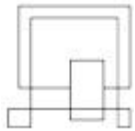
VIA University College



Software Development with UML and Java 2

Course Assignment 1

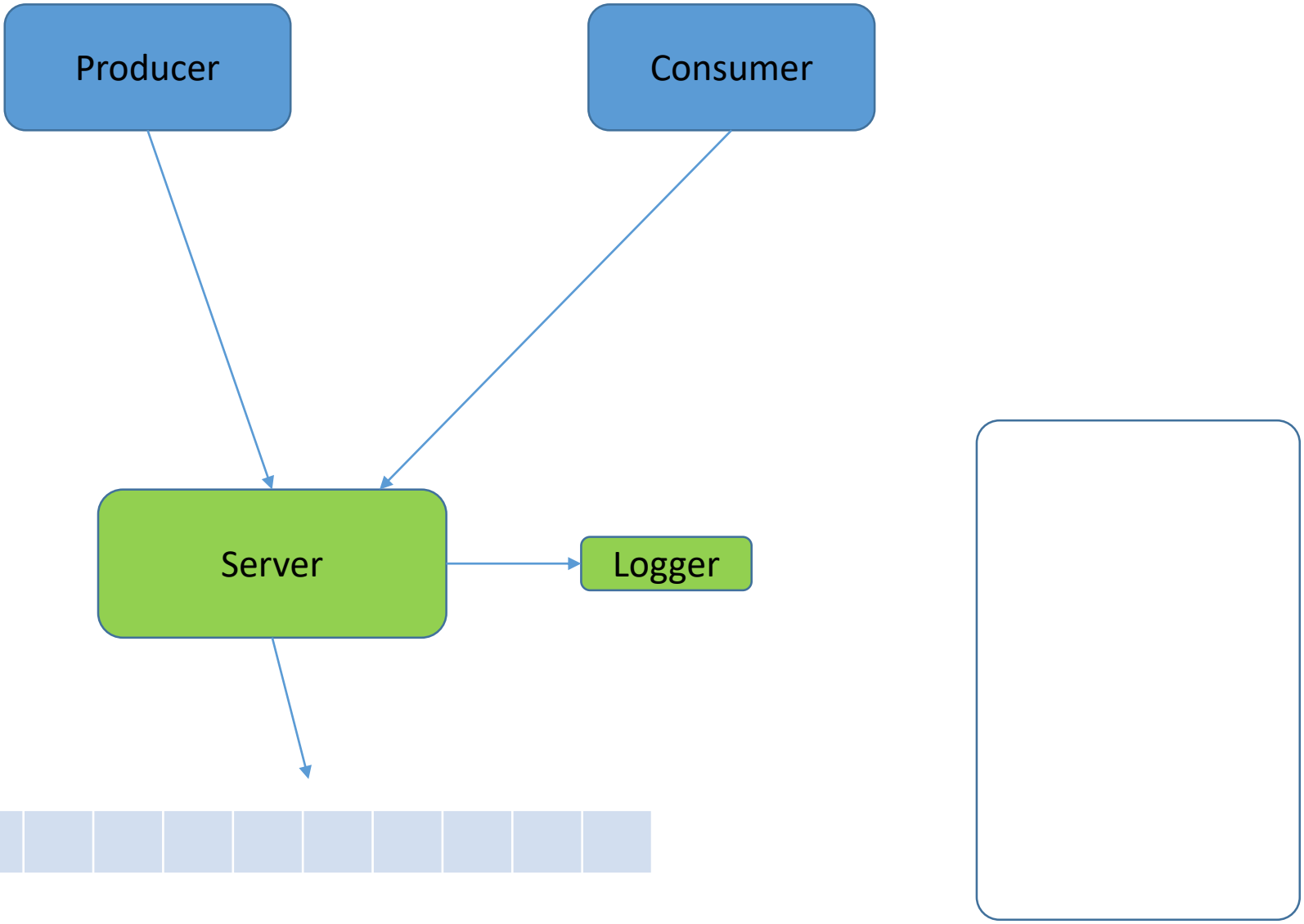
Topics covered

- Sockets
- Threads
- Producer/consumer
- Blocking queue
- Singleton
- Adapter

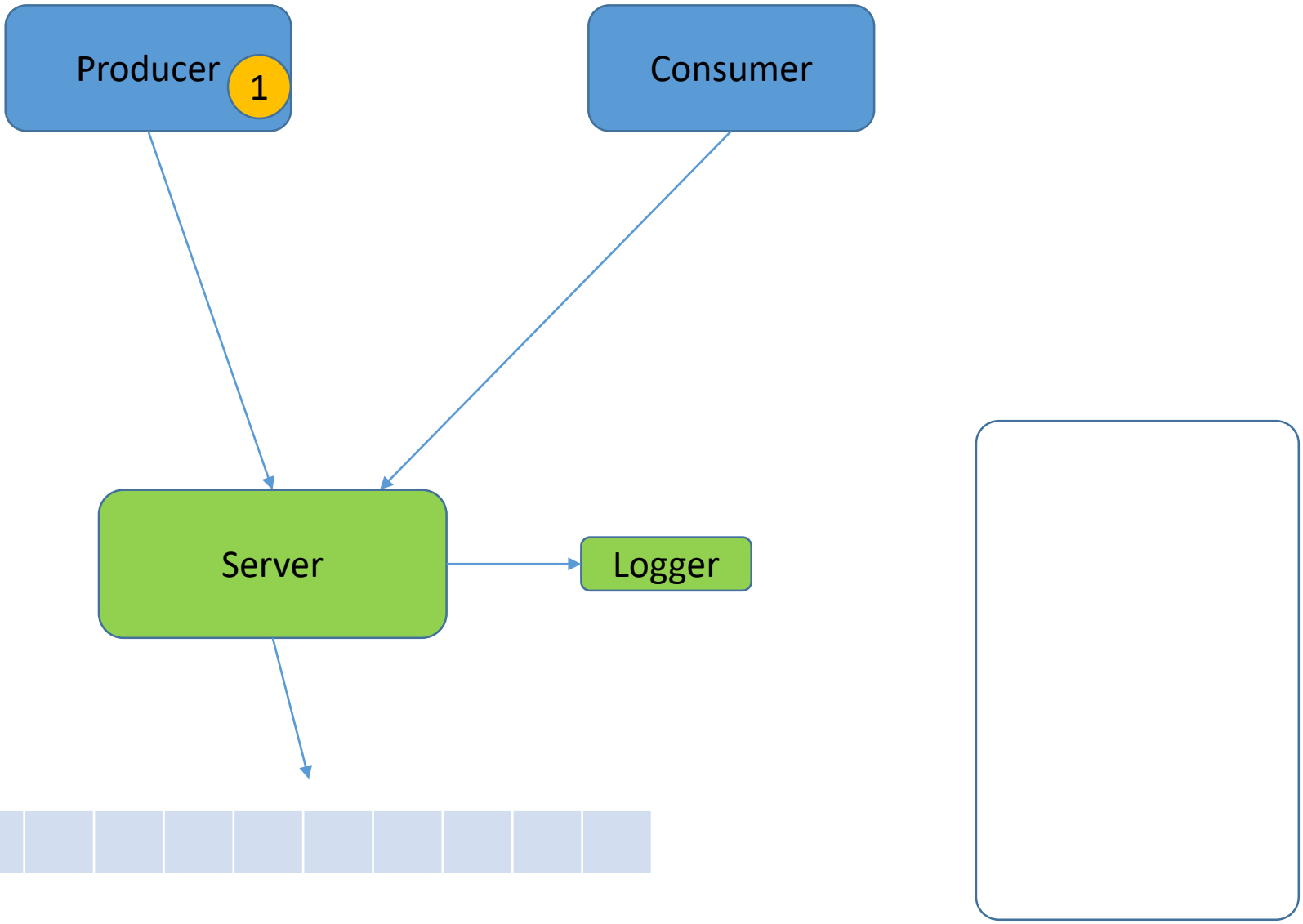
The idea

- Two types of client
  - One client sends numbers to the server
  - Another retrieves some numbers, do a mathematical operation, sends the result back to the server
- The server
  - Handles multiple clients
  - Contains the list of numbers
  - Prints out what is going on

