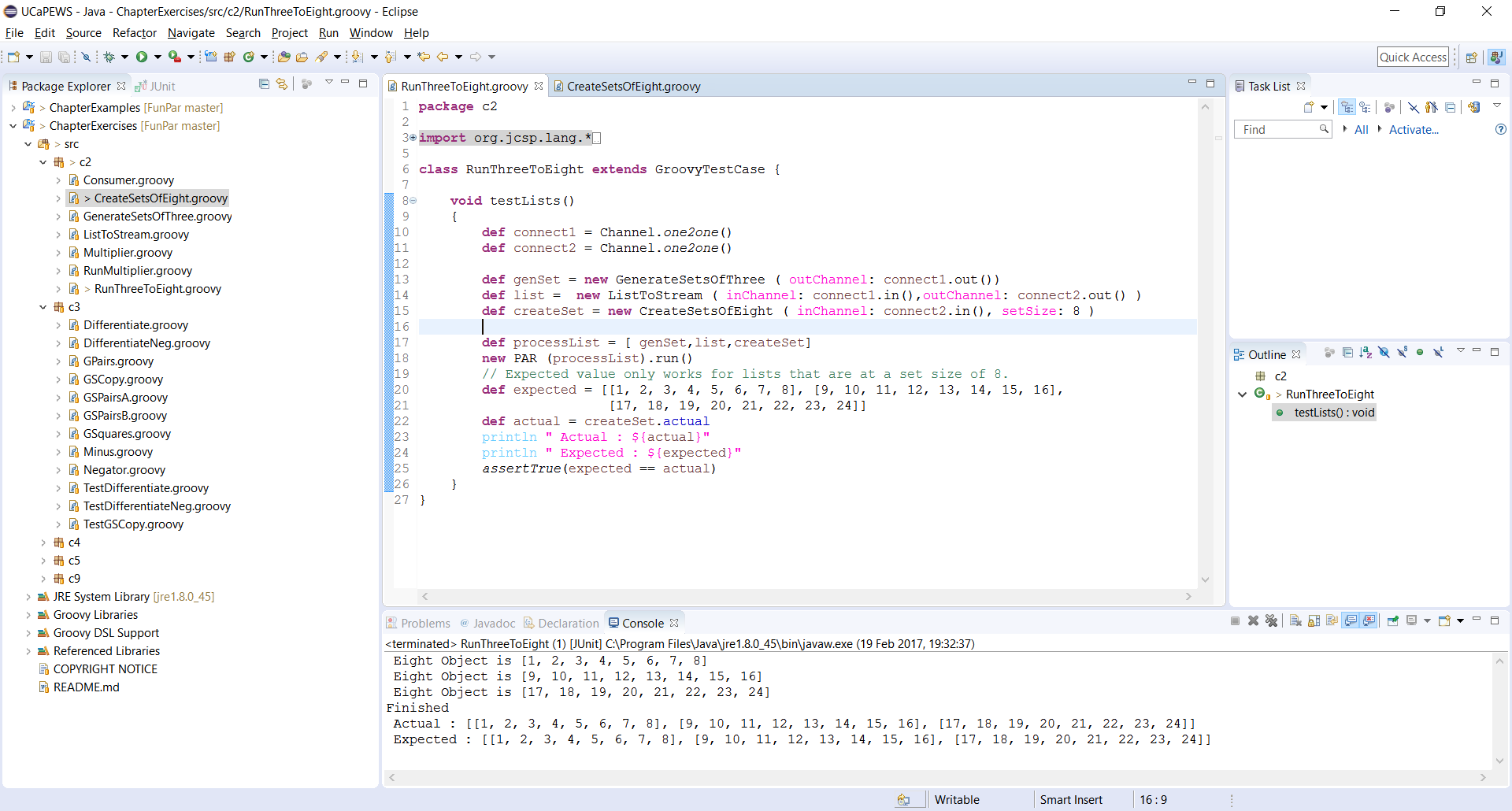
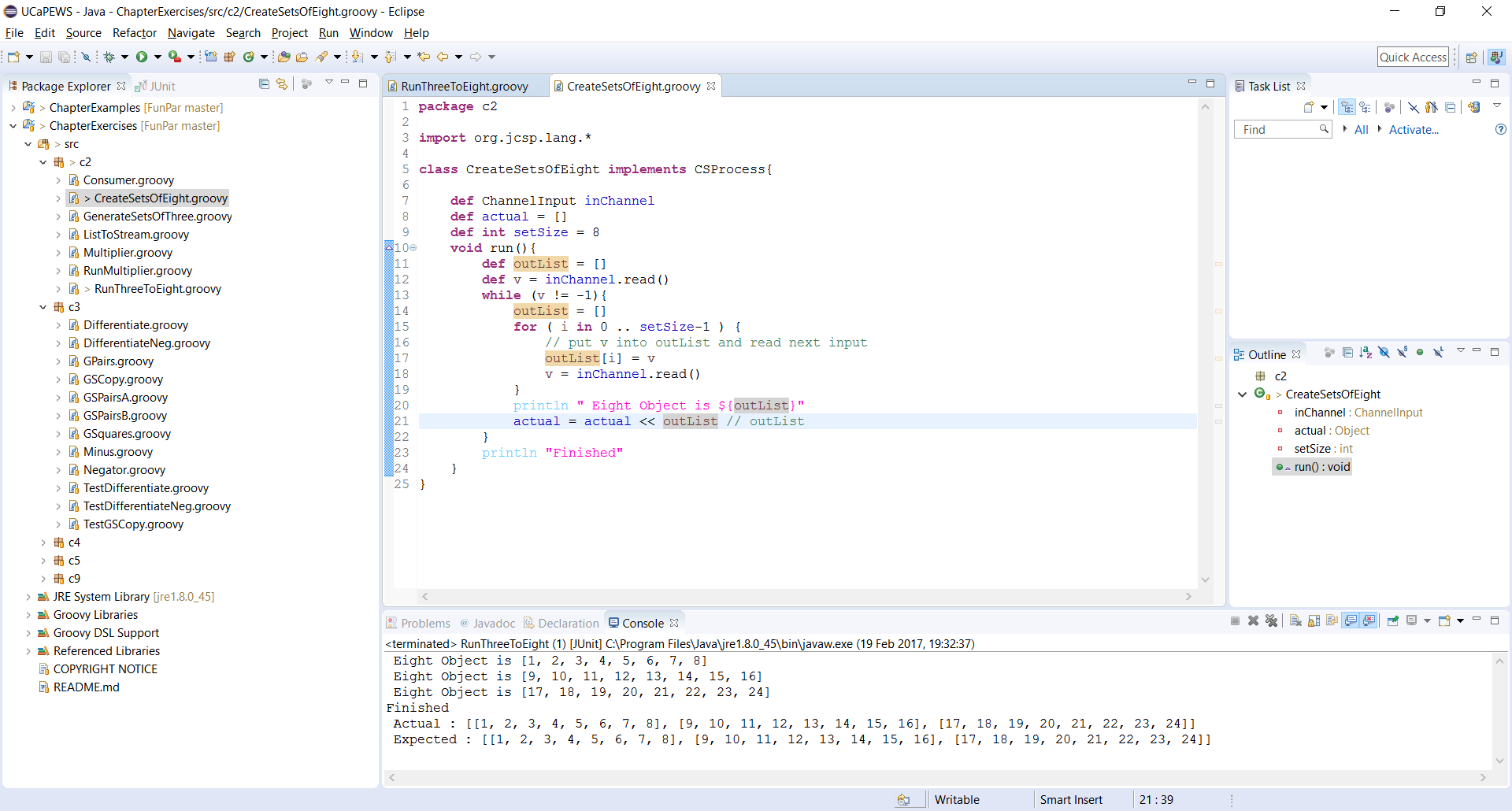
TimedData

