

Szymon Zieliński
nr indexu 296722

Zad 1a (1 punkt). Zaobserwuj działanie mechanizmu niezawodności w różnych scenariuszach potwierdzeń wiadomości.

- Po otrzymaniu wiadomości:

- Wiadomość zostaje normalnie odebrana w obydwu przypadkach

Messages		
Ready	Unacked	Total
0	0	0

- Po przetworzeniu wiadomości:

- Konsument zostaje zrestartowany po zakończeniu przetwarzania wiadomości:

Messages		
Ready	Unacked	Total
0	0	0

Wiadomość zostaje normalnie odebrana

- Konsument zostaje zrestartowany w trakcie przetwarzania wiadomości:

Messages		
Ready	Unacked	Total
1	0	1

Wiadomość przechodzi po chwili ze stanu unacked do stanu ready, można ją ponownie odebrać.

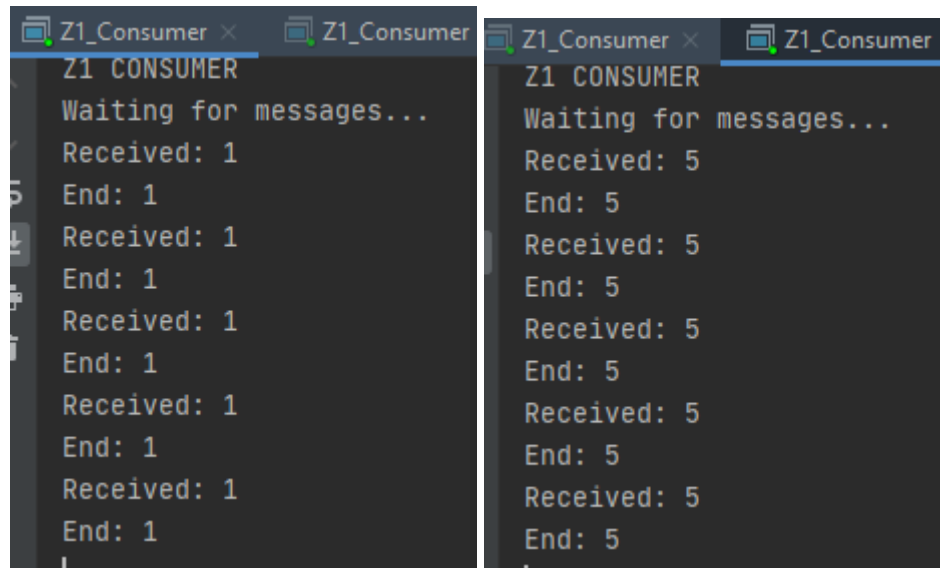
- Który sposób potwierdzeń zapewnia większą niezawodność?
Potwierdzenie dopiero po przetworzeniu wiadomości.
- Co się stanie, jeśli nie będziemy potwierdzać wiadomości ani po otrzymaniu, ani po przetworzeniu?

Messages		
Ready	Unacked	Total
0	1	1

Pozostaje w stanie unacked do momentu akceptacji lub odrzucenia (np. wyłączenia consumera, wtedy przechodzi ponownie do stanu ready)