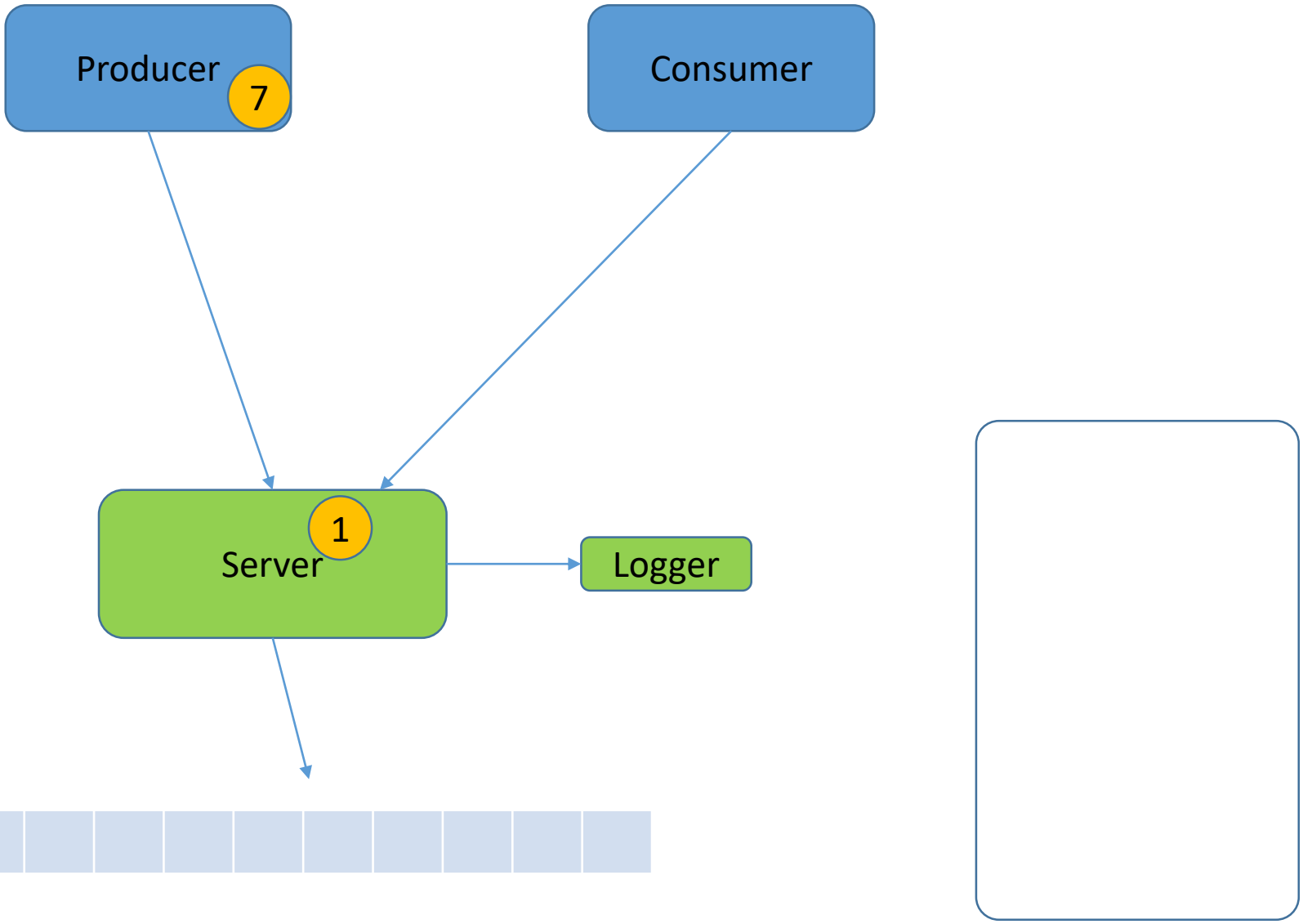
The Idea



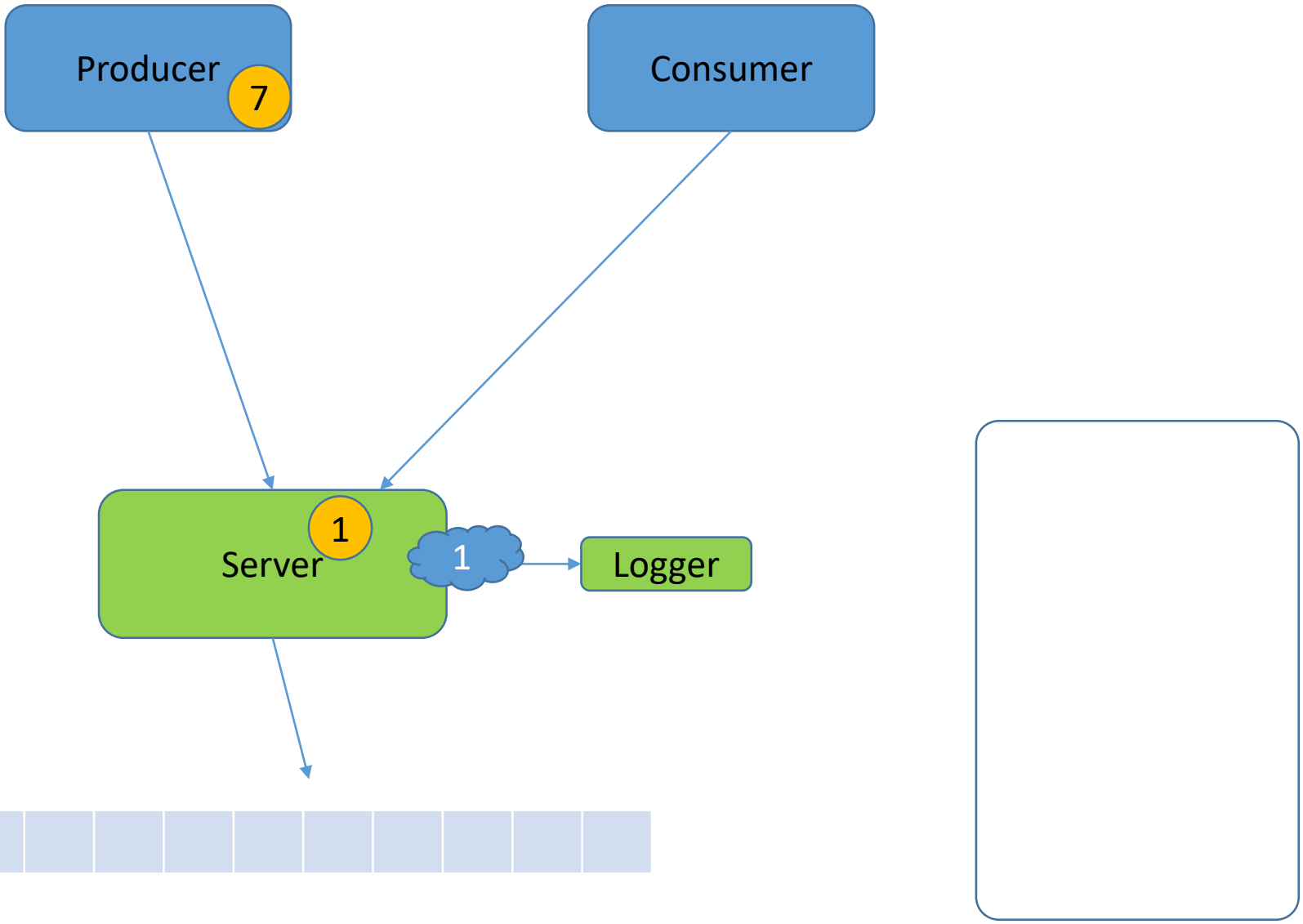
The Idea



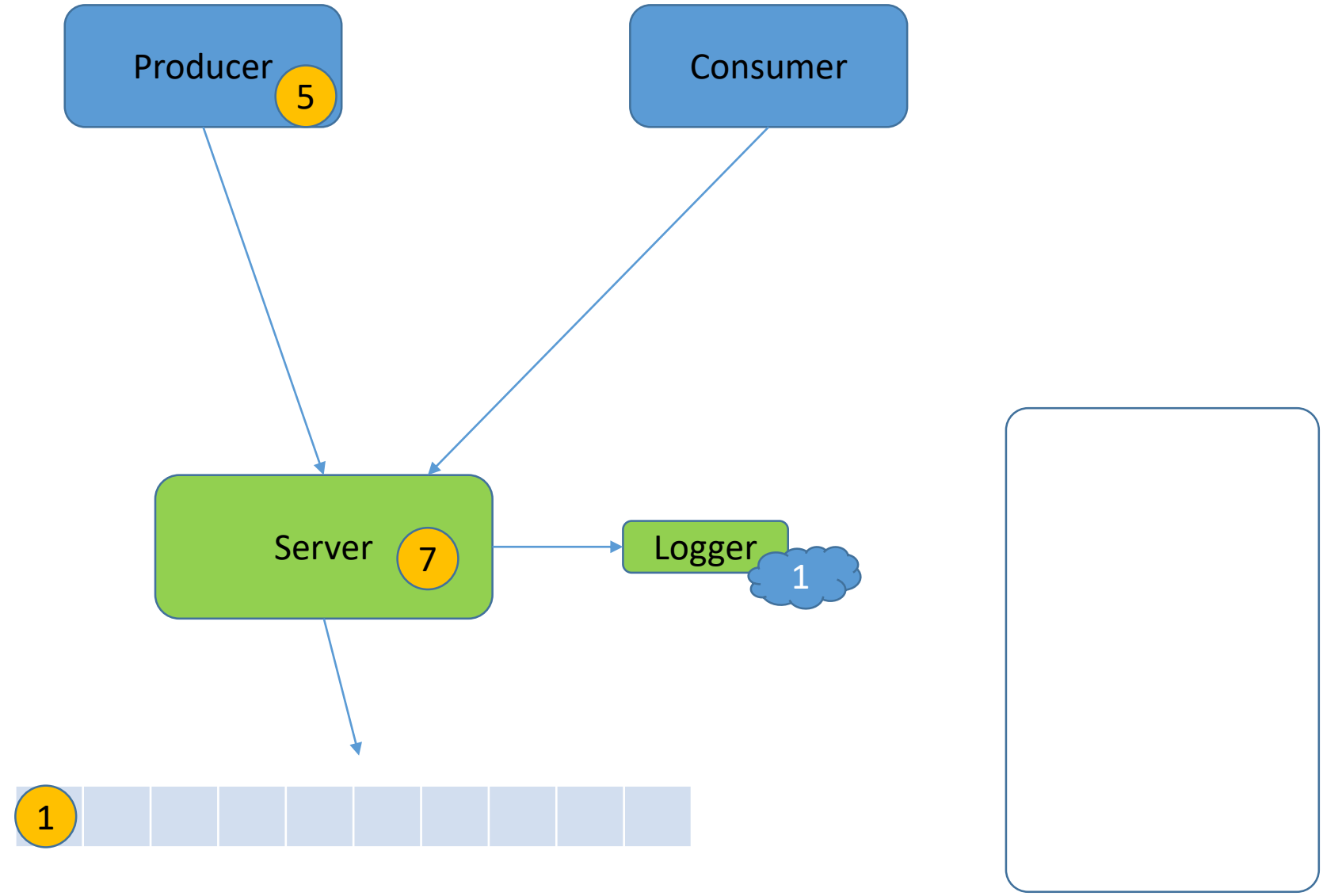
The Idea



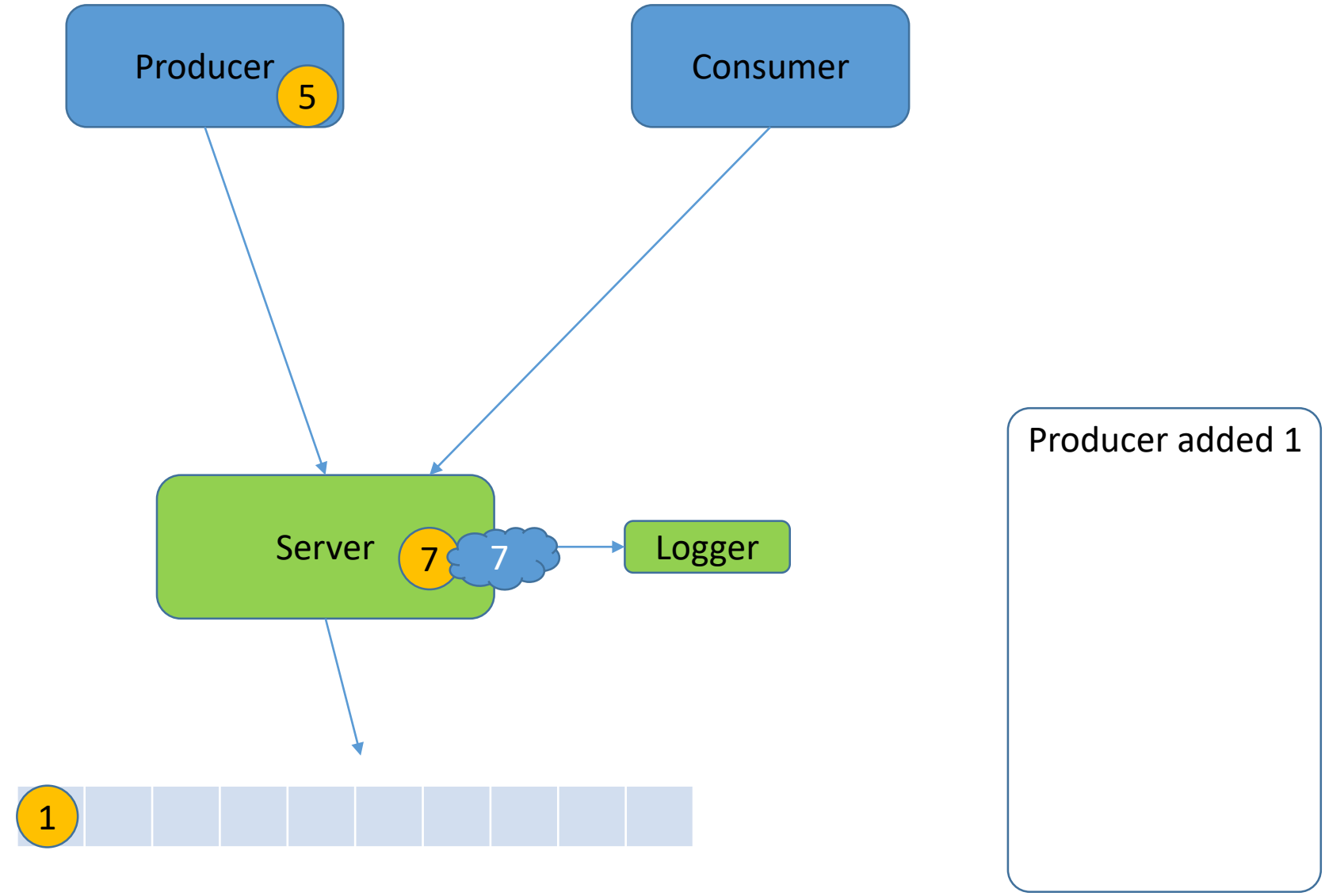
The Idea



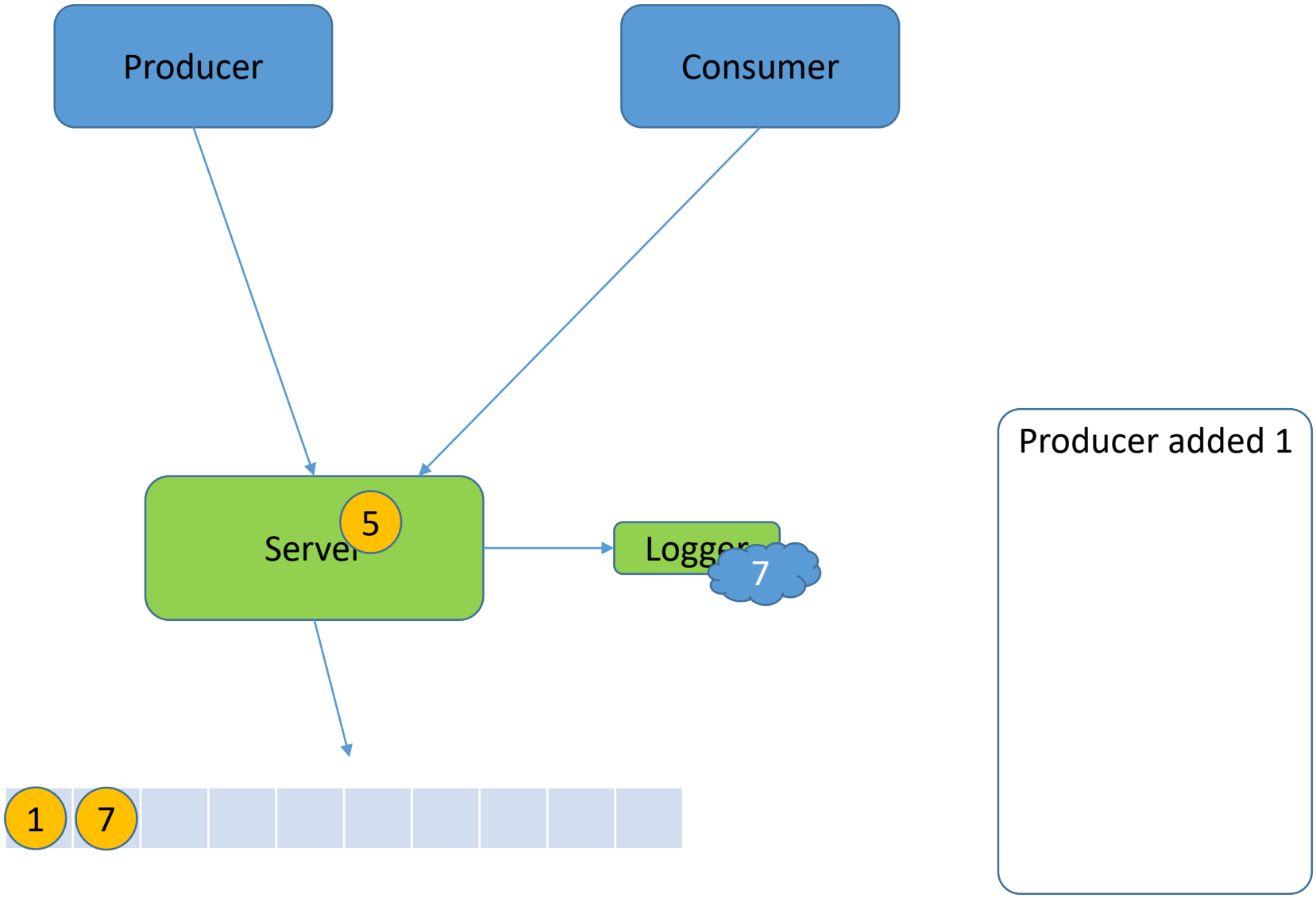
The Idea



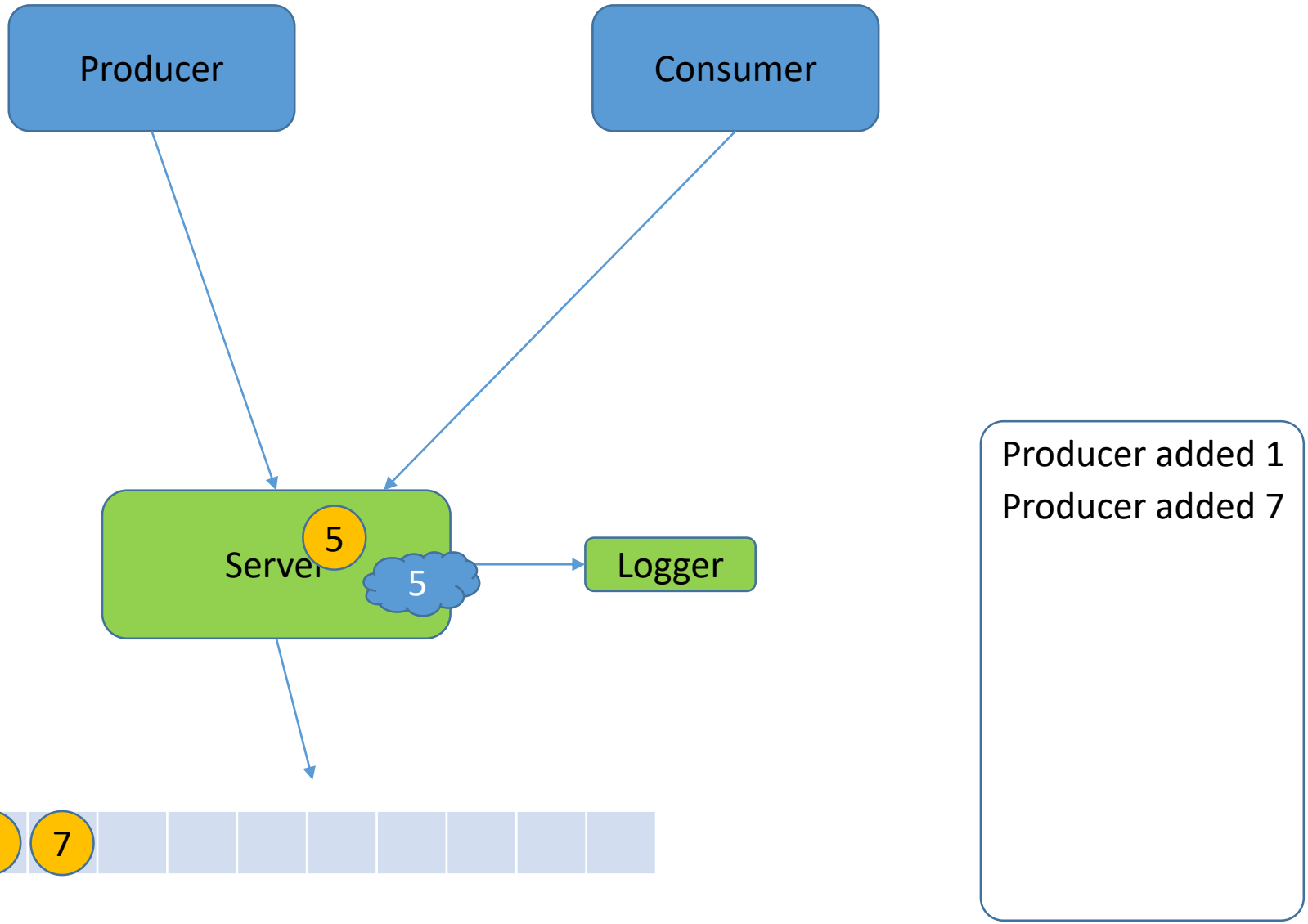
The Idea



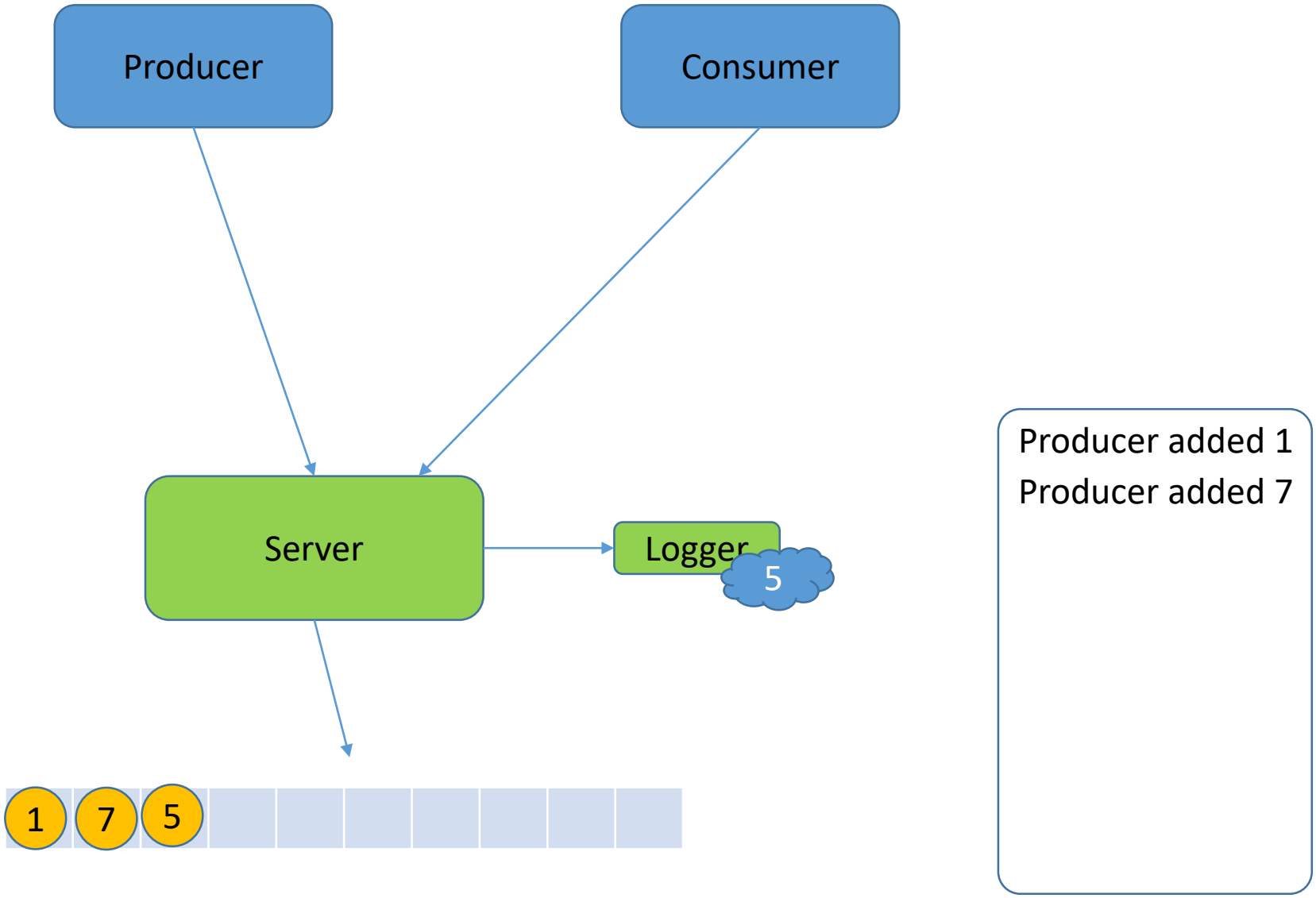
The Idea



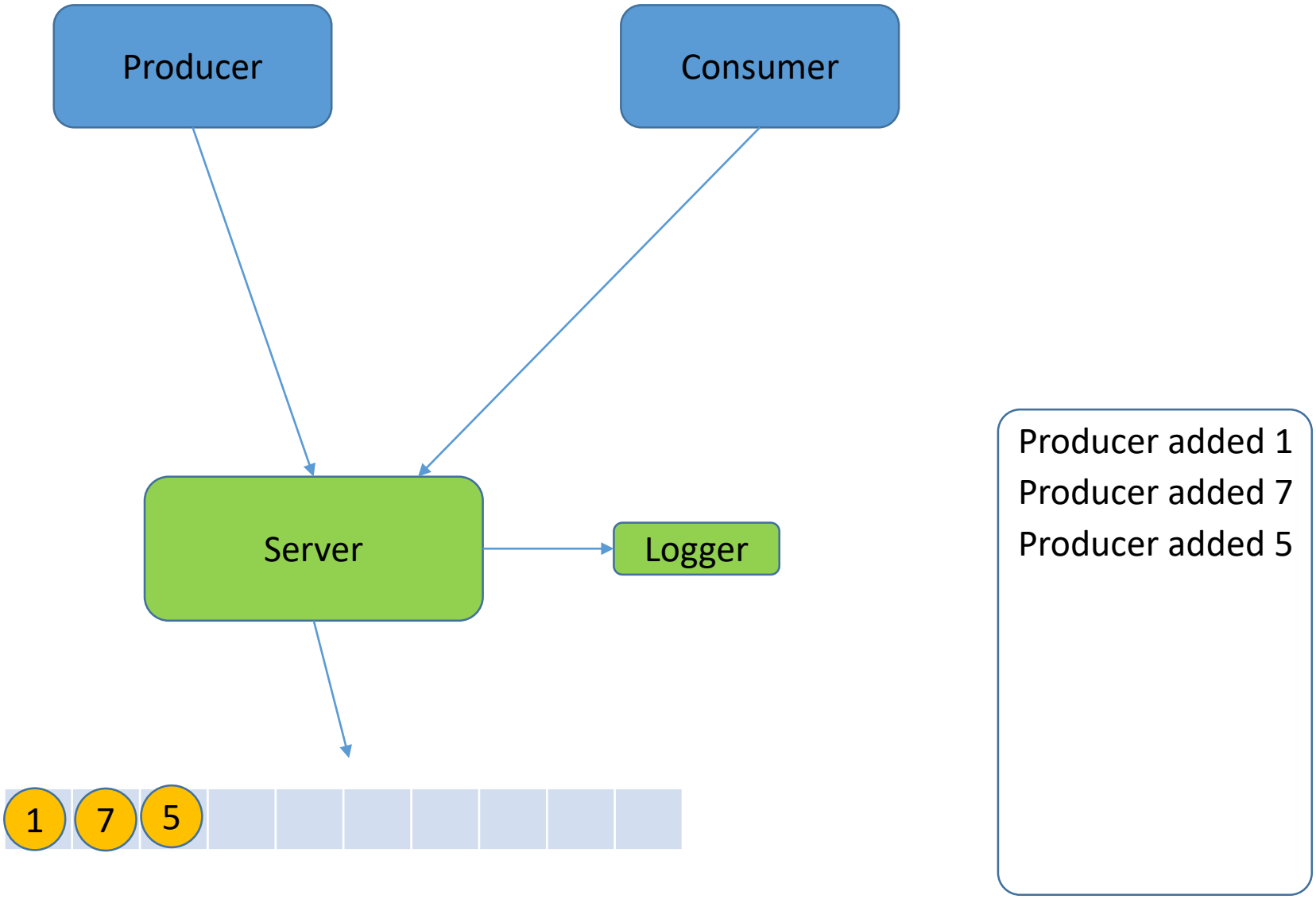
The Idea


Producer
Consumer
5
Server
5
Logger
1
7
Producer added 1
Producer added 7

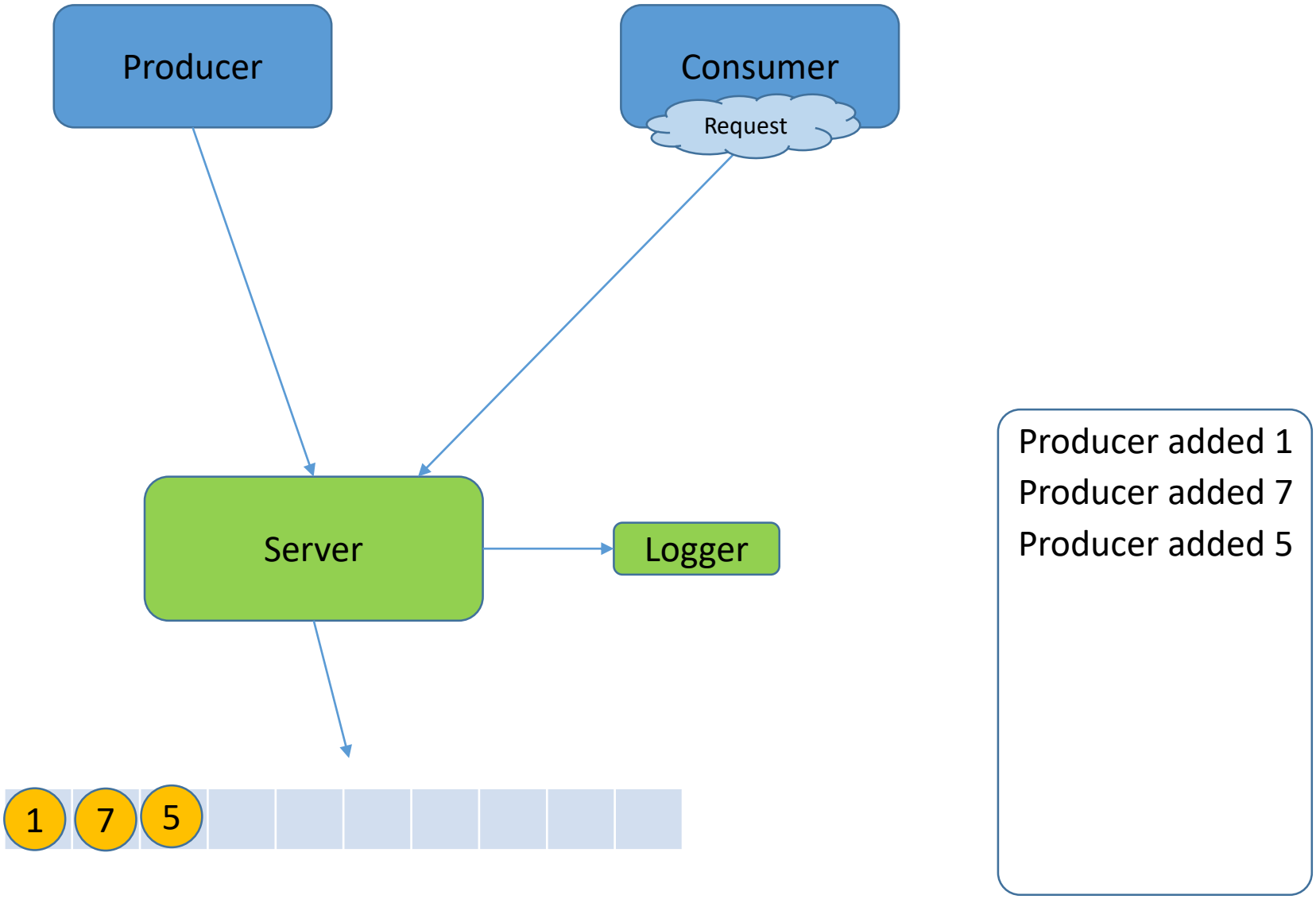
The Idea


Producer
Consumer
Server
Logger
5
1
7
5
Producer added 1
Producer added 7

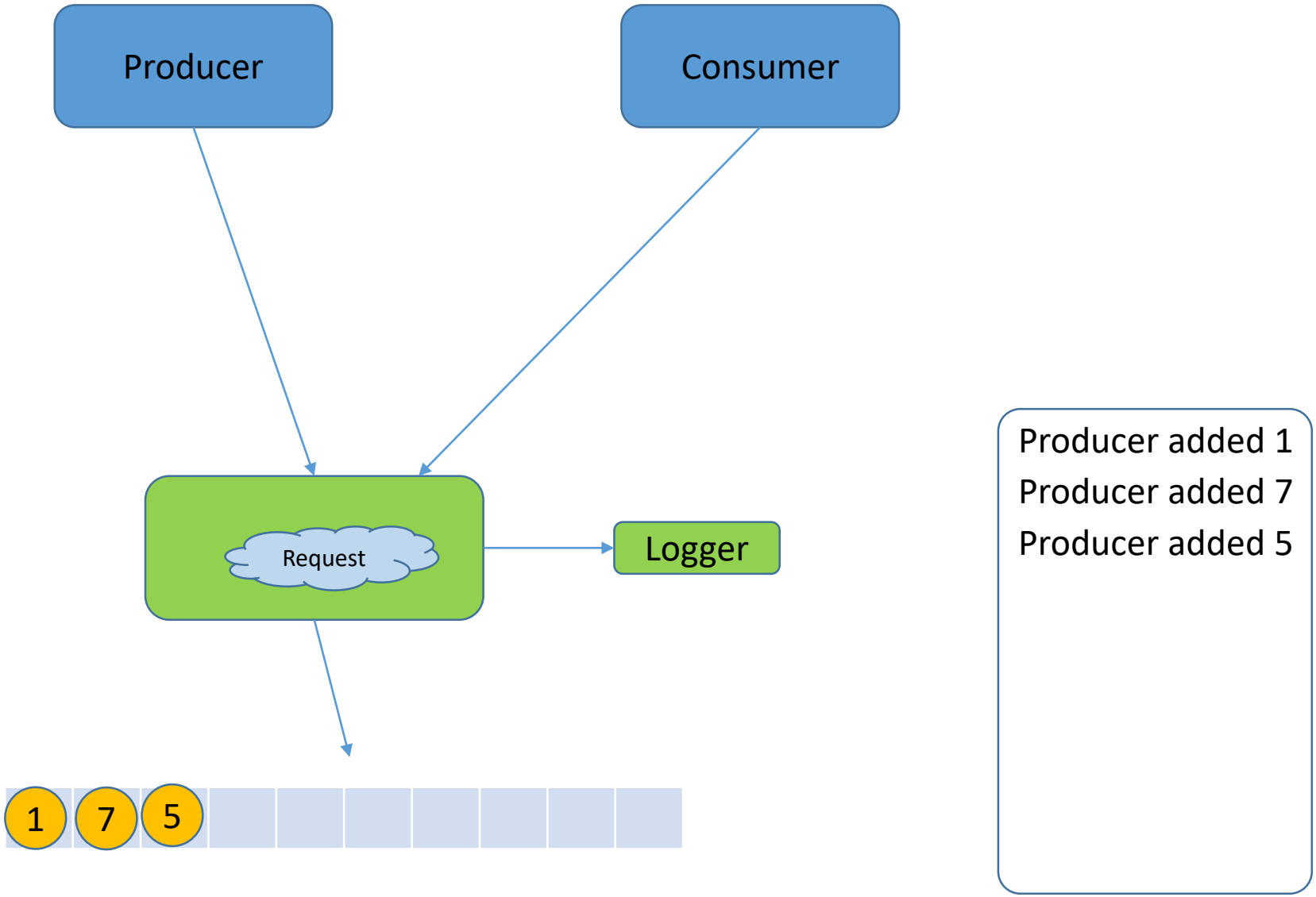
The Idea