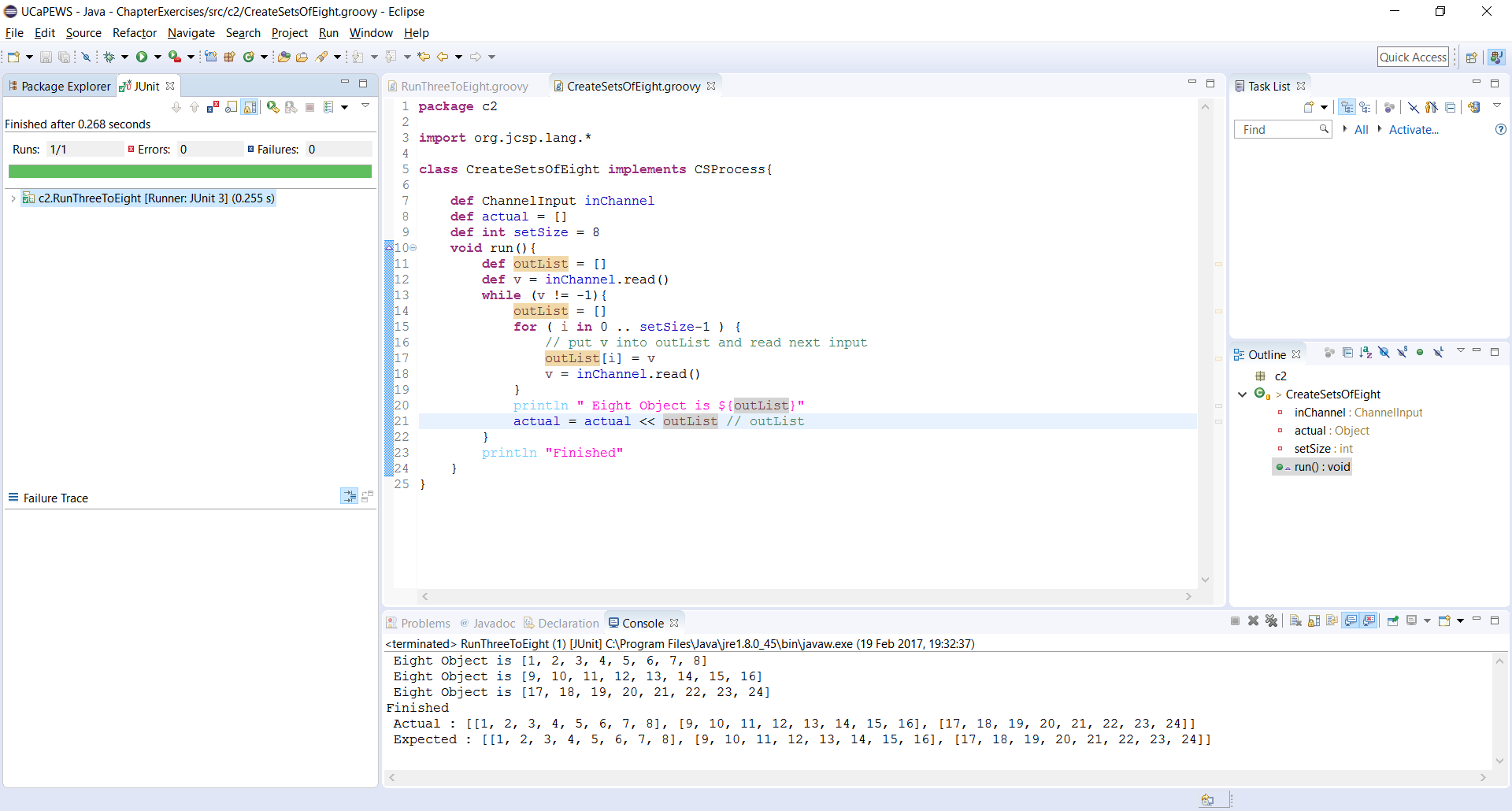
scaledData

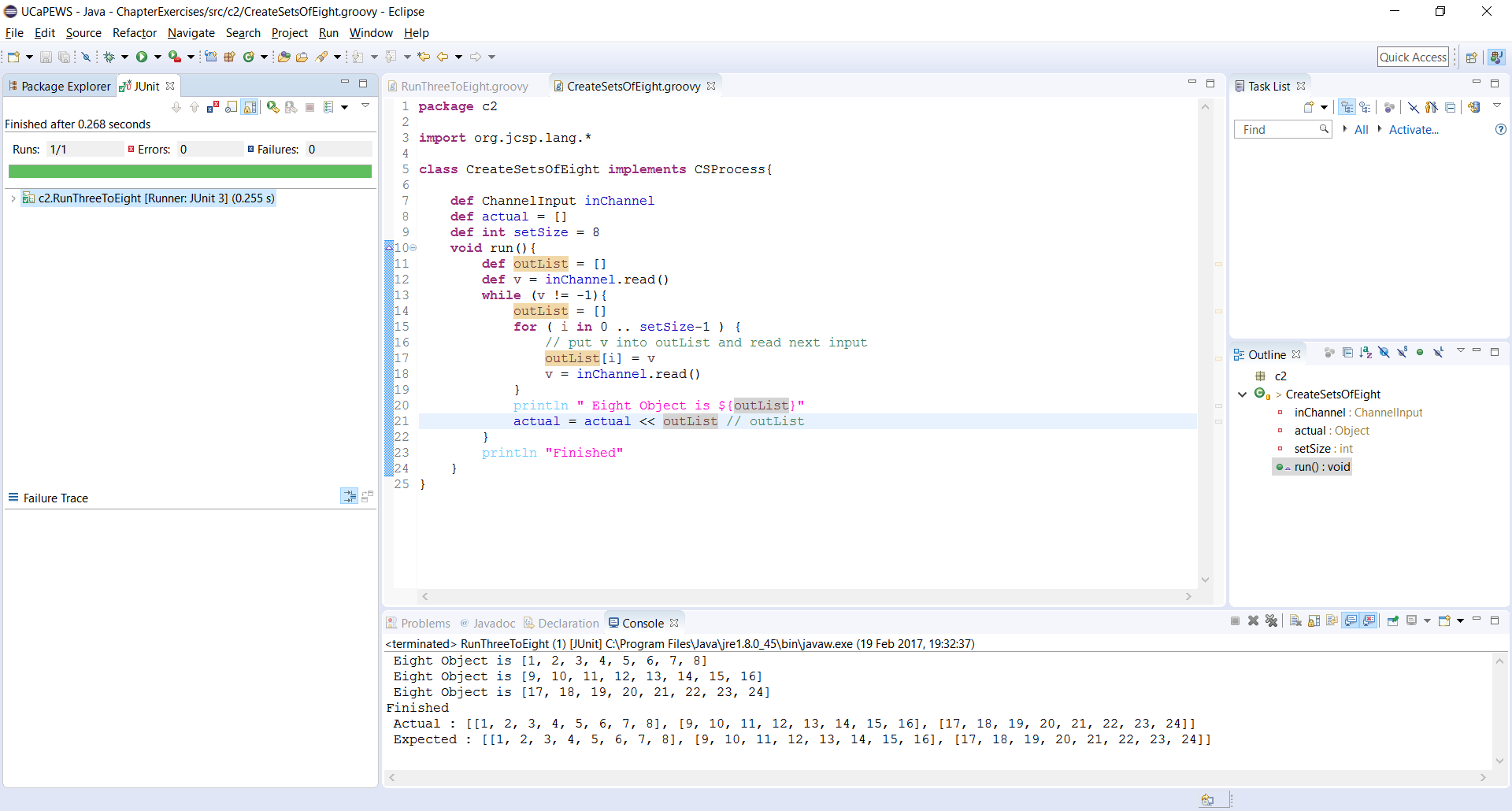
GPrint

Pre-conditions are the more elegant solution as it makes the solution more modular. Also It helps reduce complexity of the program as it eliminates the nesting issues that occurs with nested alternatives.

