**Sistem Paralel dan Terdistribusi (CSH3J3)**

**Tugas 3: Message Passing**

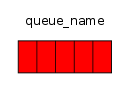
**RabbitMQ**

RabbitMQ adalah pengantar pesan (*message broker*). RabbitMQ akan menerima pesan kemudian mengirimkan ke tujuannya. Beberapa istilah yang digunakan dalam RabbitMQ atau sistem terdistribusi secara umum.

*Producing* berarti mengirim pesan. Sebuah program yang mengirim pesan disebut *produser. Produser* digambarkan dengan huruf P dalam lingkaran.



*Queue* adalah antrian yang ada dalam RabbitMQ. Pesan akan disimpan di dalam *queue*. Panjang *Queue* tidak terbatas dan dapat menyimpan pesan sebanyak mungkin (hanya dibatasi oleh ukuran memory komputer). *Producer* akan mengirimkan pesan ke *queue* dan *consumer* akan mengambil pesan dari *queue*. *Queue* digambarkan seperti gambar berikut ini.



*Consumer* adalah program yang menerima pesan.



Catatan: *producer, queue (rabbitMQ) dan consumer* tidak berada dalam komputer yang sama pada sistem yang sebenarnya.

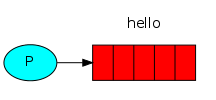
**1. Hello World**

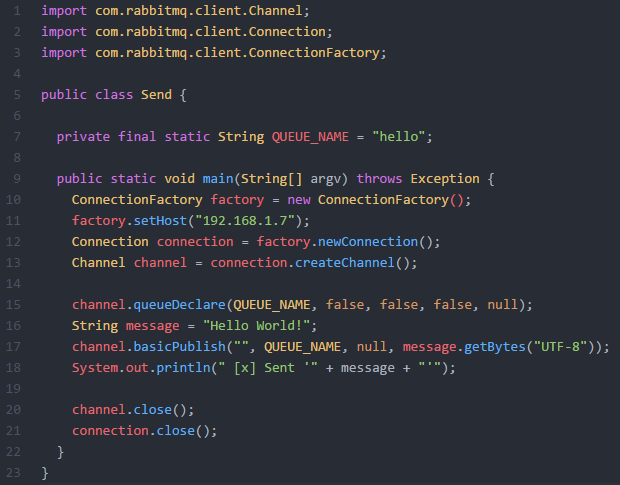
Pada tutorial ini, kita akan membuat 2 program java. Producer yang akan mengirimkan sebuah pesan dan sebuah consumer yang akan menrima pesan kemudian menampilkannya. P adalah producer yang akan kita buat, C adalah consumer yang akan menrima pesan dan menampilkan. Queue ditengah adalah antrian yang dibuat oleh RabbitMQ server. P, C dan Queue berada pada komputer yang berbeda-beda.



**Producer**

Program producer kita namai dengan **Send**. Producer (IP 192.168.1.5) akan berhubungan dengan RabbitMQ server (192.168.1.7), mengirim sebuah pesan (Hello world!) kemudian keluar.