Producer
Consumer
Server
Logger
1
7
5
Producer added 1
Producer added 7
Producer added 5

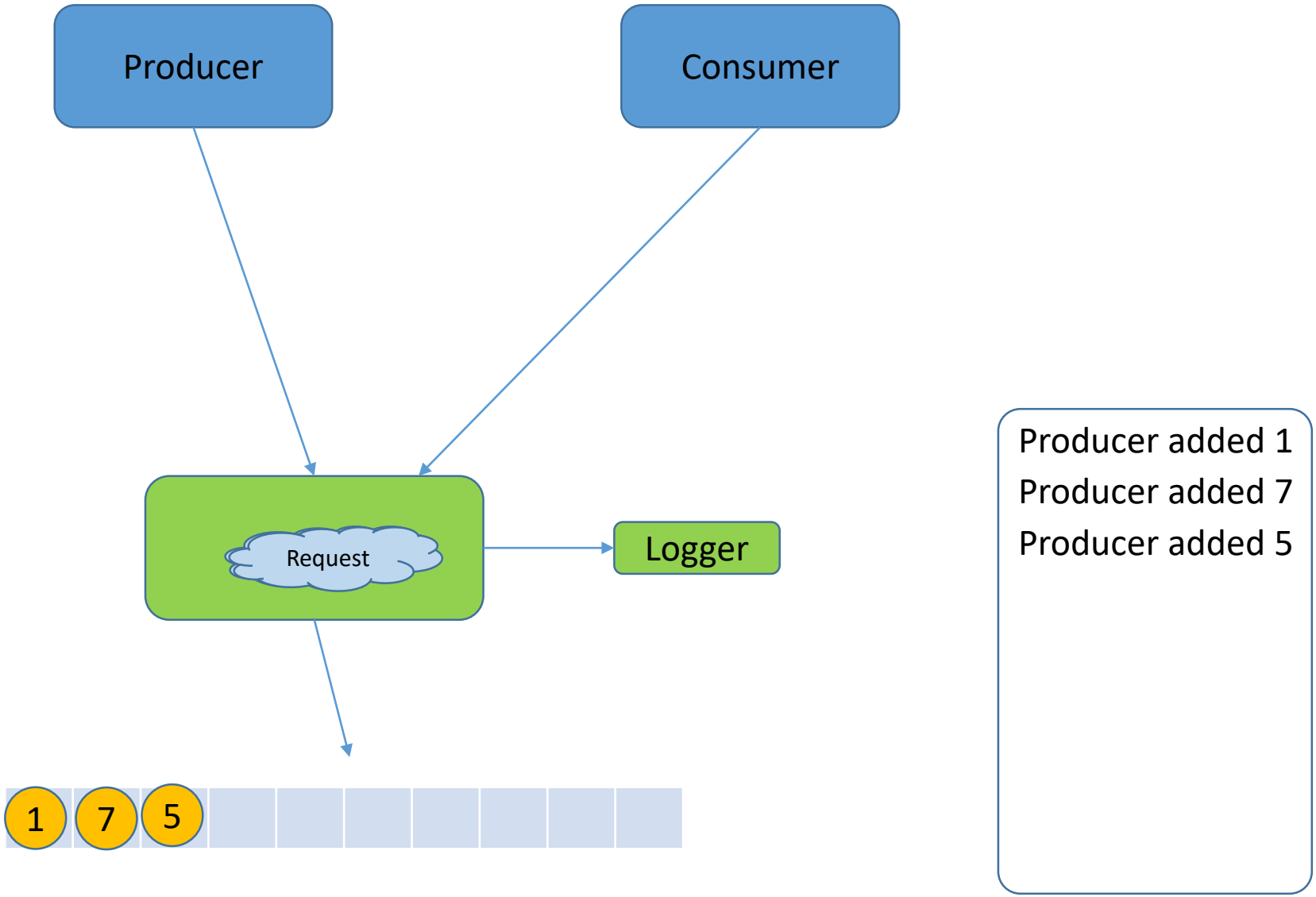
The Idea


Producer
Consumer
Request
Server
Logger
1
7
5
Producer added 1
Producer added 7
Producer added 5

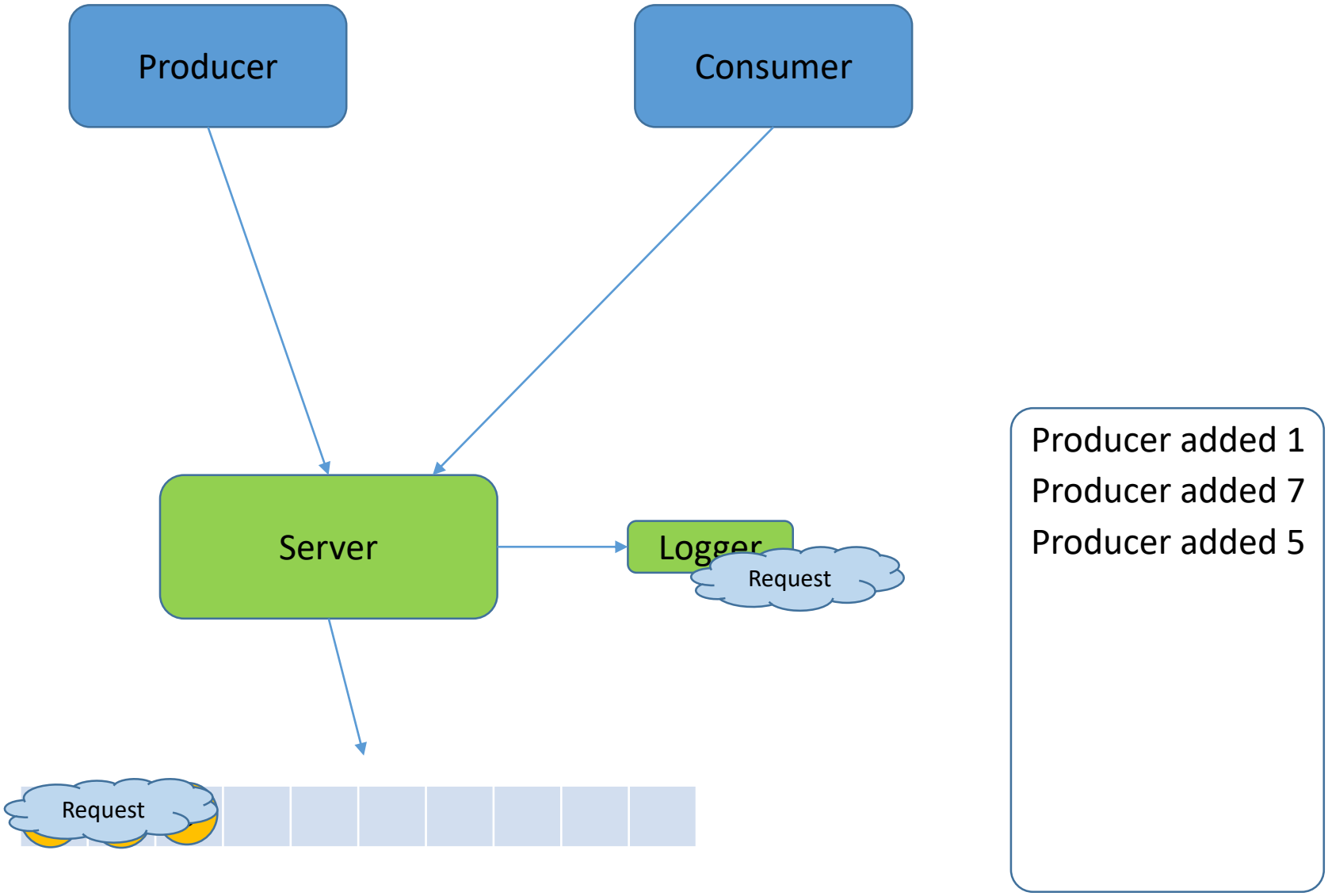
The Idea


Producer
Consumer
Request
Logger
1
7
5
Producer added 1
Producer added 7
Producer added 5

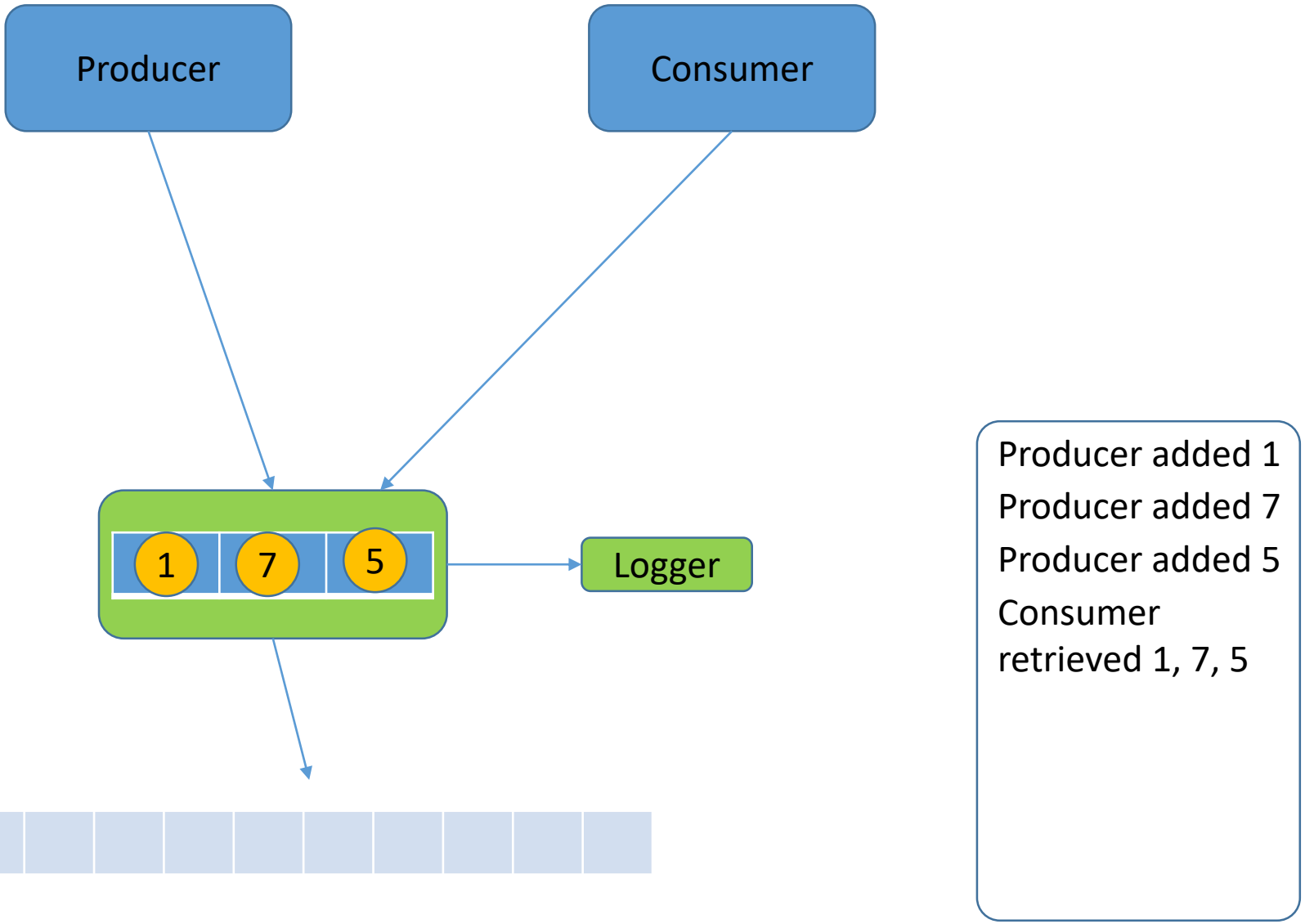
The Idea


Producer
Consumer
Request
Logger
1
7
5
Producer added 1
Producer added 7
Producer added 5

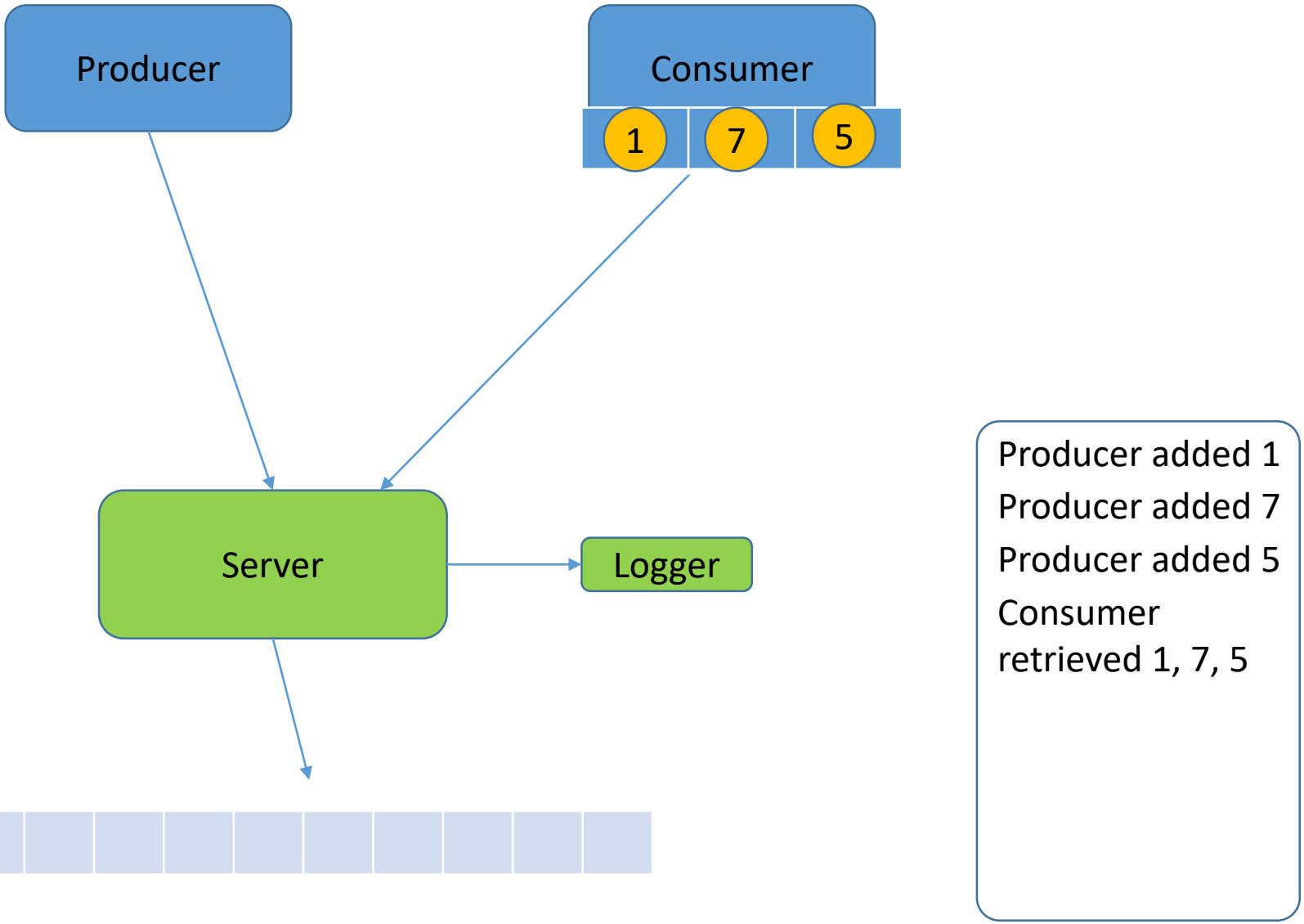
The Idea


Producer
Consumer
Server
Logger
Request
Request
Producer added 1
Producer added 7
Producer added 5