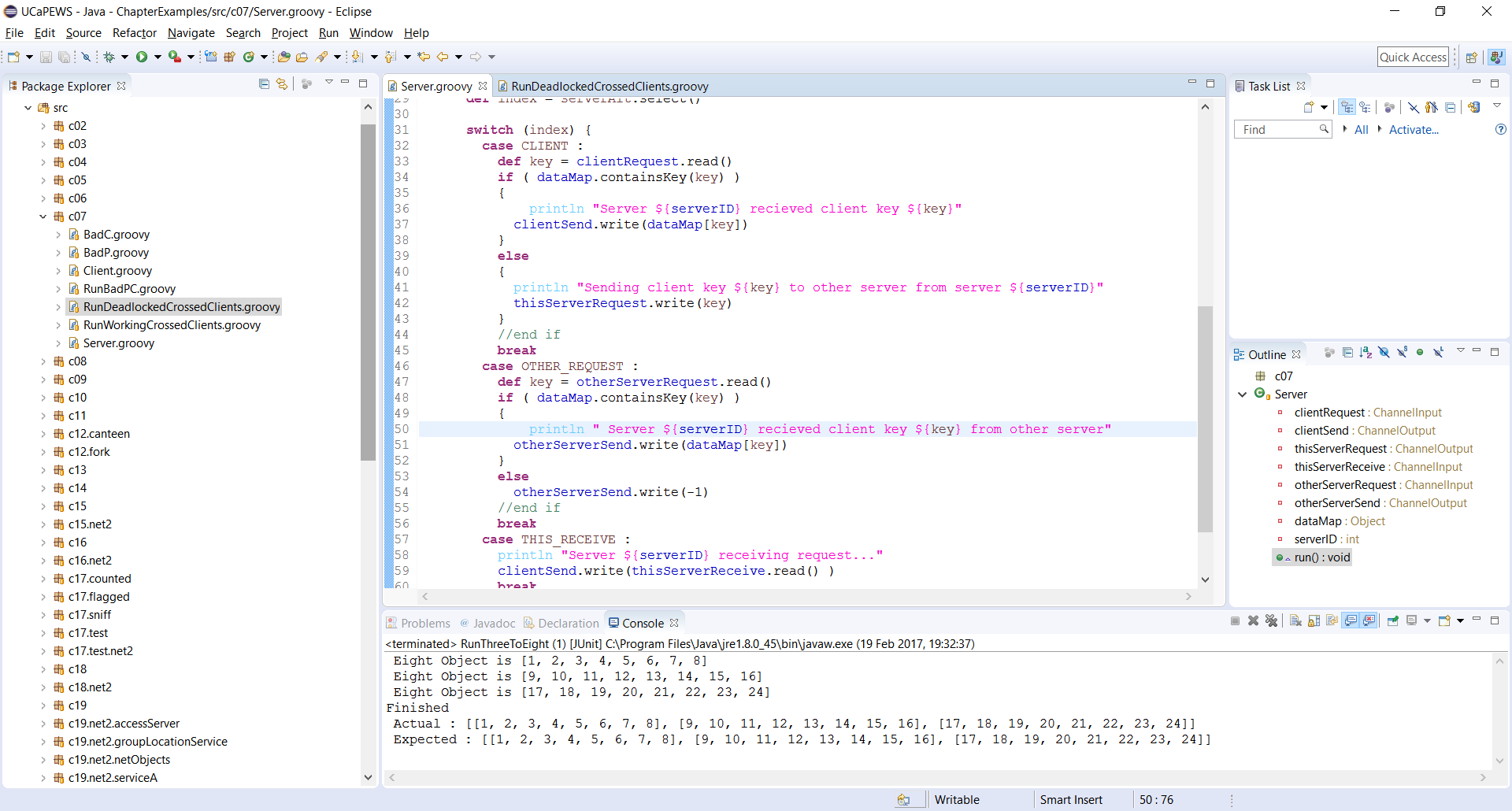
Exercise 6-1

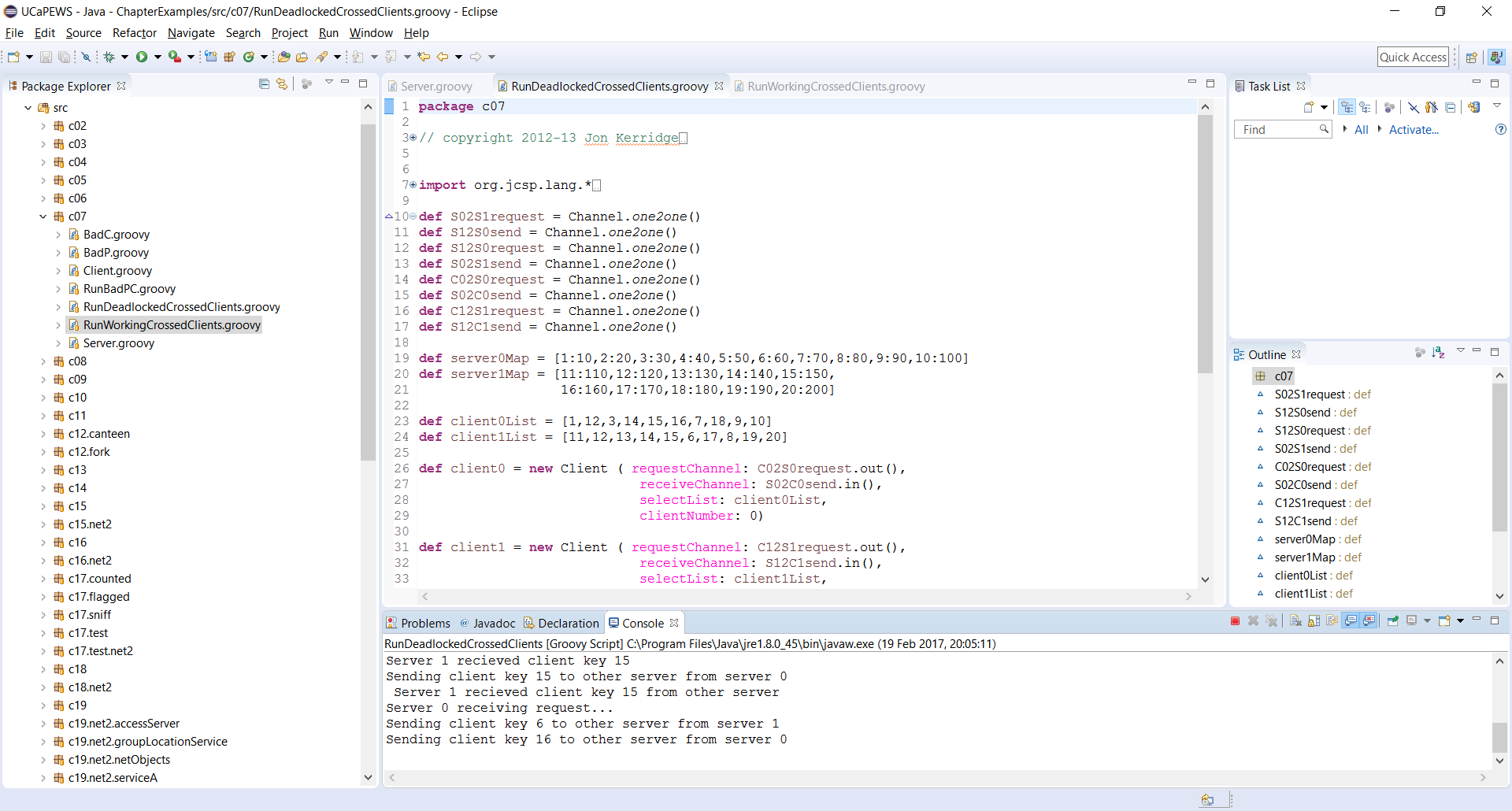


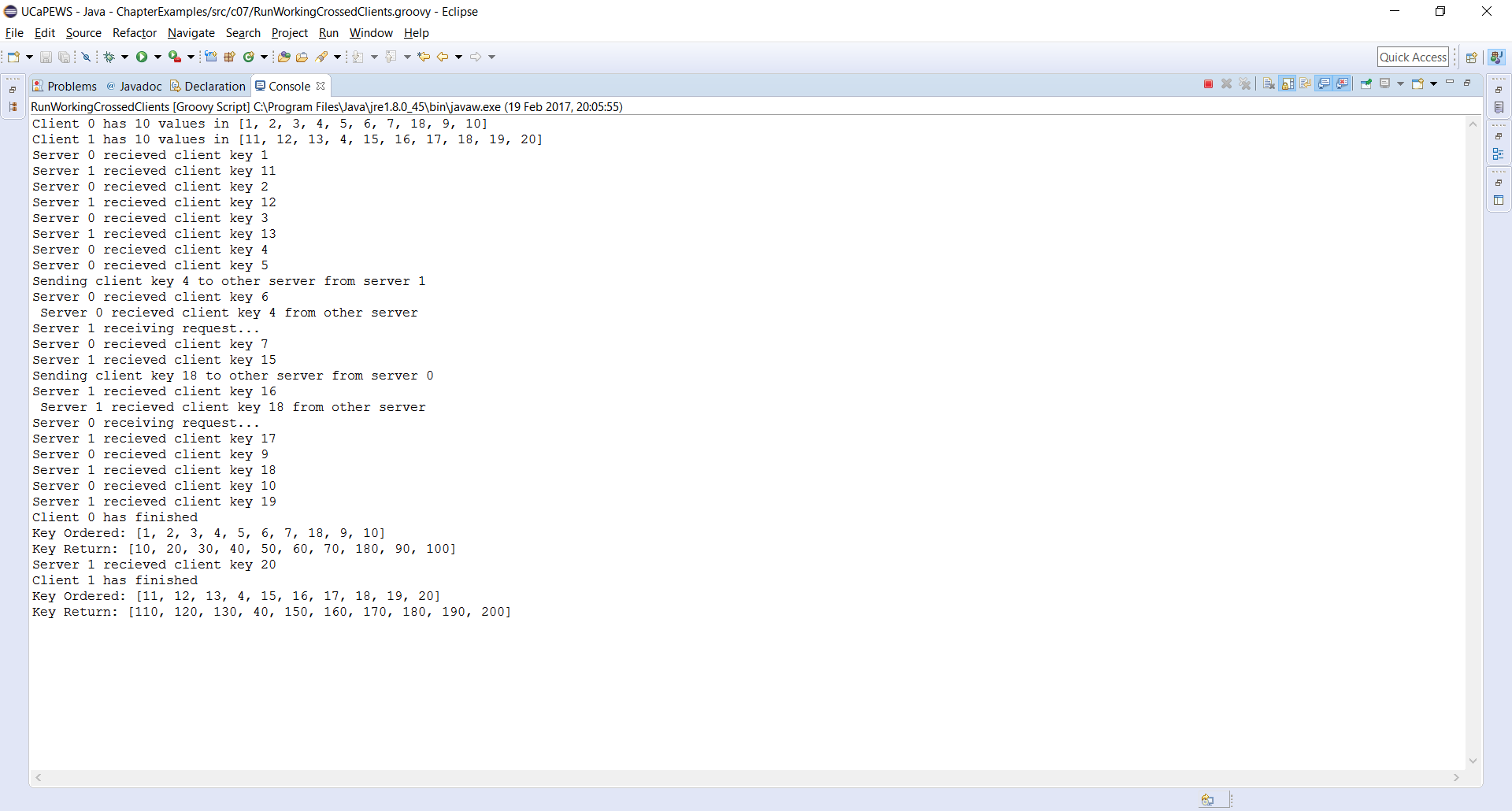




Exercise 7-1