Zad 1b (1 punkt). Zaobserwuj działanie mechanizmu load-balancing

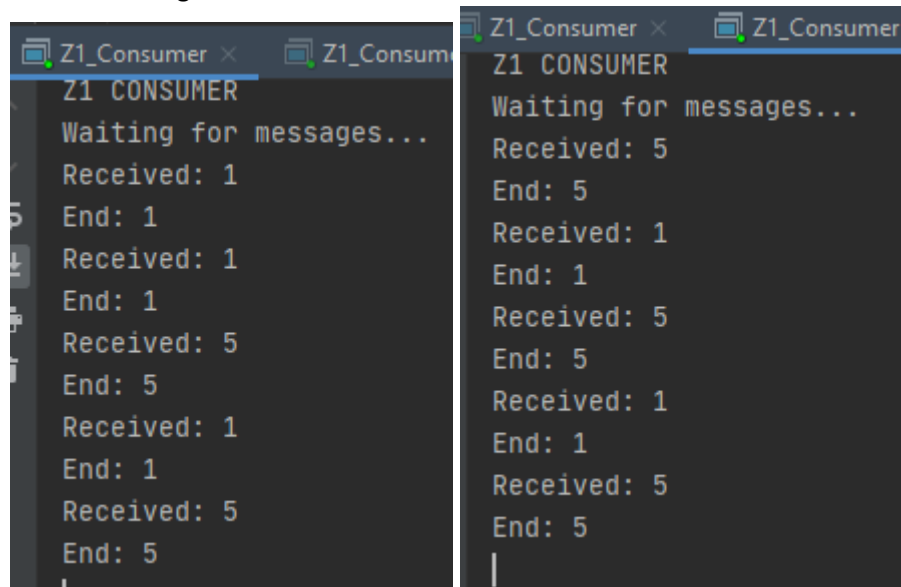
➤ Round-robin:



```
Z1_Consumer x Z1_Consumer
Z1 CONSUMER
Waiting for messages...
Received: 1
End: 1
Received: 1
End: 1
Received: 1
End: 1
Received: 1
End: 1
Received: 1
End: 1
Received: 1
End: 1

Z1_Consumer x Z1_Consumer
Z1 CONSUMER
Waiting for messages...
Received: 5
End: 5
Received: 5
End: 5
Received: 5
End: 5
Received: 5
End: 5
Received: 5
End: 5
Received: 5
End: 5
```

➤ Load-balancing:

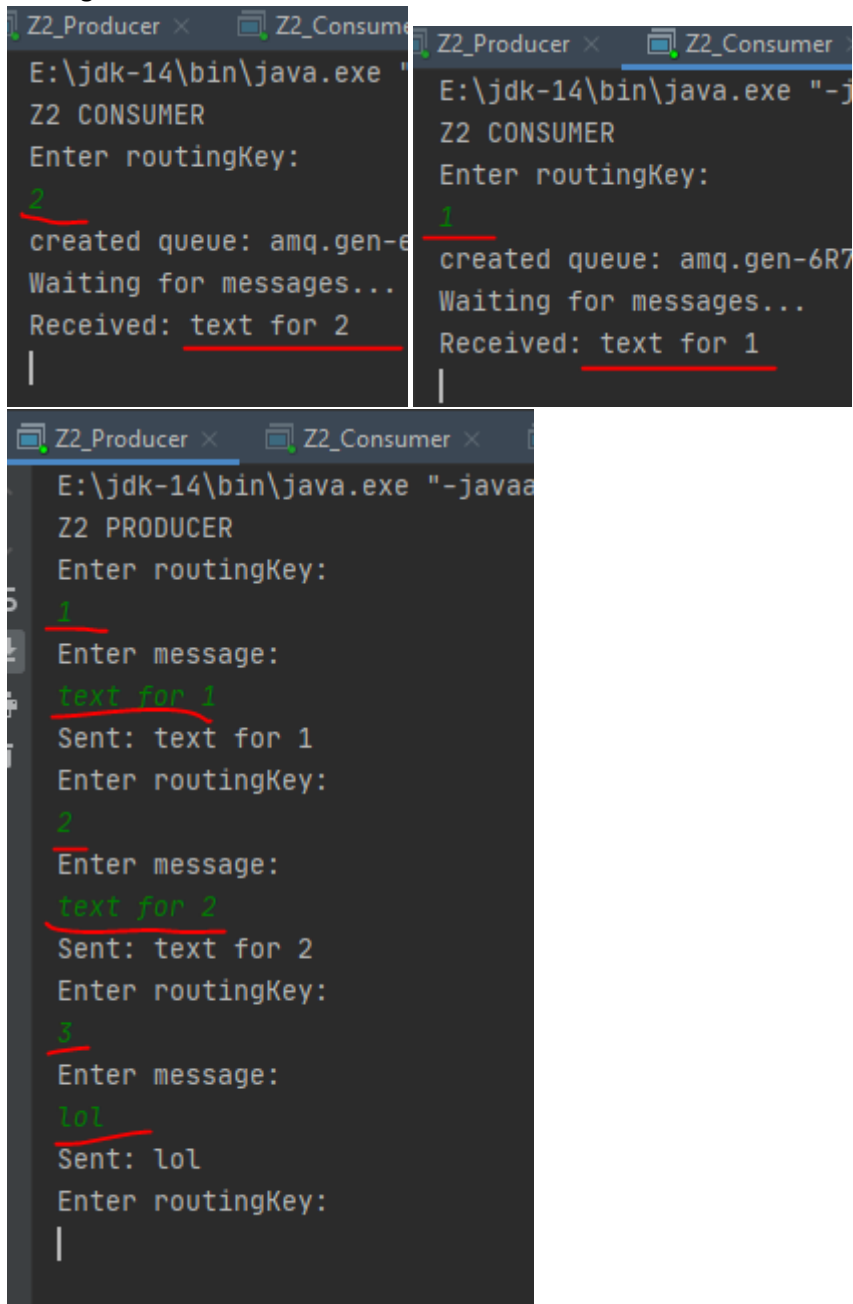


```
Z1_Consumer x Z1_Consumer
Z1 CONSUMER
Waiting for messages...
Received: 1
End: 1
Received: 1
End: 1
Received: 5
End: 5
Received: 1
End: 1
Received: 5
End: 5
Received: 5
End: 5

Z1_Consumer x Z1_Consumer
Z1 CONSUMER
Waiting for messages...
Received: 5
End: 5
Received: 1
End: 1
Received: 5
End: 5
Received: 1
End: 1
Received: 5
End: 5
Received: 5
End: 5
```