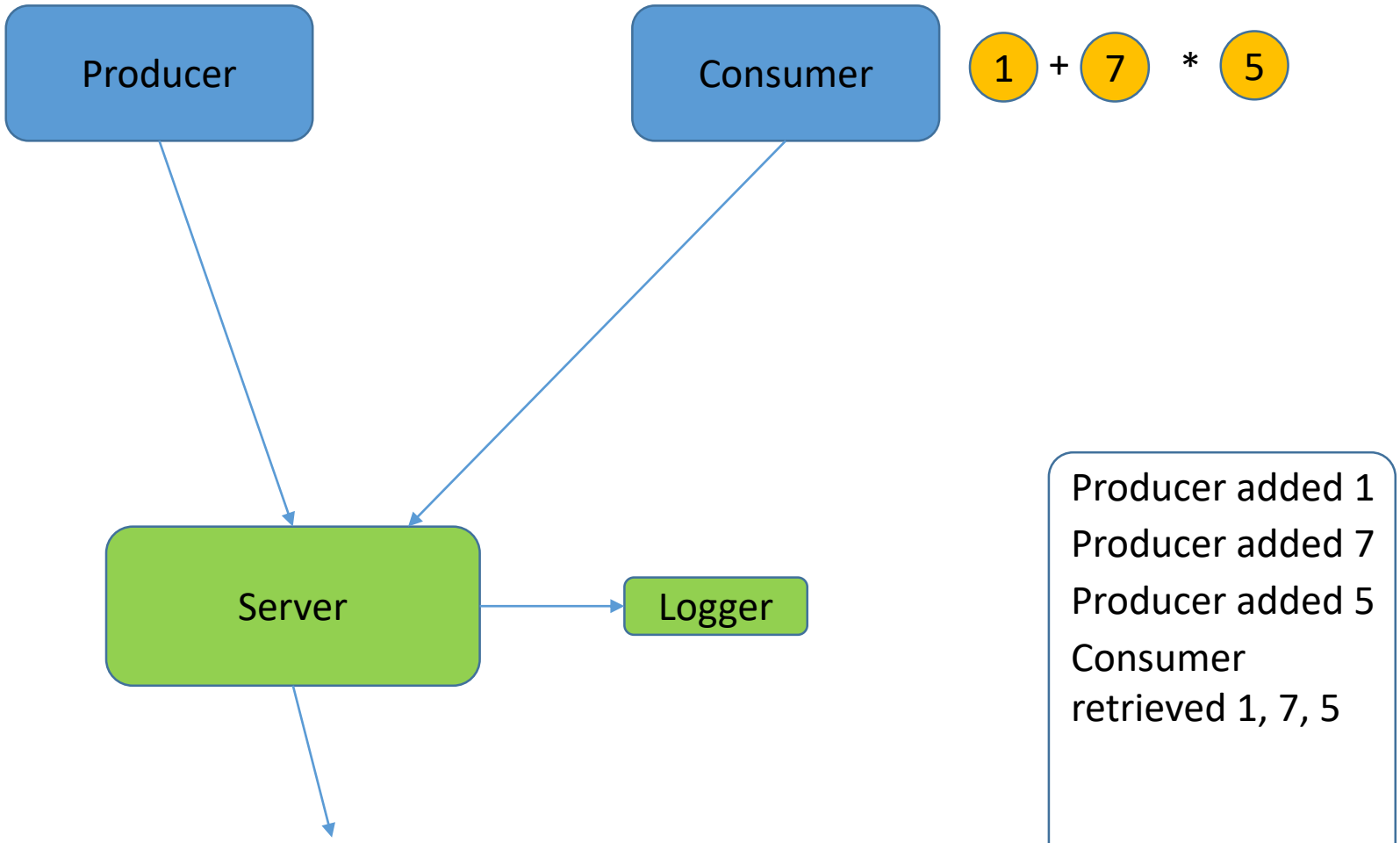
The Idea


Producer
Consumer
1
7
5
Logger
Producer added 1
Producer added 7
Producer added 5
Consumer retrieved 1, 7, 5

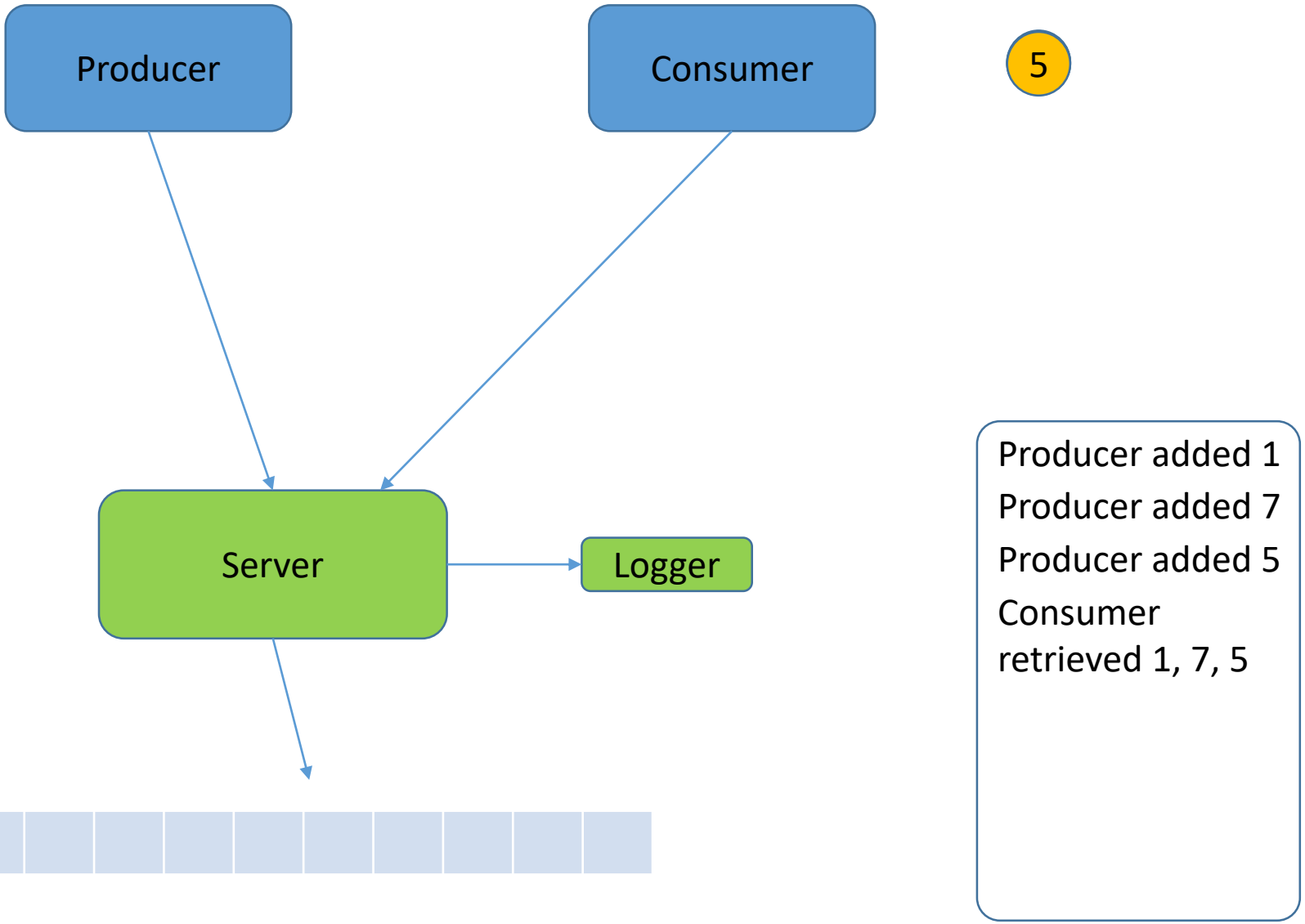
The Idea


Producer
Consumer
1
7
5
Server
Logger
Producer added 1
Producer added 7
Producer added 5
Consumer retrieved 1, 7, 5

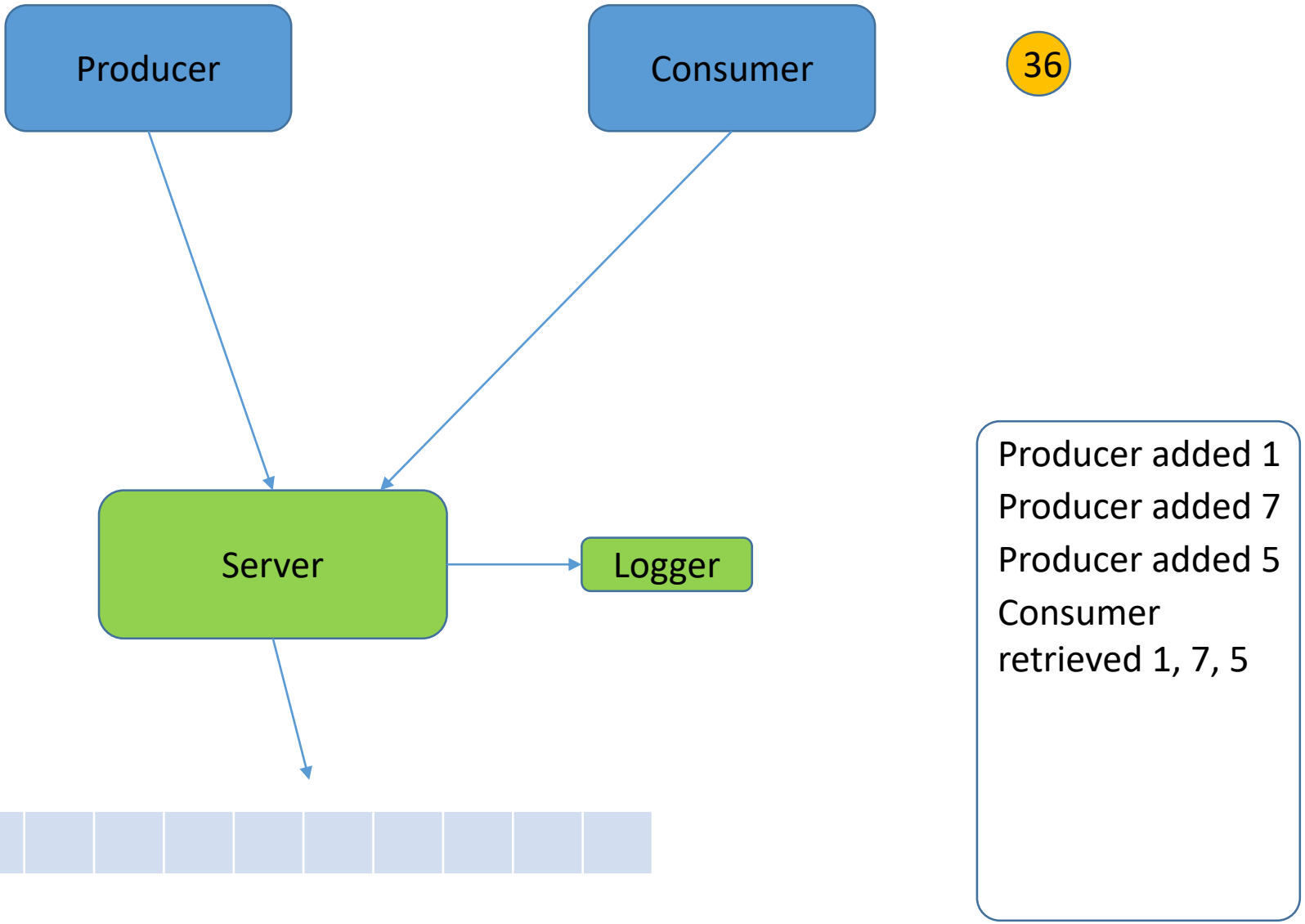
The Idea


Producer
Consumer
1
+
7
*
5
Server
Logger
Producer added 1
Producer added 7
Producer added 5
Consumer retrieved 1, 7, 5

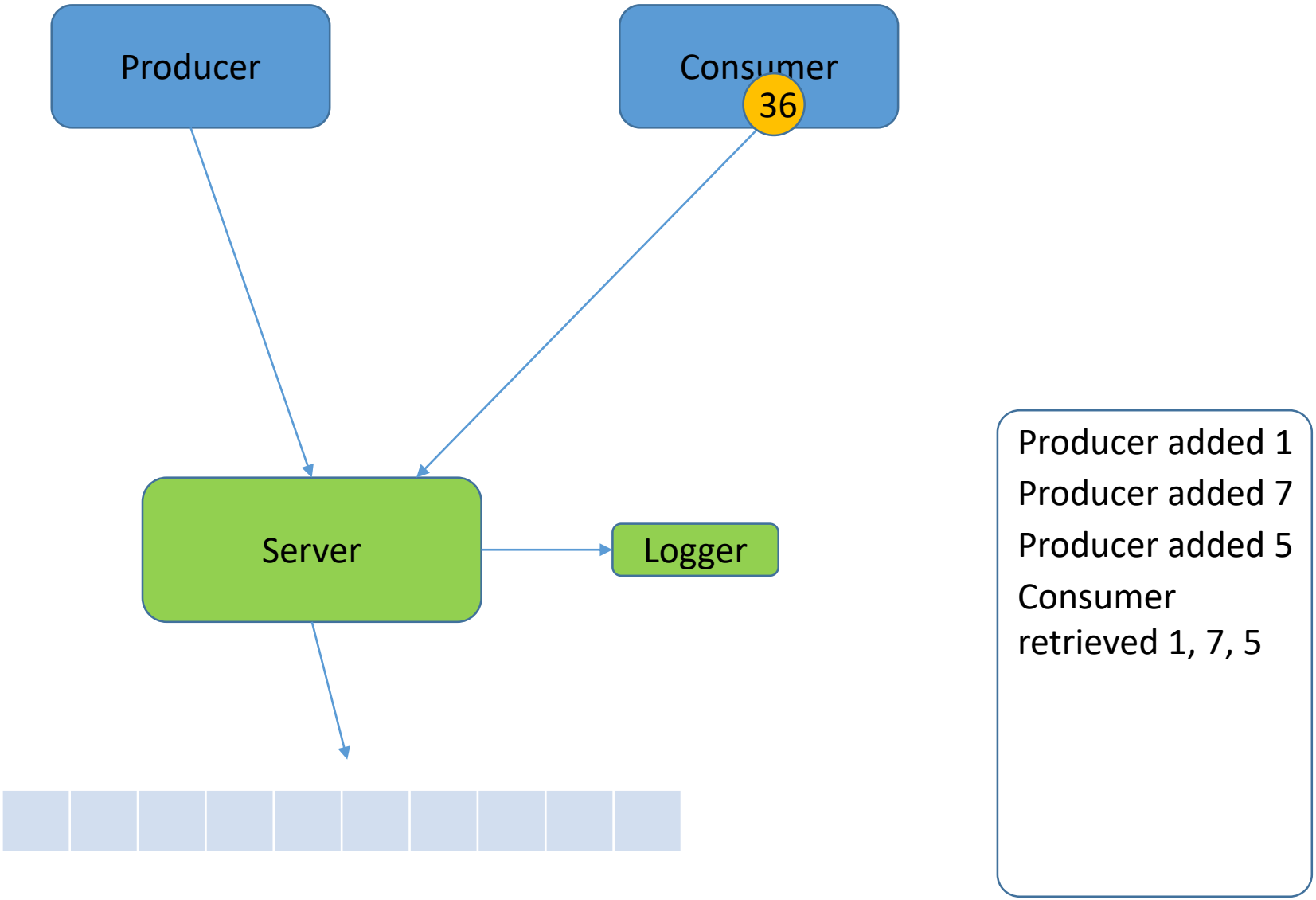
The Idea


Producer
Consumer
5
Server
Logger
Producer added 1
Producer added 7
Producer added 5
Consumer retrieved 1, 7, 5

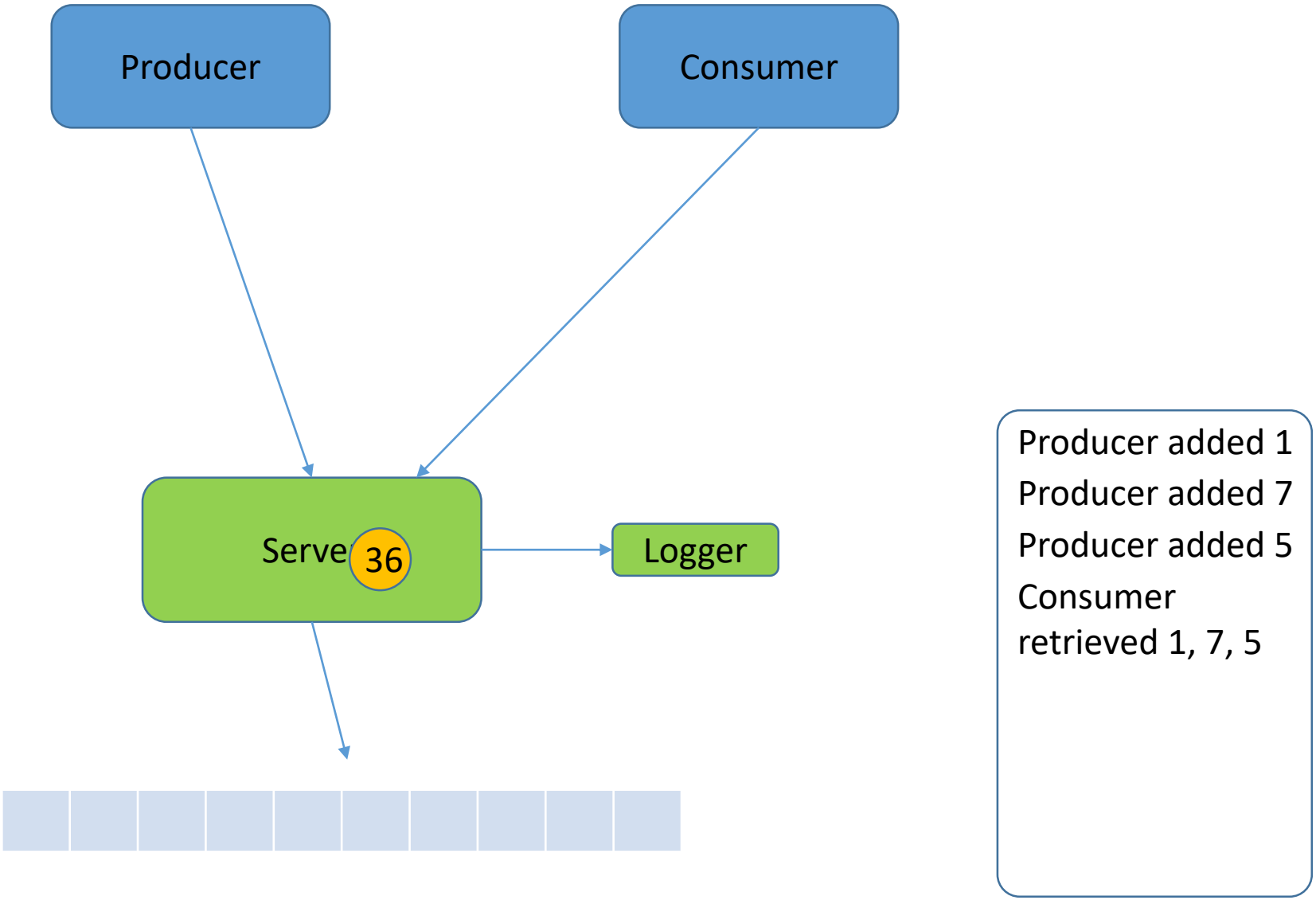
The Idea


Producer
Consumer
36
Server
Logger
Producer added 1
Producer added 7
Producer added 5
Consumer retrieved 1, 7, 5