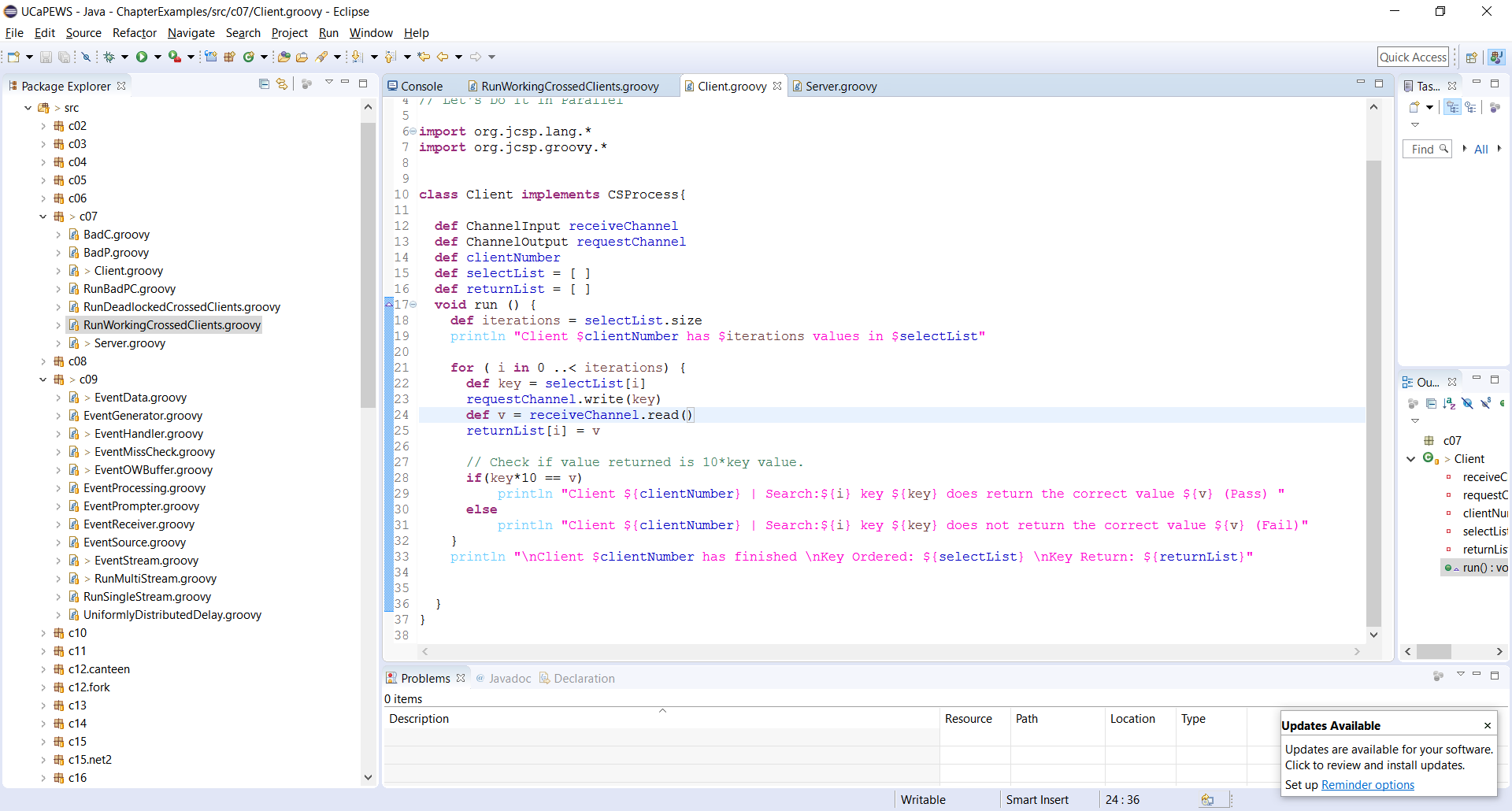




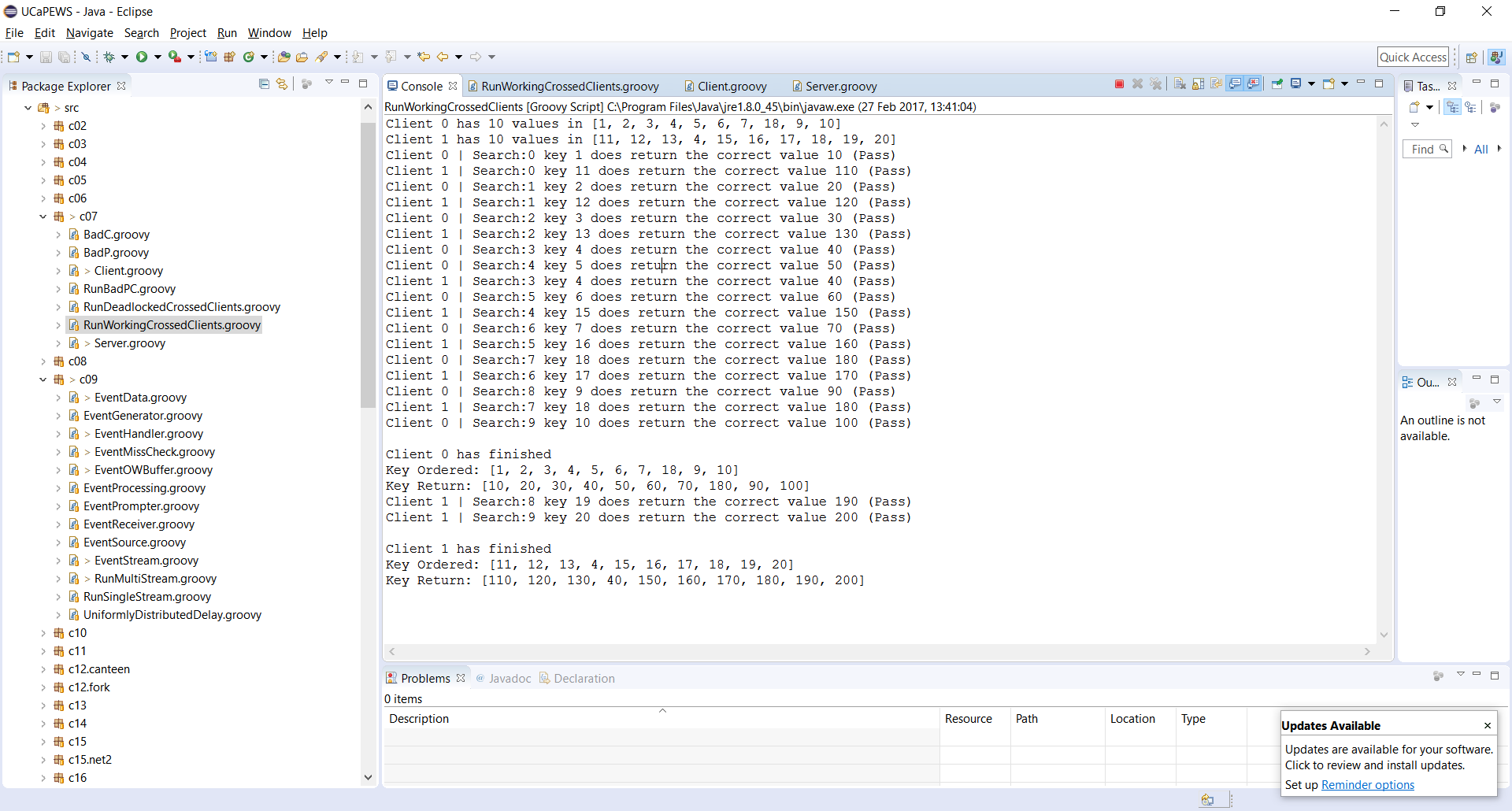
The deadlock in this system occurs when both servers try to access the other server. Due to this neither server can