Zad 2 (2 punkty). Zaimplementuj oraz pokaż w działaniu routing Direct oraz Topic

➤ routingu Direct:



```
E:\jdk-14\bin\java.exe "Z2_PRODUCER"
Z2 PRODUCER
Enter routingKey:
1
Enter message:
text for 1
Sent: text for 1
Enter routingKey:
2
Enter message:
text for 2
Sent: text for 2
Enter routingKey:
3
Enter message:
lol
Sent: lol
Enter routingKey:
|

E:\jdk-14\bin\java.exe "Z2_CONSUMER"
Z2 CONSUMER
Enter routingKey:
1
created queue: amq.gen-6R7
Waiting for messages...
Received: text for 1
|

E:\jdk-14\bin\java.exe "Z2_CONSUMER"
Z2 CONSUMER
Enter routingKey:
2
created queue: amq.gen-6R7
Waiting for messages...
Received: text for 2
|
```

➤ routingu Topic:

```
Z2_Producer x Z2_Consumer x
E:\jdk-14\bin\java.exe "-jav
Z2 CONSUMER C2
Enter routingKey:
*.b.#
created queue: amq.gen-zQv5y
Waiting for messages...
Received: message to a.b.1
Received: message for b.b.2
|

Z2_Producer x Z2_Consumer x
E:\jdk-14\bin\java.exe "-jav
Z2 CONSUMER C1
Enter routingKey:
*.b.1
created queue: amq.gen-oe803
Waiting for messages...
Received: message to a.b.1
|

Z2_Producer x Z2_Consumer x
E:\jdk-14\bin\java.exe "-java
Z2 PRODUCER
Enter topic:
u.a.1 C3
Enter message:
message to u.a.1
Sent: message to u.a.1
Enter topic:
a.b.1 C1, C2
Enter message:
message to a.b.1
Sent: message to a.b.1
Enter topic:
b.b.2 C2
Enter message:
message for b.b.2
Sent: message for b.b.2
Enter topic:
|

Z2_Producer x Z2_Consumer x
E:\jdk-14\bin\java.exe "-java
Z2 CONSUMER C3
Enter routingKey:
#.a.1
created queue: amq.gen-JIyTyV
Waiting for messages...
Received: message to u.a.1
|
```

Topic ma taką przewagę nad direct, że jeśli chcemy wysłać wiadomości do dwóch consumerów przez direct to musimy im nadać ten sam klucz, co oznacza że nie będziemy w stanie wysłać wiadomości tylko do jednego z nich. (Topic pozwala tworzyć grupy odbiorców)