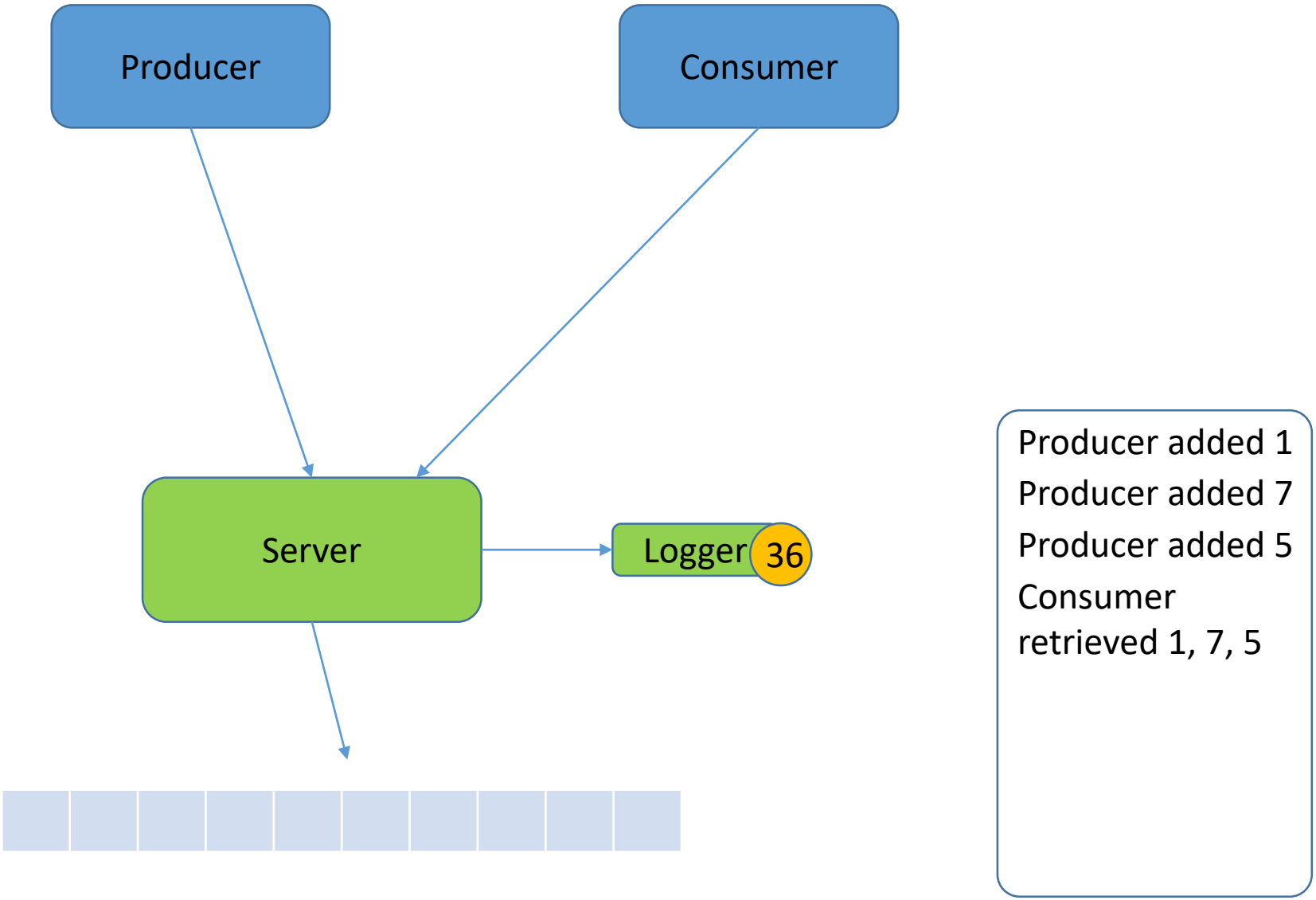
The Idea


Producer
Consumer
36
Server
Logger
Producer added 1
Producer added 7
Producer added 5
Consumer retrieved 1, 7, 5

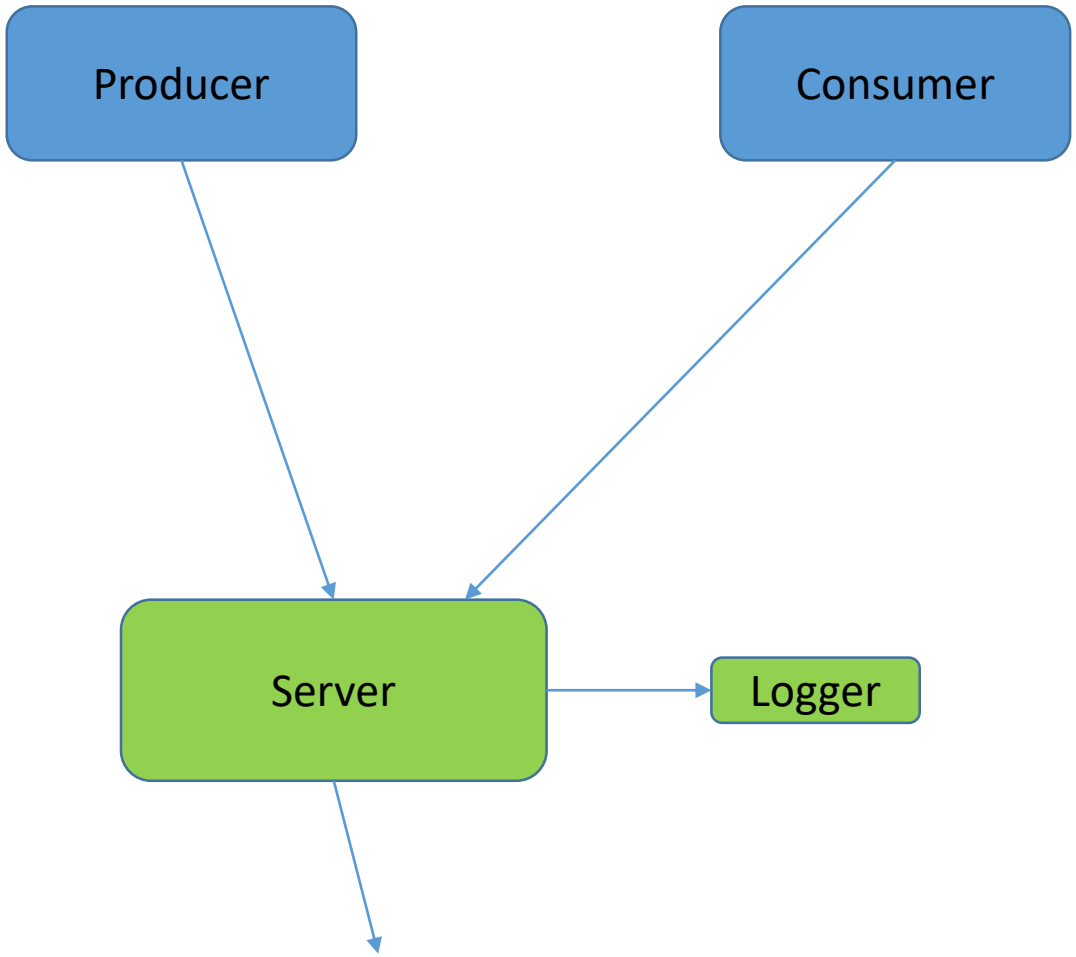
The Idea


Producer
Consumer
Server
36
Logger
Producer added 1
Producer added 7
Producer added 5
Consumer retrieved 1, 7, 5

The Idea

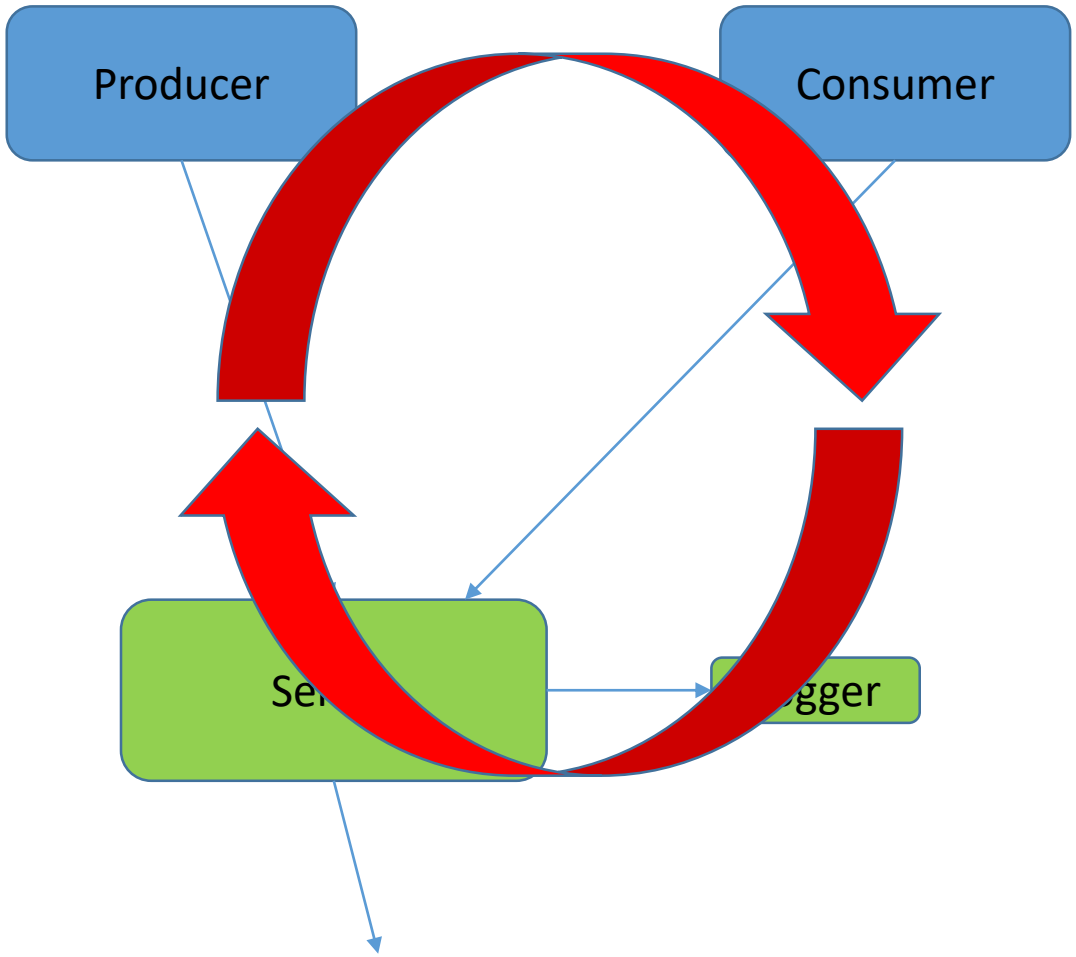

Producer
Consumer
Server
Logger
36
Producer added 1
Producer added 7
Producer added 5
Consumer retrieved 1, 7, 5

The Idea

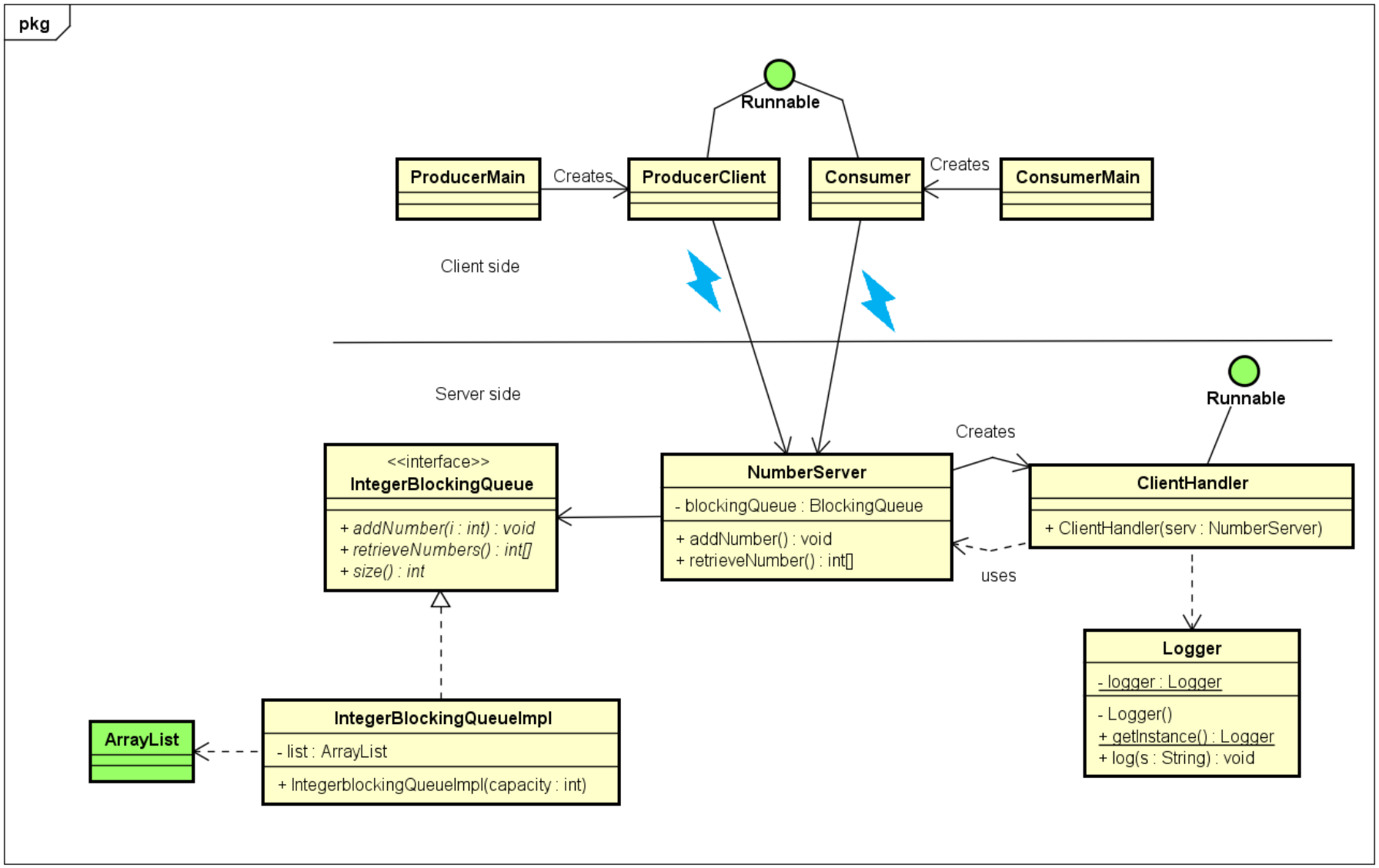


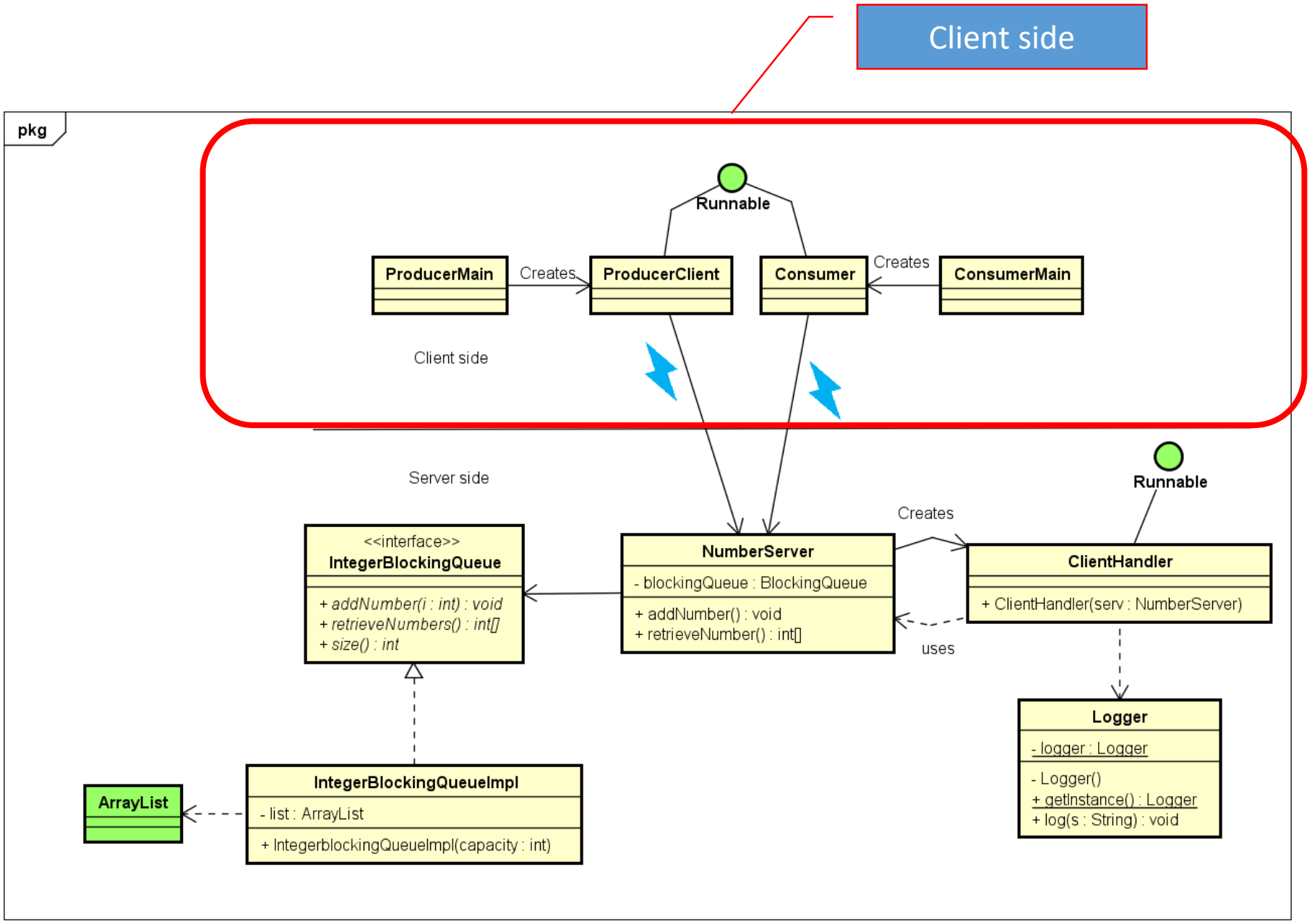
Producer added 1
Producer added 7
Producer added 5
Consumer retrieved 1, 7, 5
Consumer result: 36