read or write which results in both servers being unable to finish their operations and thus ‘deadlocking’.

Exercise 8-1

In client.groovy.



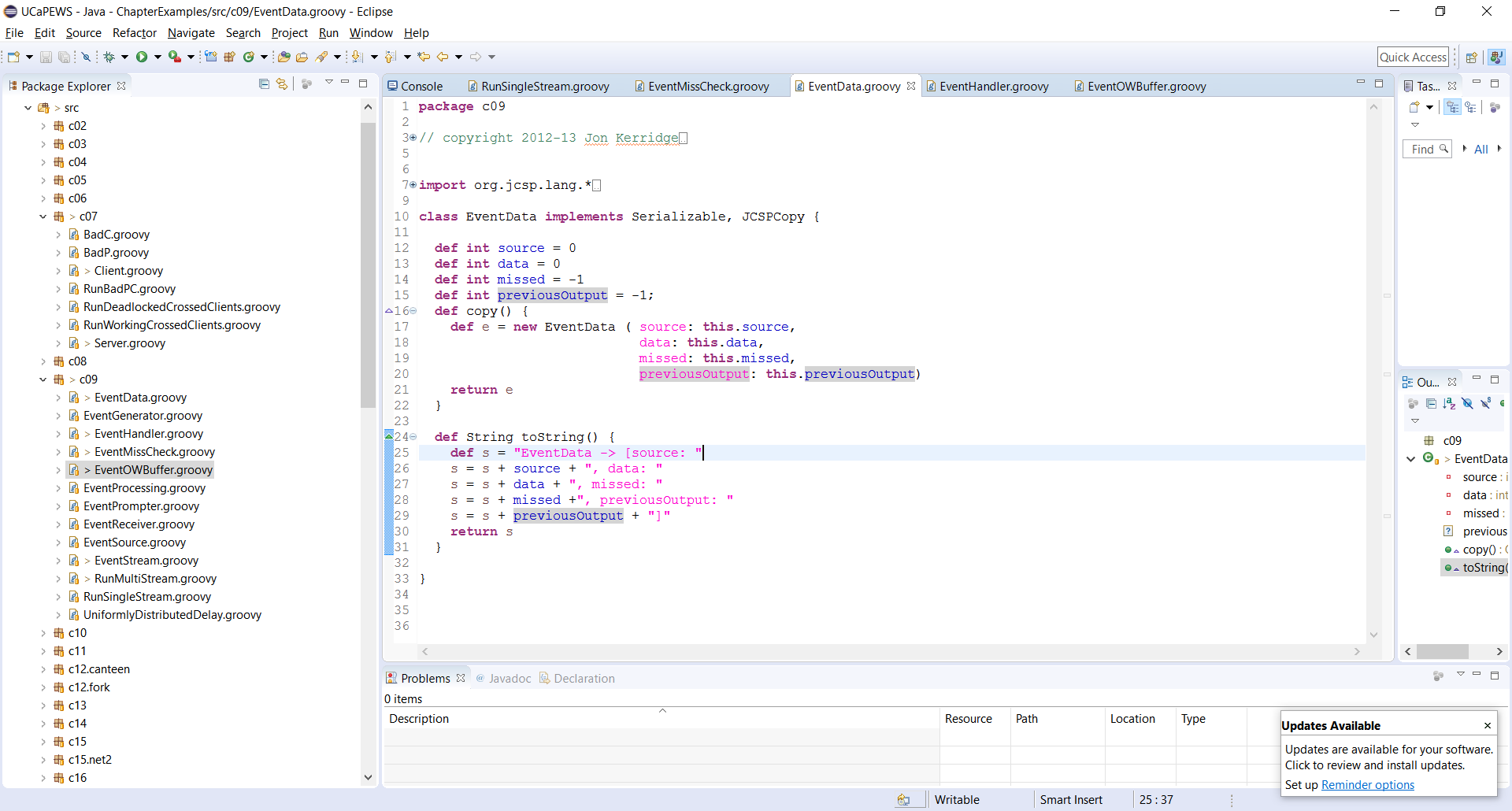
This snippet of code is to ensure that the value returned from the search returns the correct value. In this test data set. The returned value should be the key value \*10.