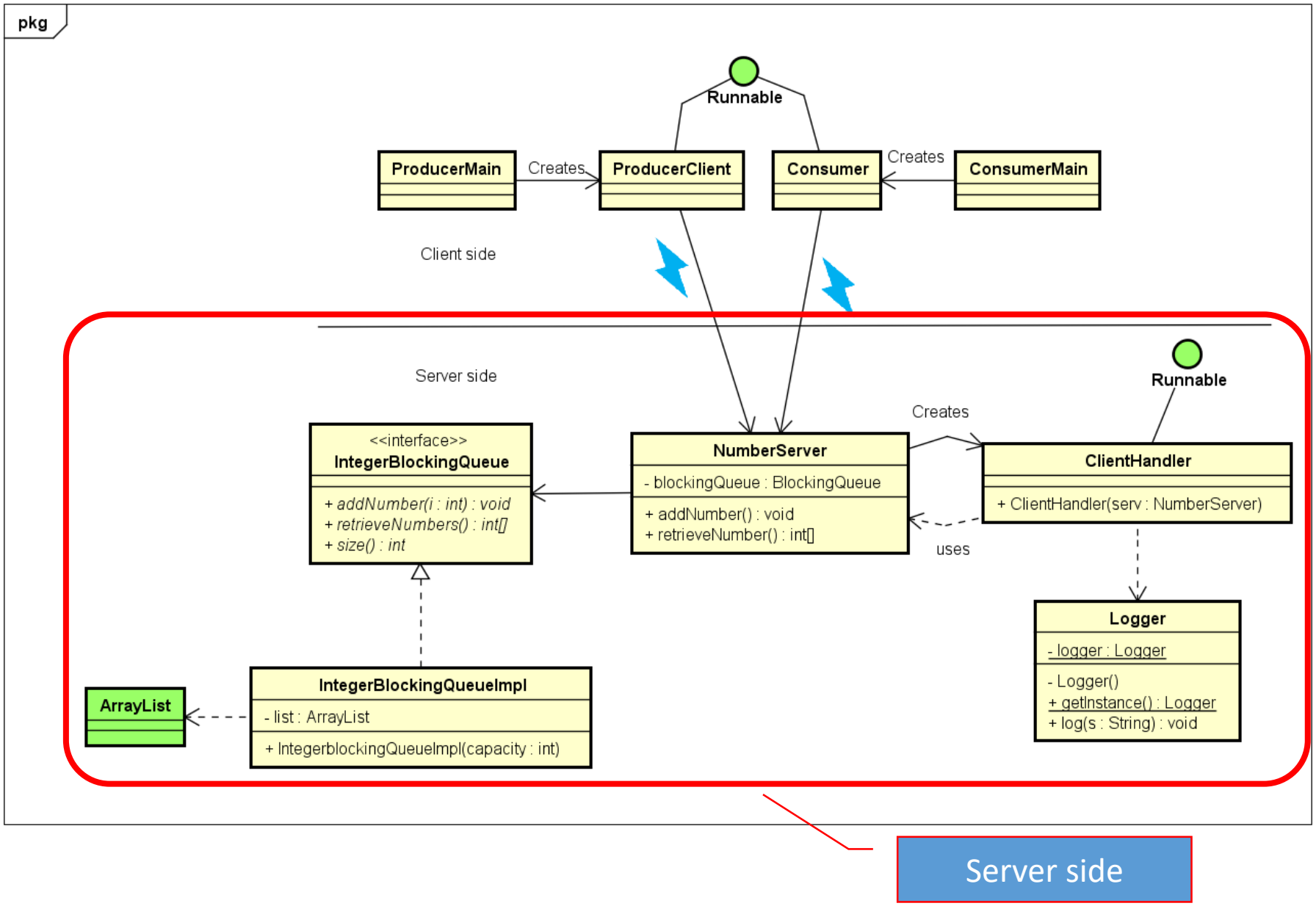
The Idea

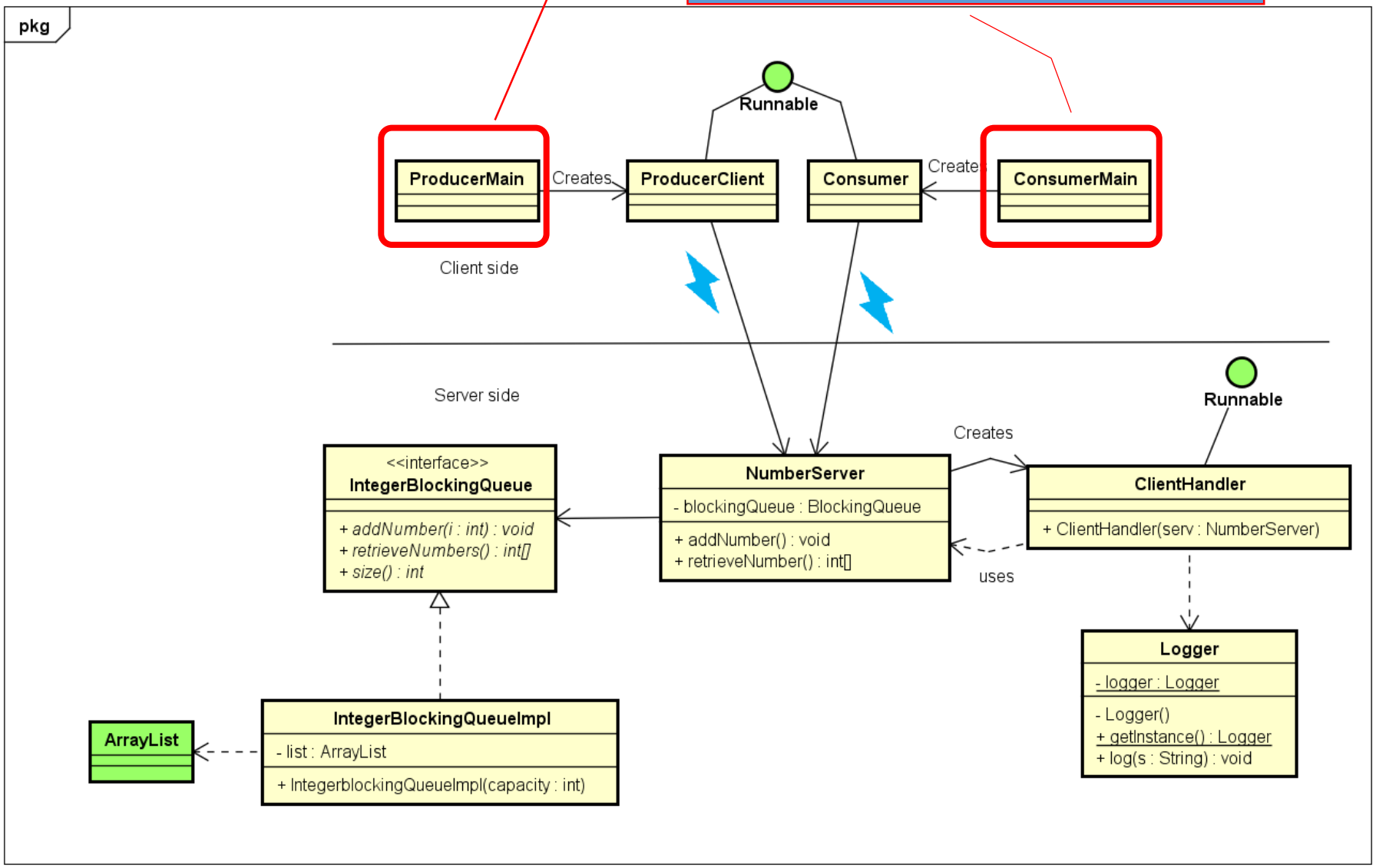


Producer added 1
Producer added 7
Producer added 5
Consumer retrieved 1, 7, 5
Consumer result: 36

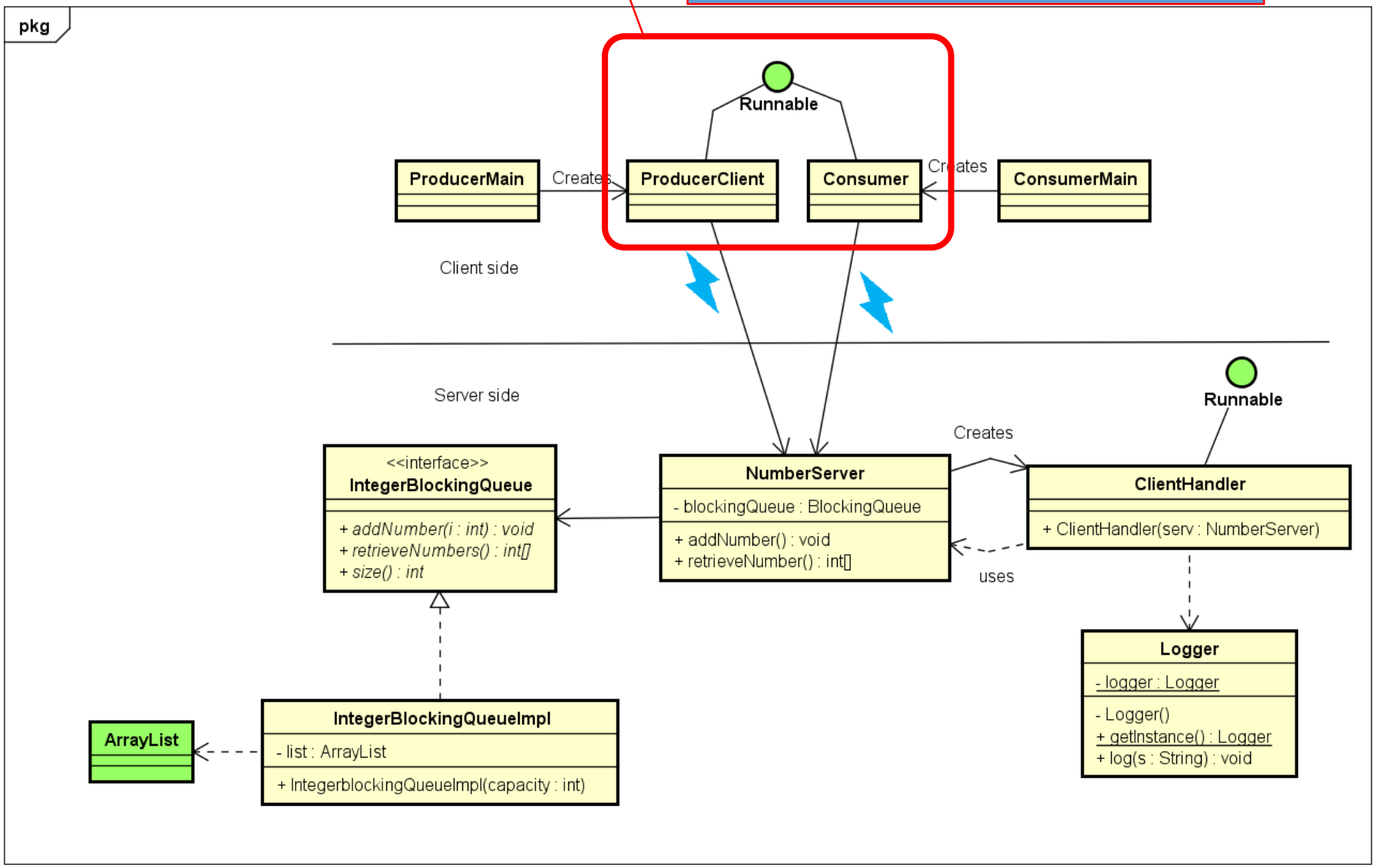

pkg
Runnable
ProducerMain
Creates
ProducerClient
Consumer
Creates
ConsumerMain
Client side
Server side
Runnable
Creates
<<interface>>
IntegerBlockingQueue
+ addNumber(i : int) : void
+ retrieveNumbers() : int[]
+ size() : int
NumberServer
- blockingQueue : BlockingQueue
+ addNumber() : void
+ retrieveNumber() : int[]
ClientHandler
+ ClientHandler(serv : NumberServer)
uses
Logger
- logger : Logger
- Logger()
+ getInstance() : Logger
+ log(s : String) : void
ArrayList
IntegerBlockingQueueImpl
- list : ArrayList
+ IntegerblockingQueueImpl(capacity : int)


Client side
pkg
Runnable
ProducerMain
Creates
ProducerClient
Consumer
Creates
ConsumerMain
Client side
Server side
Runnable
Creates
<<interface>>
IntegerBlockingQueue
+ addNumber(i : int) : void
+ retrieveNumbers() : int[]
+ size() : int
NumberServer
- blockingQueue : BlockingQueue
+ addNumber() : void
+ retrieveNumber() : int[]
ClientHandler
+ ClientHandler(serv : NumberServer)
uses
Logger
- logger : Logger
- Logger()
+ getInstance() : Logger
+ log(s : String) : void
ArrayList
IntegerBlockingQueueImpl
- list : ArrayList
+ IntegerblockingQueueImpl(capacity : int)

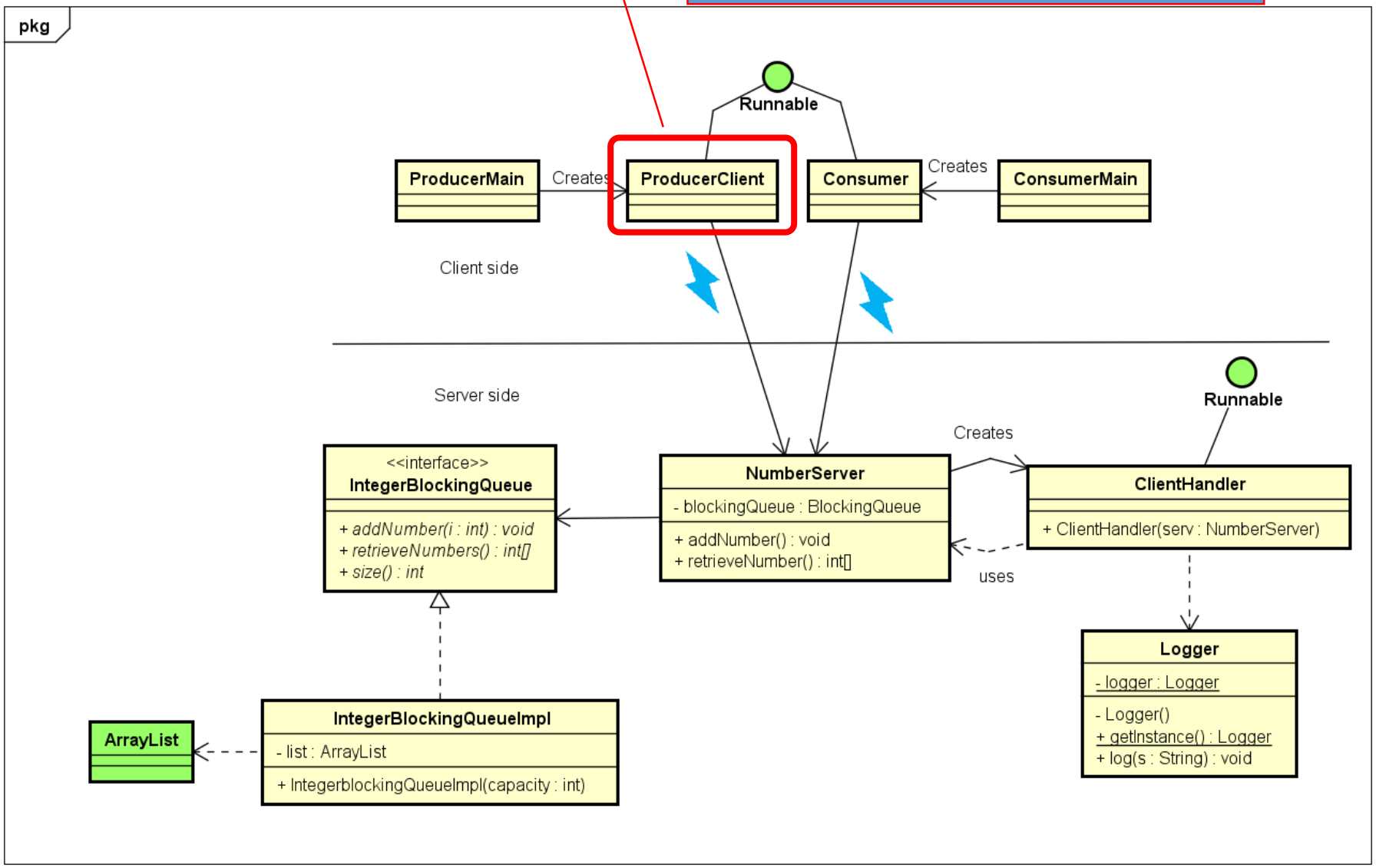

pkg
Runnable
ProducerMain
Creates
ProducerClient
Consumer
Creates
ConsumerMain
Client side
Server side
Runnable
Creates
<<interface>>
IntegerBlockingQueue
+ addNumber(i : int) : void
+ retrieveNumbers() : int[]
+ size() : int
NumberServer
- blockingQueue : BlockingQueue
+ addNumber() : void
+ retrieveNumber() : int[]
ClientHandler
+ ClientHandler(serv : NumberServer)
uses
Logger
- logger : Logger
- Logger()
+ getInstance() : Logger
+ log(s : String) : void
ArrayList
IntegerBlockingQueueImpl
- list : ArrayList
+ IntegerblockingQueueImpl(capacity : int)
Server side


Classes with a main method to start each client.
pkg
Runnable
ProducerMain
Creates
ProducerClient
Consumer
Creates
ConsumerMain
Client side
Server side
Runnable
Creates
<<interface>>
IntegerBlockingQueue
+ addNumber(i : int) : void
+ retrieveNumbers() : int[]
+ size() : int
NumberServer
- blockingQueue : BlockingQueue
+ addNumber() : void
+ retrieveNumber() : int[]
ClientHandler
+ ClientHandler(serv : NumberServer)
uses
Logger
- logger : Logger
- Logger()
+ getInstance() : Logger
+ log(s : String) : void
ArrayList
IntegerBlockingQueueImpl
- list : ArrayList
+ IntegerblockingQueueImpl(capacity : int)

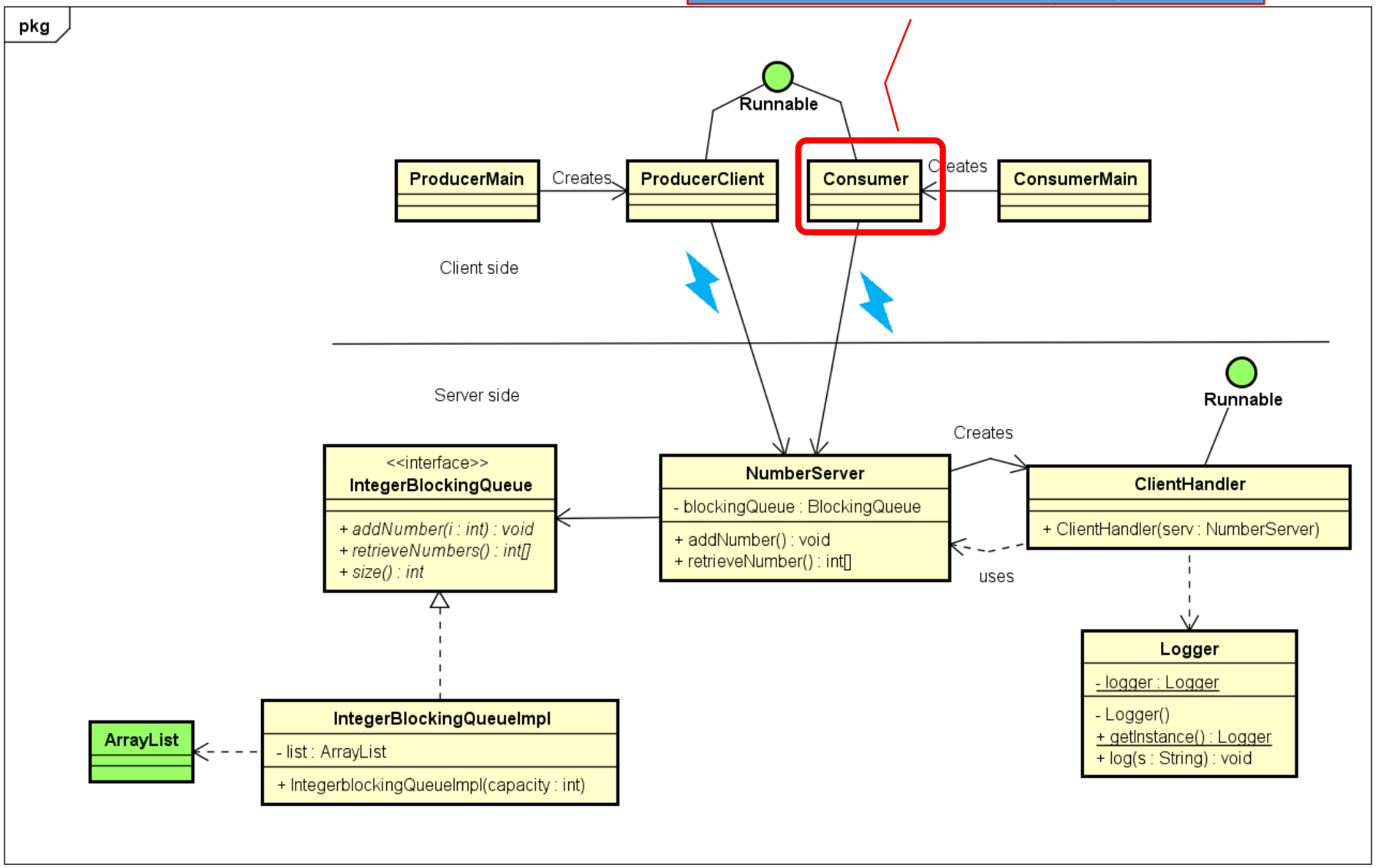
Two client programs. Both are Threads, started from the two main classes.