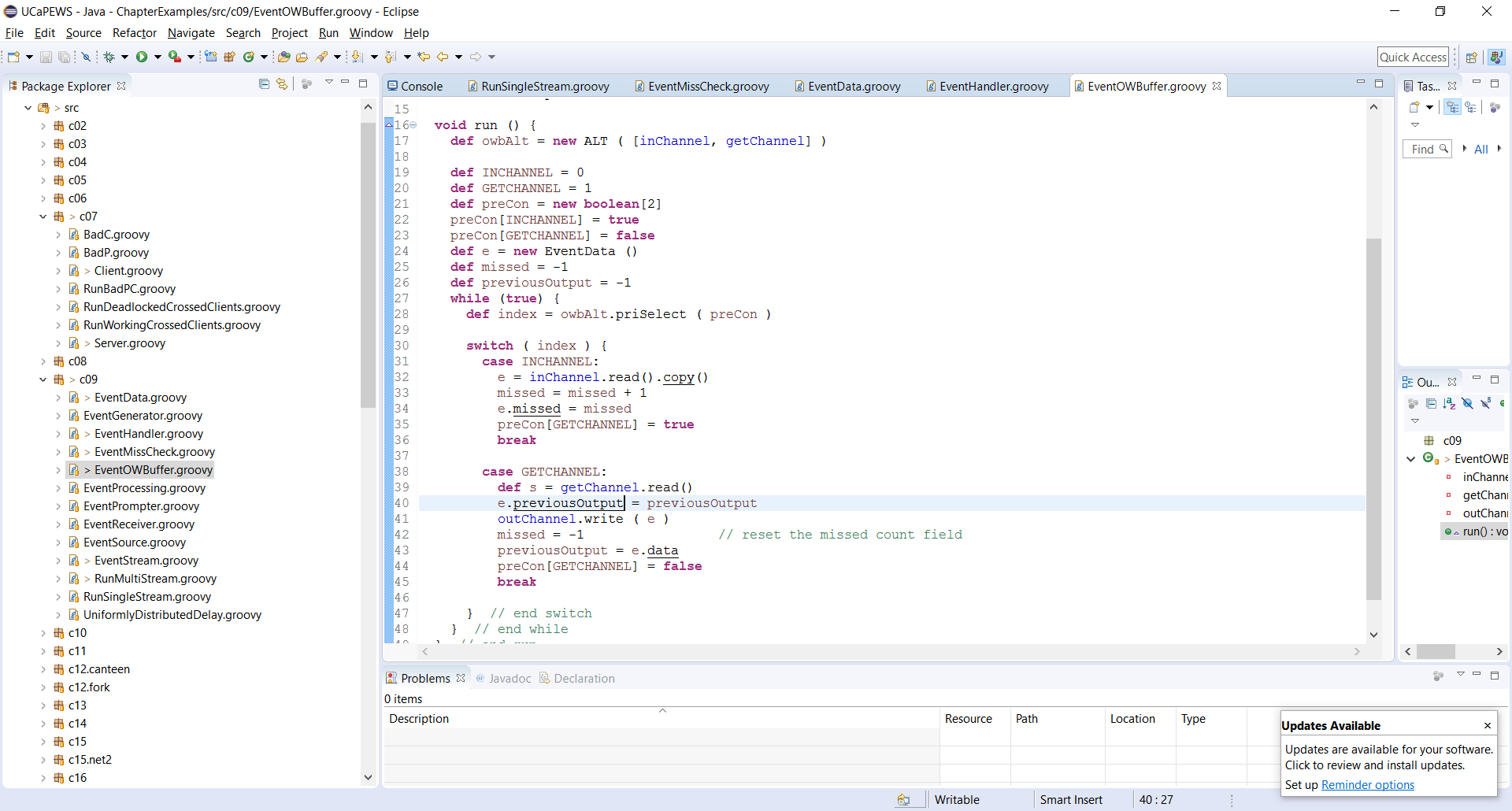
From the results shown it can be seen that the client searches according to the selectList property as well as returning the correct value.