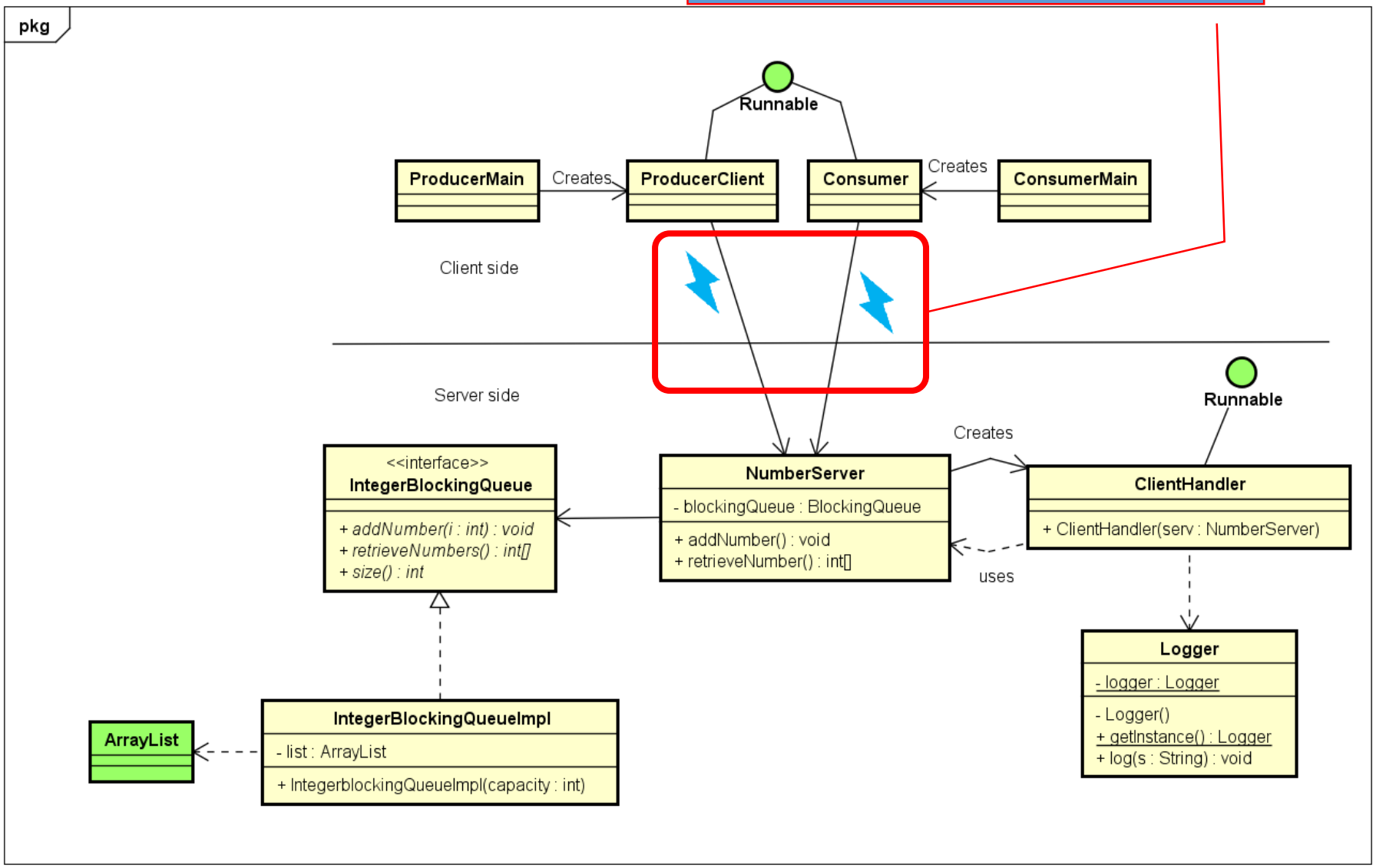
Generates random number, sends to server. In a while(true)

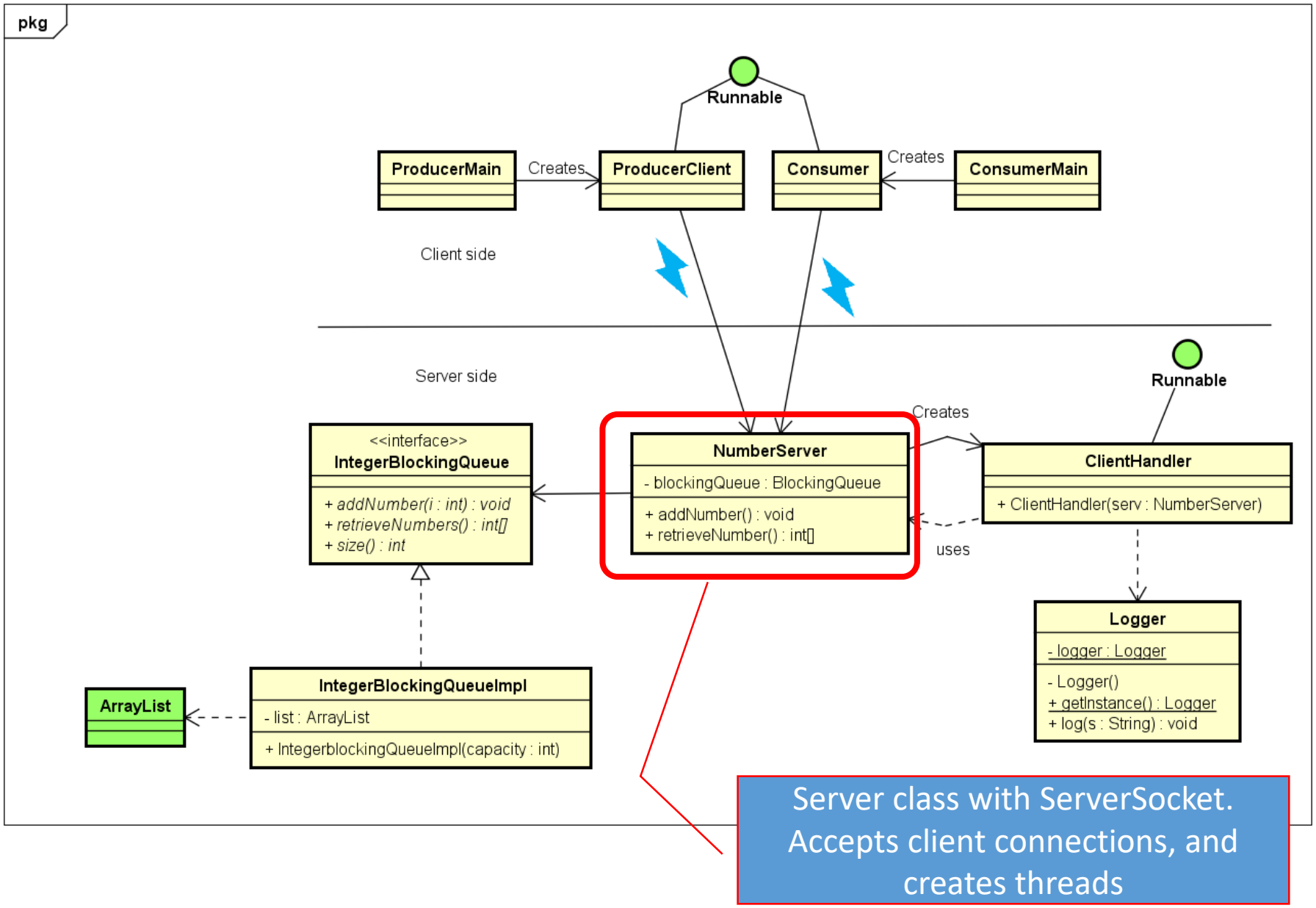


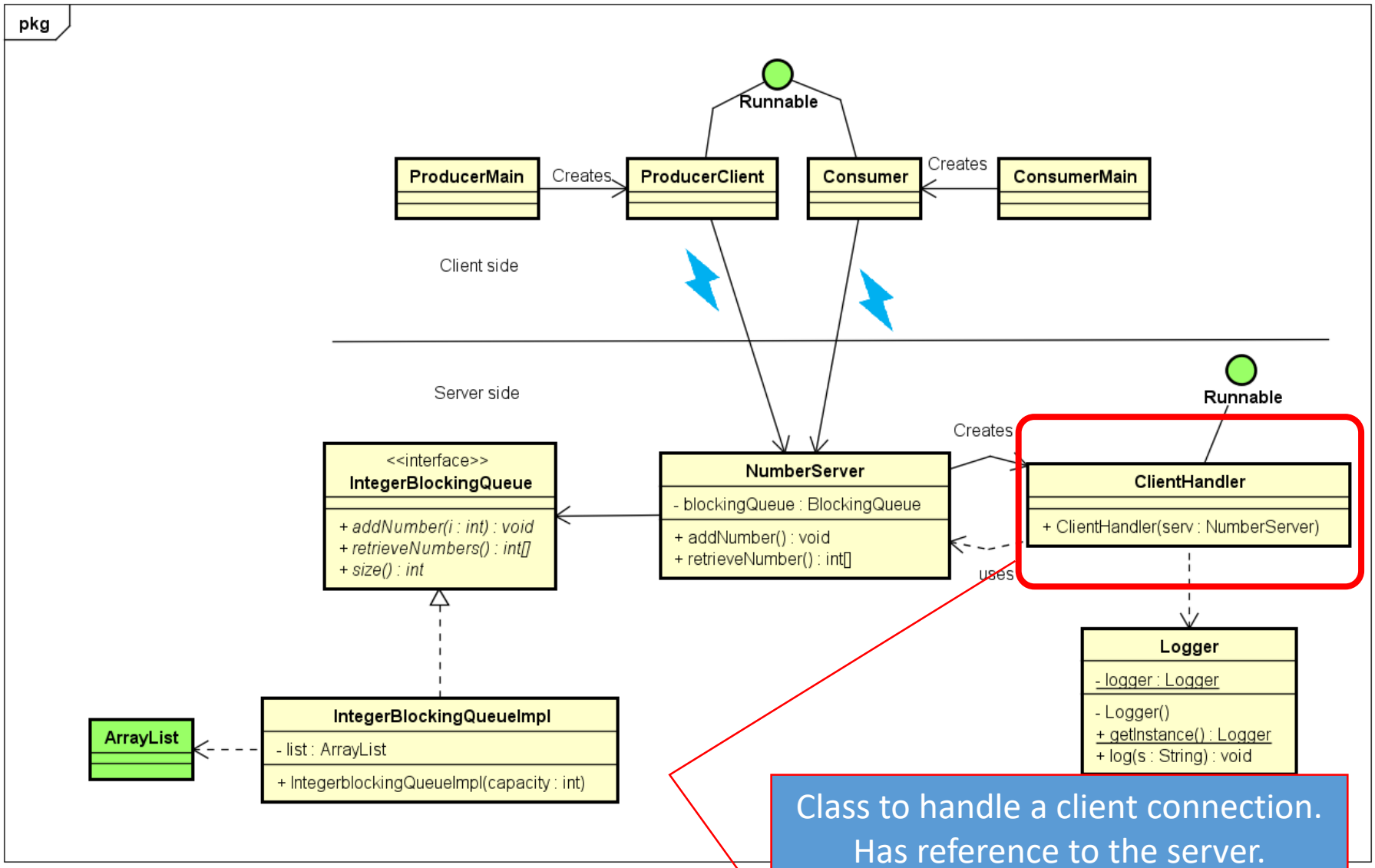
Retrieves 3 numbers at a time. Does calculation. Sends result back. Also in a while(true)



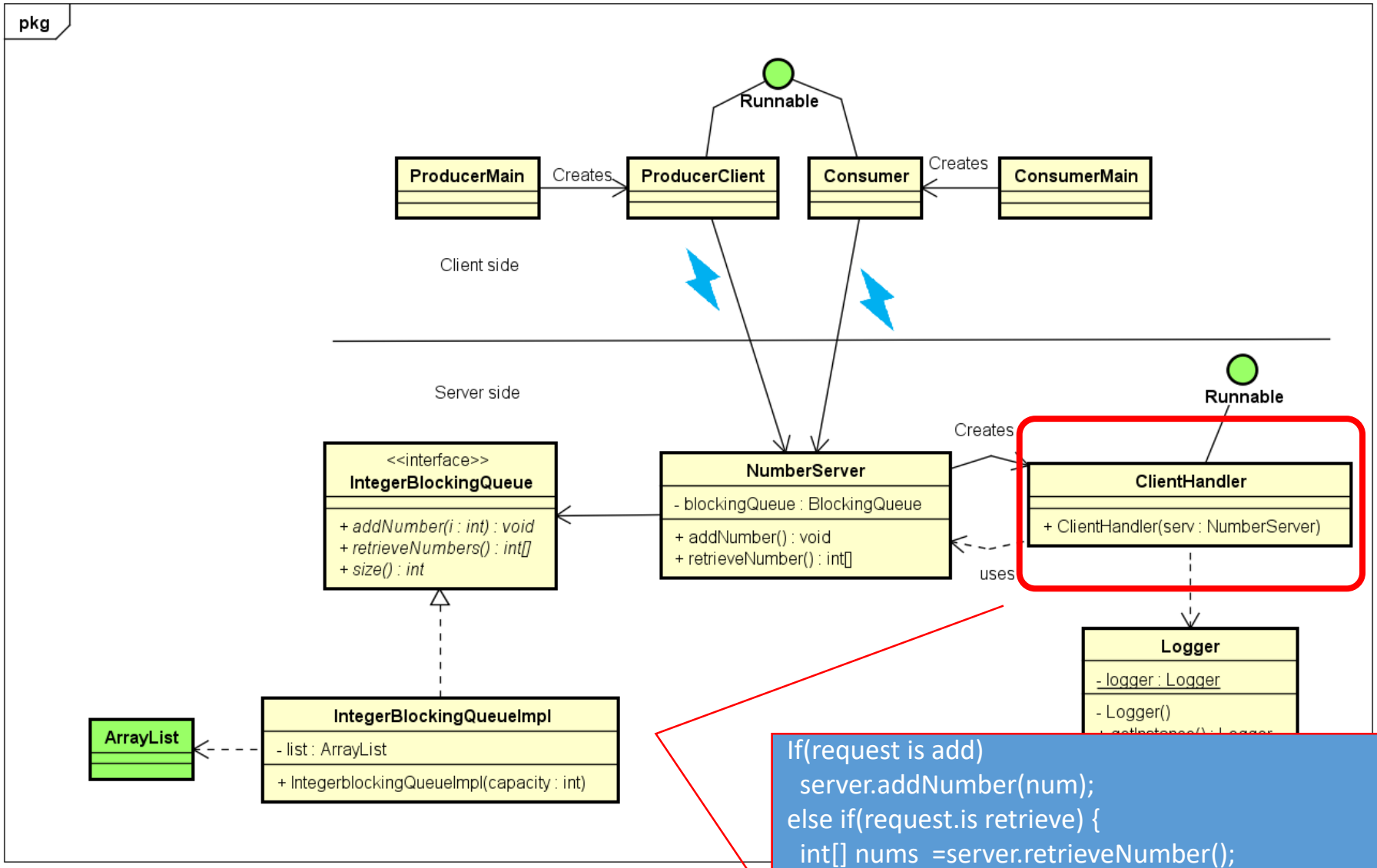
Socket connections



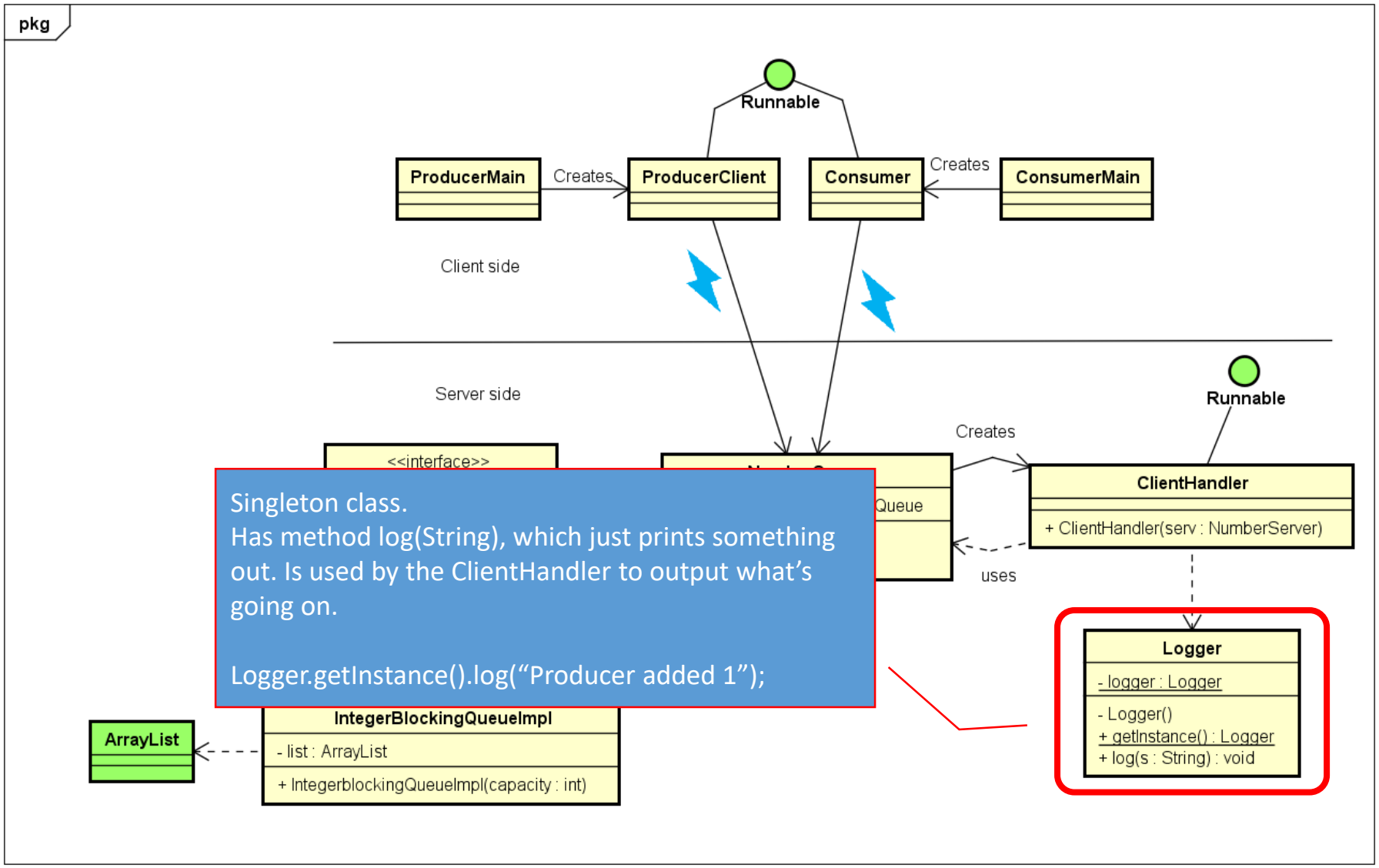

pkg
Runnable
ProducerMain
Creates
ProducerClient
Consumer
Creates
ConsumerMain
Client side
Server side
Runnable
Creates
<<interface>>
IntegerBlockingQueue
+ addNumber(i : int) : void
+ retrieveNumbers() : int[]
+ size() : int
NumberServer
- blockingQueue : BlockingQueue
+ addNumber() : void
+ retrieveNumber() : int[]
ClientHandler
+ ClientHandler(serv : NumberServer)
uses
Logger
- logger : Logger
- Logger()
+ getInstance() : Logger
+ log(s : String) : void
ArrayList
IntegerBlockingQueueImpl
- list : ArrayList
+ IntegerblockingQueueImpl(capacity : int)
Server class with ServerSocket. Accepts client connections, and creates threads

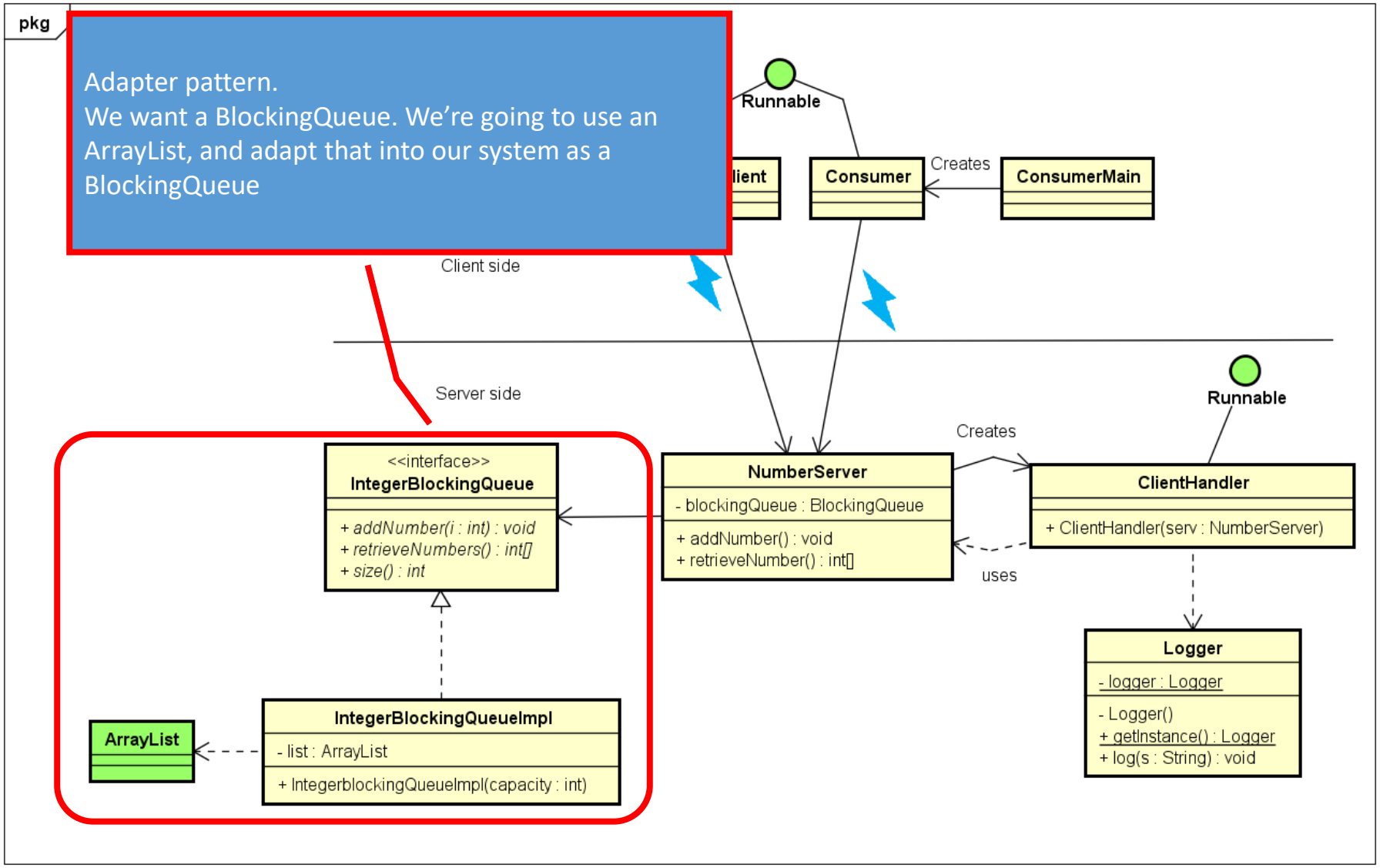


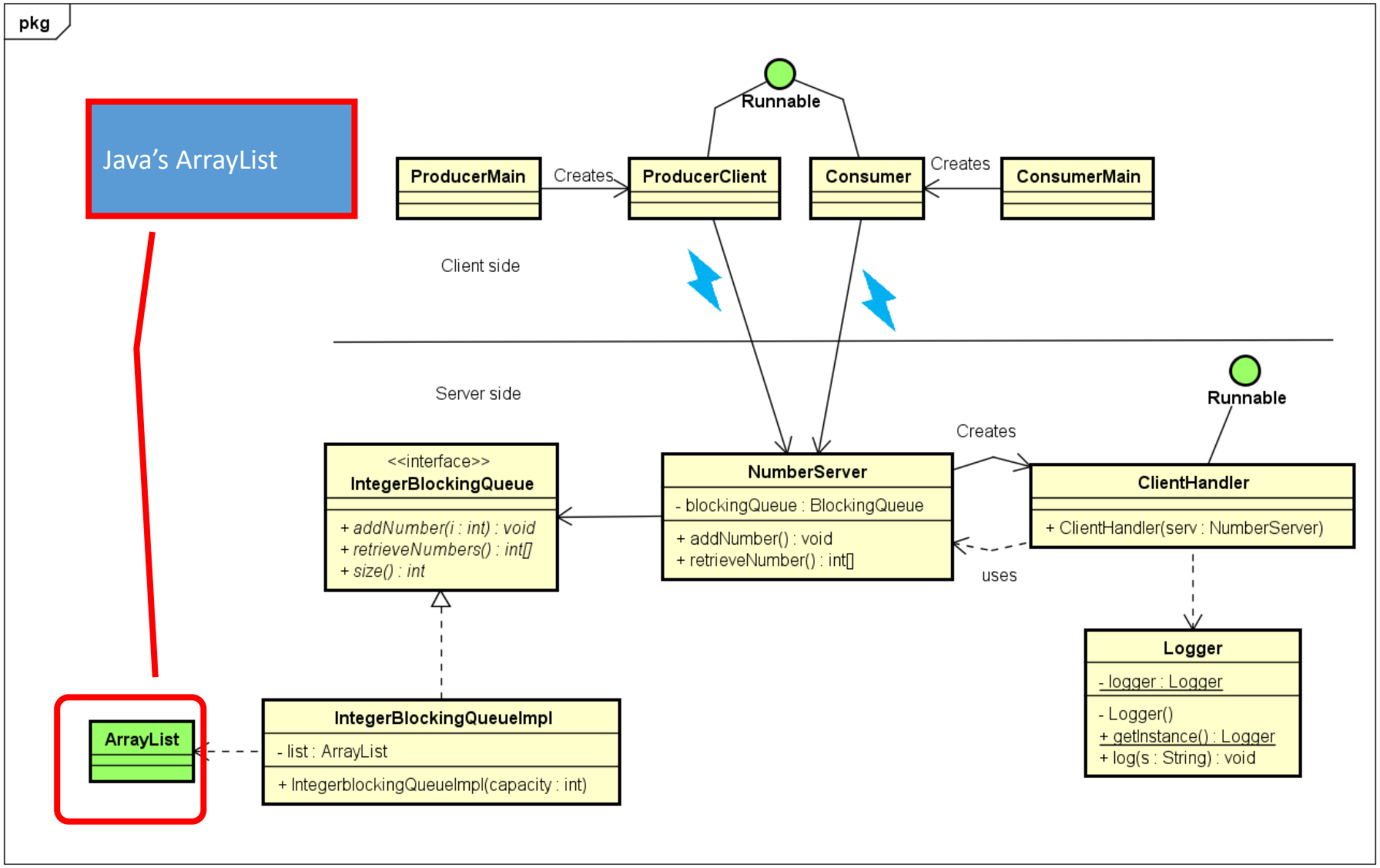
Class to handle a client connection. Has reference to the server. Implement protocol, so this can be used by both client types.

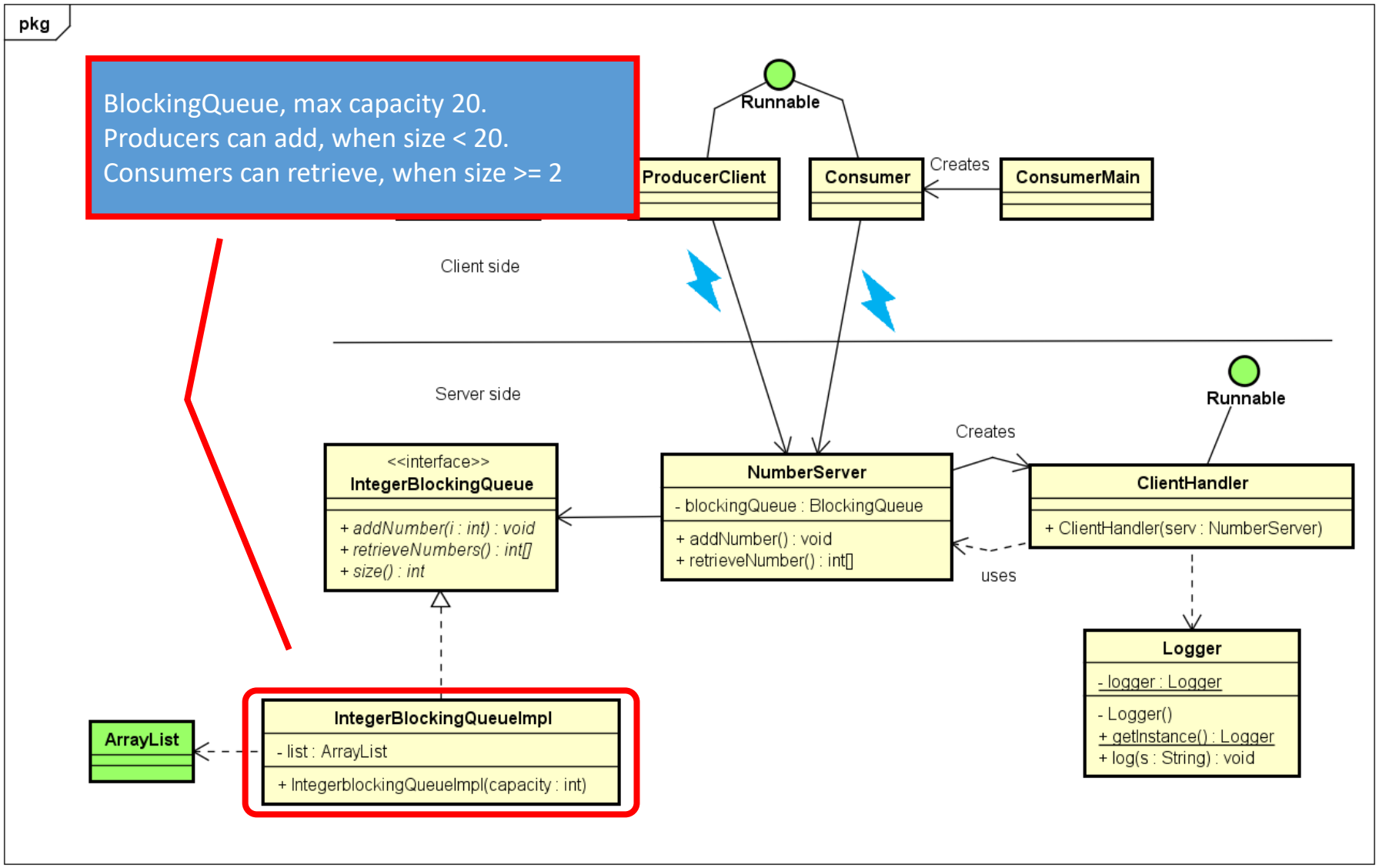


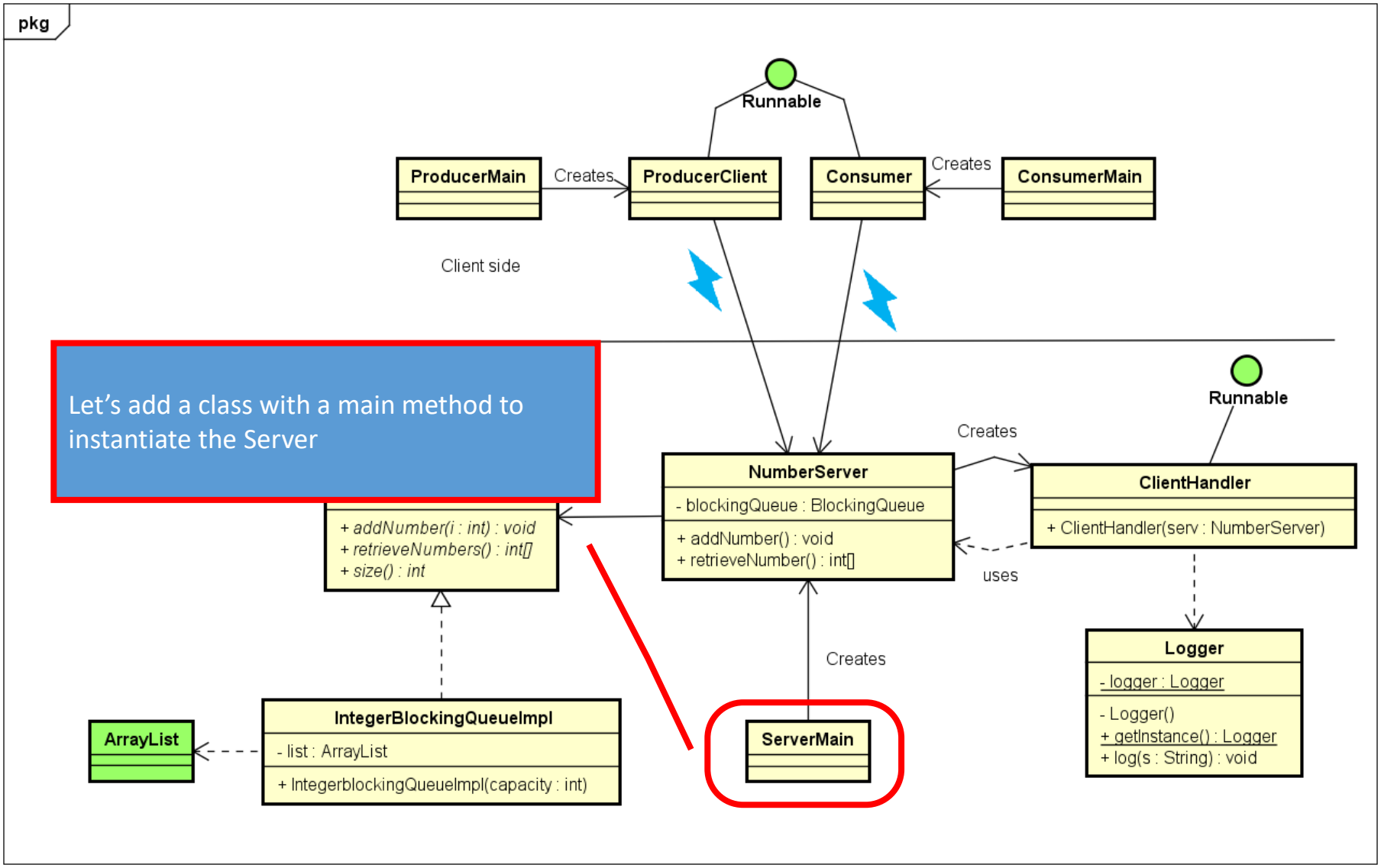

```
If(request is add)
  server.addNumber(num);
else if(request.is retrieve) {
  int[] nums  =server.retrieveNumber();
  // send number
....
```

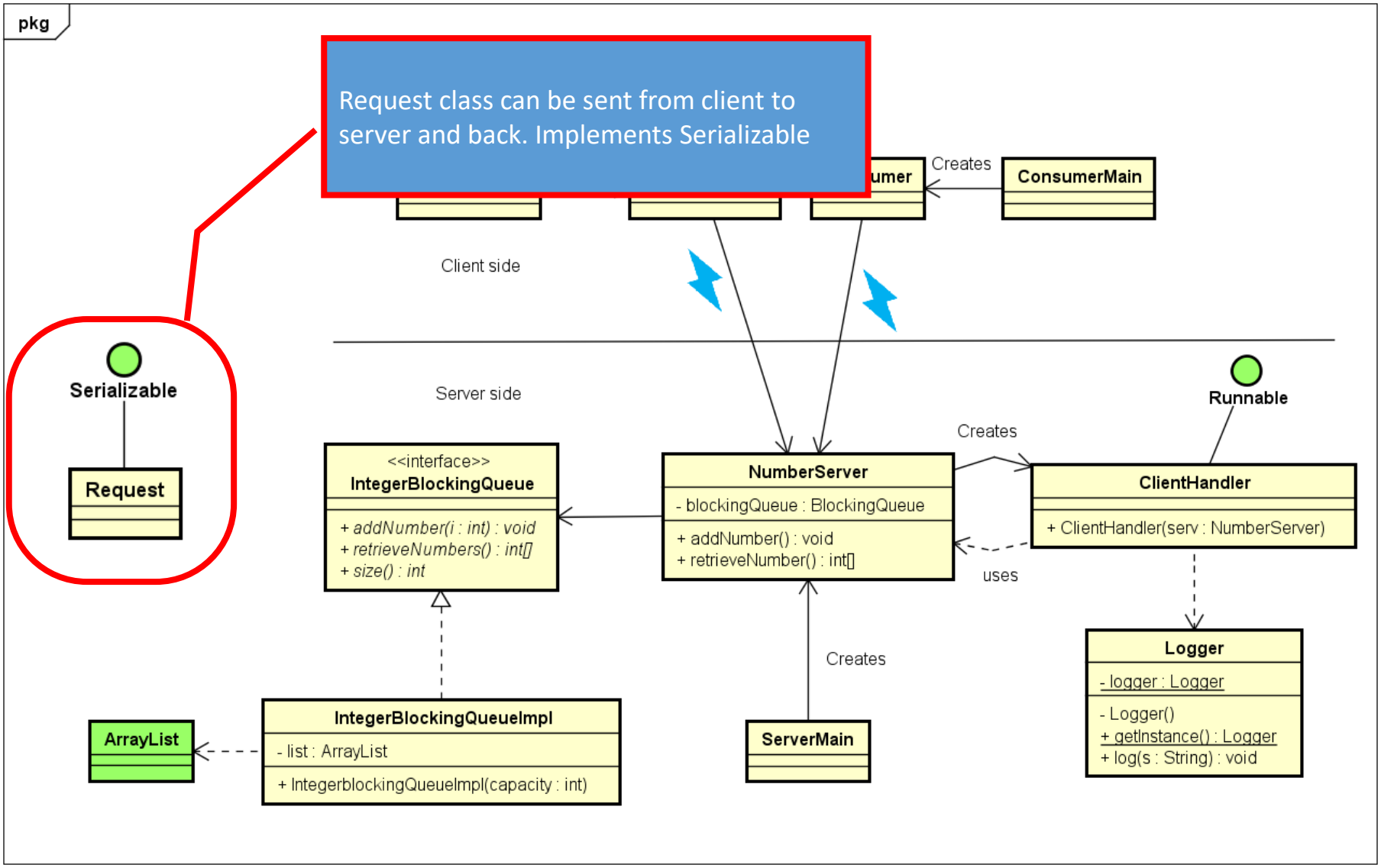

pkg
Runnable
ProducerMain
Creates
ProducerClient
Consumer
Creates
ConsumerMain
Client side
Server side
Runnable
Creates
<<interface>>
Queue
ClientHandler
+ ClientHandler(serv : NumberServer)
uses
Singleton class.
Has method log(String), which just prints something out. Is used by the ClientHandler to output what's going on.
Logger.getInstance().log("Producer added 1");
Logger
- logger : Logger
- Logger()
+ getInstance() : Logger
+ log(s : String) : void
ArrayList
IntegerBlockingQueueImpl
- list : ArrayList
+ IntegerblockingQueueImpl(capacity : int)


pkg
Adapter pattern.
We want a BlockingQueue. We're going to use an ArrayList, and adapt that into our system as a BlockingQueue
Runnable
lient
Consumer
Creates
ConsumerMain
Client side
Server side
Runnable
Creates
<<interface>>
IntegerBlockingQueue
+ addNumber(i : int) : void
+ retrieveNumbers() : int[]
+ size() : int
NumberServer
- blockingQueue : BlockingQueue
+ addNumber() : void
+ retrieveNumber() : int[]
ClientHandler
+ ClientHandler(serv : NumberServer)
uses
Logger
- logger : Logger
- Logger()
+ getInstance() : Logger
+ log(s : String) : void
ArrayList
IntegerBlockingQueueImpl
- list : ArrayList
+ IntegerblockingQueueImpl(capacity : int)


pkg
Java's ArrayList
Runnable
ProducerMain
Creates
ProducerClient
Consumer
Creates
ConsumerMain
Client side
Server side
Runnable
Creates
<<interface>>
IntegerBlockingQueue
+ addNumber(i : int) : void
+ retrieveNumbers() : int[]
+ size() : int
NumberServer
- blockingQueue : BlockingQueue
+ addNumber() : void
+ retrieveNumber() : int[]
ClientHandler
+ ClientHandler(serv : NumberServer)
uses
Logger
- logger : Logger
- Logger()
+ getInstance() : Logger
+ log(s : String) : void
ArrayList
IntegerBlockingQueueImpl
- list : ArrayList
+ IntegerblockingQueueImpl(capacity : int)


pkg
BlockingQueue, max capacity 20.
Producers can add, when size < 20.
Consumers can retrieve, when size >= 2
Runnable
ProducerClient
Consumer
Creates
ConsumerMain
Client side
Server side
Runnable
Creates
<<interface>>
IntegerBlockingQueue
+ addNumber(i : int) : void
+ retrieveNumbers() : int[]
+ size() : int
NumberServer
- blockingQueue : BlockingQueue
+ addNumber() : void
+ retrieveNumber() : int[]
ClientHandler
+ ClientHandler(serv : NumberServer)
uses
Logger
- logger : Logger
- Logger()
+ getInstance() : Logger
+ log(s : String) : void
ArrayList
IntegerBlockingQueueImpl
- list : ArrayList
+ IntegerblockingQueueImpl(capacity : int)


pkg
Runnable
ProducerMain
Creates
ProducerClient
Consumer
Creates
ConsumerMain
Client side
Let's add a class with a main method to instantiate the Server
Runnable
Creates
NumberServer
- blockingQueue : BlockingQueue
+ addNumber() : void
+ retrieveNumber() : int[]
ClientHandler
+ ClientHandler(serv : NumberServer)
+ addNumber(i : int) : void
+ retrieveNumbers() : int[]
+ size() : int
uses
Logger
- logger : Logger
- Logger()
+ getInstance() : Logger
+ log(s : String) : void
Creates
ArrayList
IntegerBlockingQueueImpl
- list : ArrayList
+ IntegerblockingQueueImpl(capacity : int)
ServerMain


pkg
Request class can be sent from client to server and back. Implements Serializable
Creates
ConsumerMain
Client side
Server side
Serializable
Request
Runnable
Creates
<<interface>>
IntegerBlockingQueue
+ addNumber(i : int) : void
+ retrieveNumbers() : int[]
+ size() : int
NumberServer
- blockingQueue : BlockingQueue
+ addNumber() : void
+ retrieveNumber() : int[]
ClientHandler
+ ClientHandler(serv : NumberServer)
uses
Logger
- logger : Logger
- Logger()
+ getInstance() : Logger
+ log(s : String) : void
Creates
ArrayList
IntegerBlockingQueueImpl
- list : ArrayList
+ IntegerblockingQueueImpl(capacity : int)
ServerMain

Deadline

- Friday the 30th of November
- Must be handed in and accepted to get access to the exam
- At the exam you will show and explain parts of your exercises