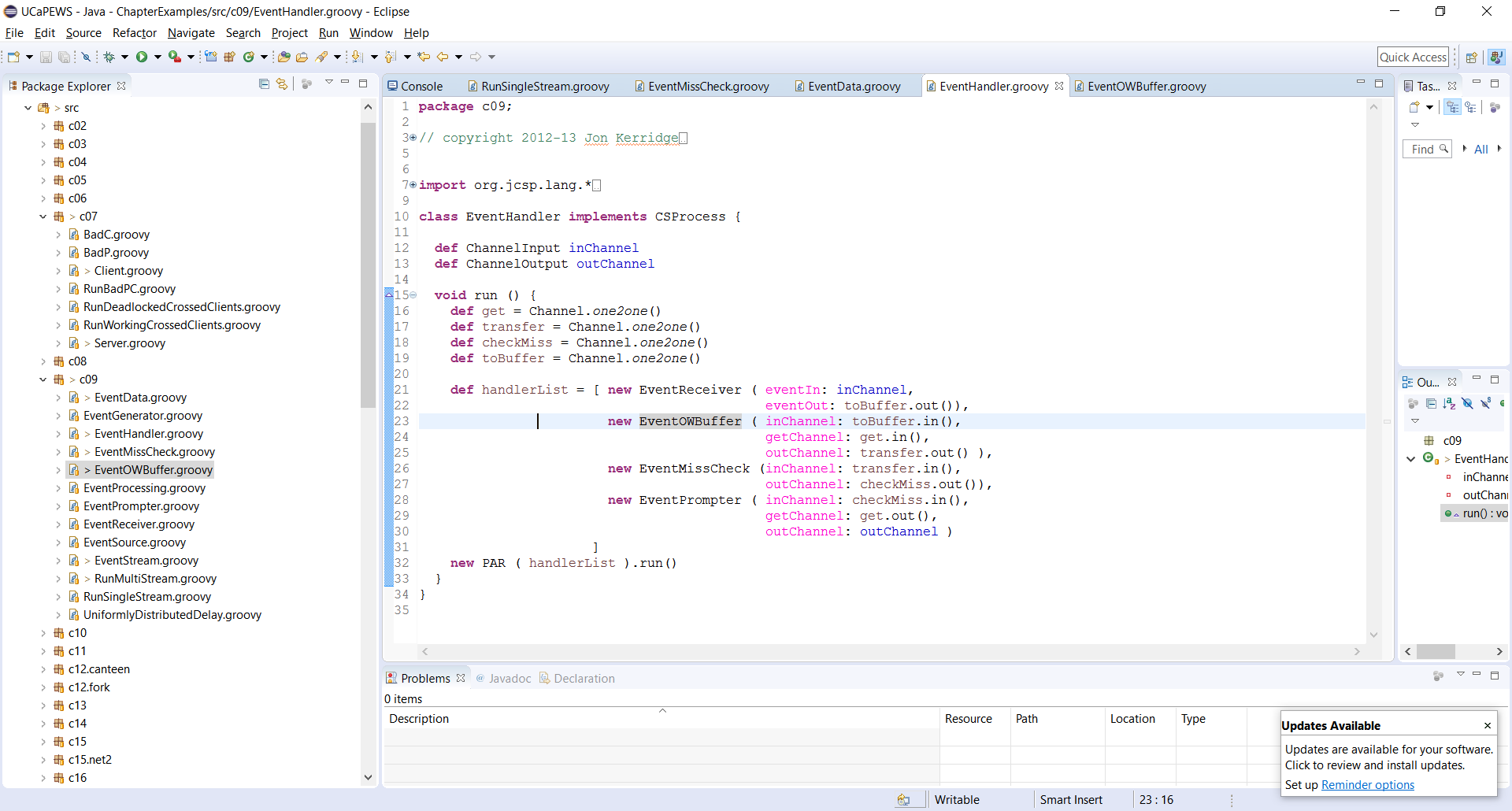
Exercise 9-1

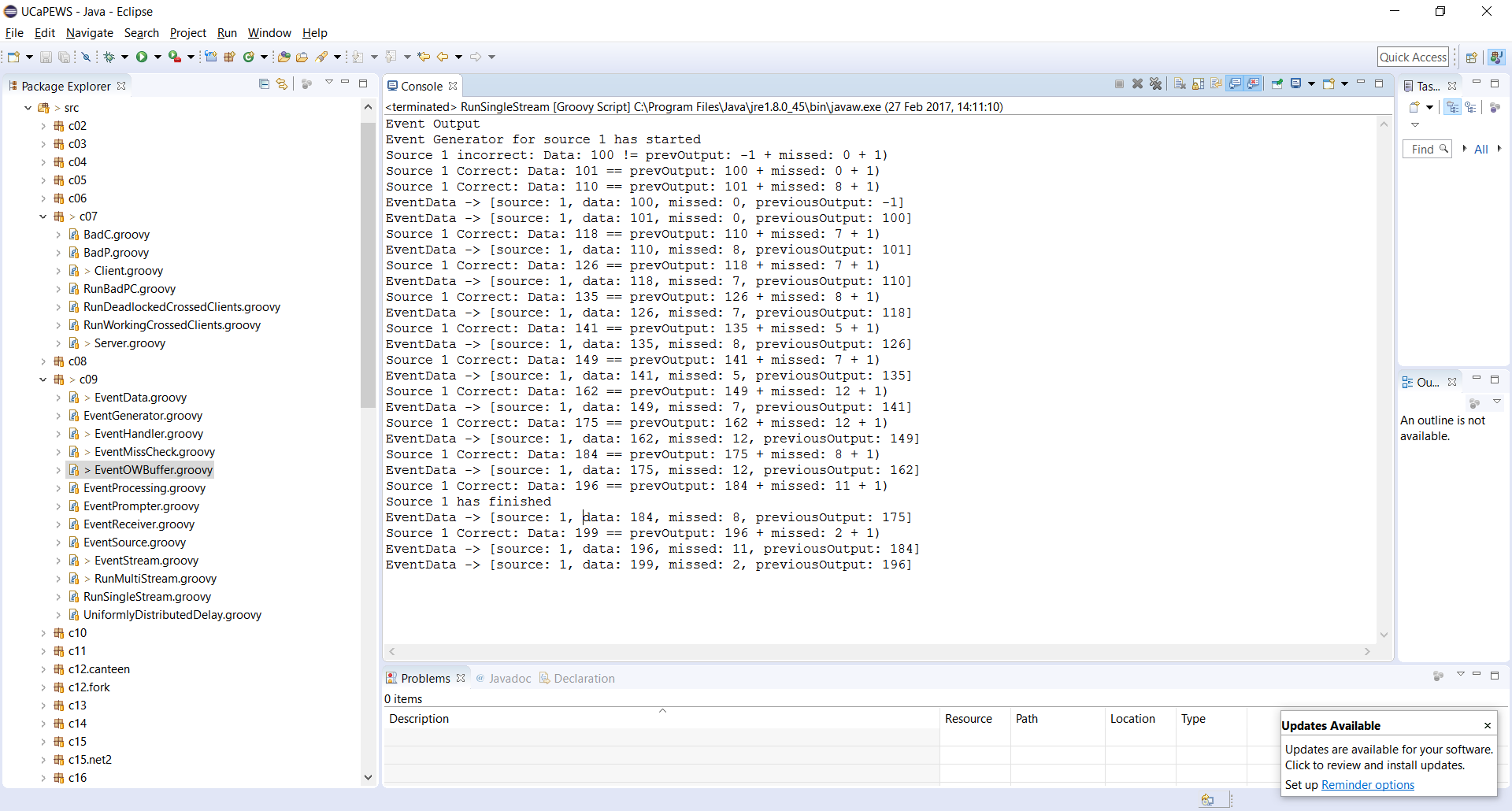
An issue to address is the output will show that the first value outputted will fail the event check misses. This is due to the fact that there was no previous output thus it cannot compare against anything.



EventOWBuffer.groovy







Exercise 9-2

Decreasing the minimum time means that more misses are possible as the time to compute might already be over.  
Increasing the maximum time allowed will result in lower misses as it will have more time in order to finish its computation.  
Increasing the minimum time means that the source will have more time on calculation but will take longer to process all the information.  
Decreasing the maximum time results in more misses as the system might now have enough time to process the data before timeout.

Evidence plz..

Exercise 9-3

FairMultiplexer allows a fair amount of time for each thing to do its thing. By default, the Multiplexer is a FairMultiplexer.

PriMultplexer gives higher priority depending on index.

Need evidence…