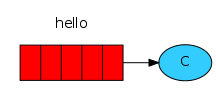


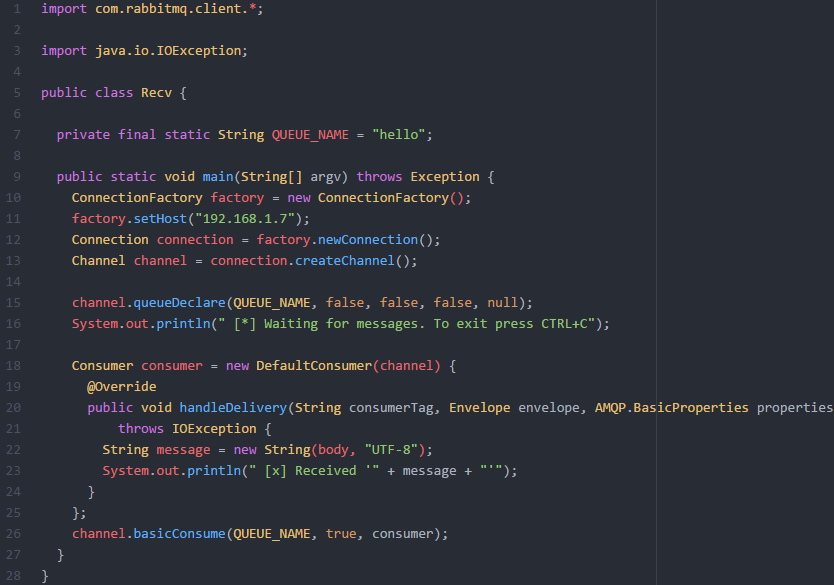
Penjelasan

* Baris 1-3: kita membutuhkan beberapa class yang berada di rabbitmq-client.jar untuk membuat koneksi dan channel.
* Baris 7: deklarasi nama queue yang akan kita gunakan yaitu “hello”. Perhatikan bahwa nama queue adalah private, final dan static artinya variabel tersebut tidak bisa diubah oleh program lain.
* Baris 10-13: setup koneksi dan channel. Pertama kita buat objek baru bernama factory. Factory tersebut akan kita gunakan untuk membuka koneksi, membuat channel, mengirim pesan, menutup channel dan menutup koneksi. Perhatikan perbedaan antara koneksi dan channel. Pada koneksi, produser akan dihubungkan dengan Server RabbitMQ yang ada di komputer lain dengan IP address 192.168.1.7. Setelah terjadi koneksi, kita kemudian membuat channel dan mengirimkan pesan melalui channel tersebut.
* Baris 15: kita membuat channel baru dengan nama “hello”
* Baris 16: deklarasi pesan yang akan kita kirimkan
* Baris 17:perintah untuk mengirimkan pesan yang kita buat. Kita menggunakan channel.basicPublish() untuk mengirim pesan biasa. Terdapat beberapa metode lain untuk mengirim pesan. Tetapi untuk saat ini basicPublish sudah cukup.
* Baris 18: mengeluarkan output ke terminal kita (komputer 192.168.1.5) yang menandakan bahwa pesan telah dikirm.
* Baris 20-21: pesan selesai dikirim, koneksi dan channel harus ditutup

**Consumer**

Penerima kita akan memperoleh pesan dari RabbitMQ server. Tidak seperti Producer yang mengirim 1 pesan, consumer akan terus “mendengarkan” channel hingga pesan tersebut tiba kemudian menampilkannya pada terminal.





**Penjelasan**

* Baris 1-3: mirip dengan Send, kita membutuhkan beberapa class di rabbitmq-client.jar untuk koneksi dan channel
* Baris 7-16: setup mirip juga dengan Send. Catatan penting yang perlu diperhatikan. Nama queue harus sama dengan yang digunakan oleh Send. Kita juga harus mendeklarasikan queue karena kita ingin memastikan bahwa queue ada sebelum kita mencoba mengkonsumsi pesan darinya.
* Baris 18-25: kita akan menggunakan class DefaultConsumer() yang telah ada. Karena pesan dikirim secara asinkron kita tidak tahu pesan akan sampai. Pada DefaultConsumer() telah ada mekanisme yang membantu kita jika pesan telah sampai. Kita cukup menulis perintah dalam kelas tersebut sesuai dengan keinginan kita. Jika kita tidak menggunakan DefaultConsumer() maka kita harus membuat mekanisme yang memberikan notifikasi jika pesan telah tiba baru kemudian memproses pesan tersebut.
* Baris 26: program mendengarkan pesan pada channel “hello”

**Running**

1. Compile Send

*javac -cp .:commons-io-2.5.jar:slf4j-api-1.7.21.jar:slf4j-simple-1.7.21.jar:amqp-client-4.0.0.jar Send.java*

1. Compile Recv

*javac -cp .:commons-io-2.5.jar:slf4j-api-1.7.21.jar:slf4j-simple-1.7.21.jar:amqp-client-4.0.0.jar Recv.java*

1. Jalankan RabbitMQ Server
2. Jalankan Recv

*java -cp .:commons-io-2.5.jar:slf4j-api-1.7.21.jar:slf4j-simple-1.7.21.jar:amqp-client-4.0.0.jar Recv*

1. Jalankan Send

*java -cp .:commons-io-2.5.jar:slf4j-api-1.7.21.jar:slf4j-simple-1.7.21.jar:amqp-client-4.0.0.jar Send*

***Kode telah berhasil dijalankan pada Sistem operasi Linux Mint 17 Rosa.***

**Tugas**

1. Jalankan Send sebanyak 5 kali kemudian Jalankan Recv. Berikan observasi!
2. Buatlah sebuah producer yang mengirim “hello world” sebanyak 10x dan sebuah producer yang mengirim “good morning!” sebanyak 10x. Hubungkan 2 producer tadi ke queue bernama “greeting” yang terhubung ke 1 consumer.

**2. Work Queue**

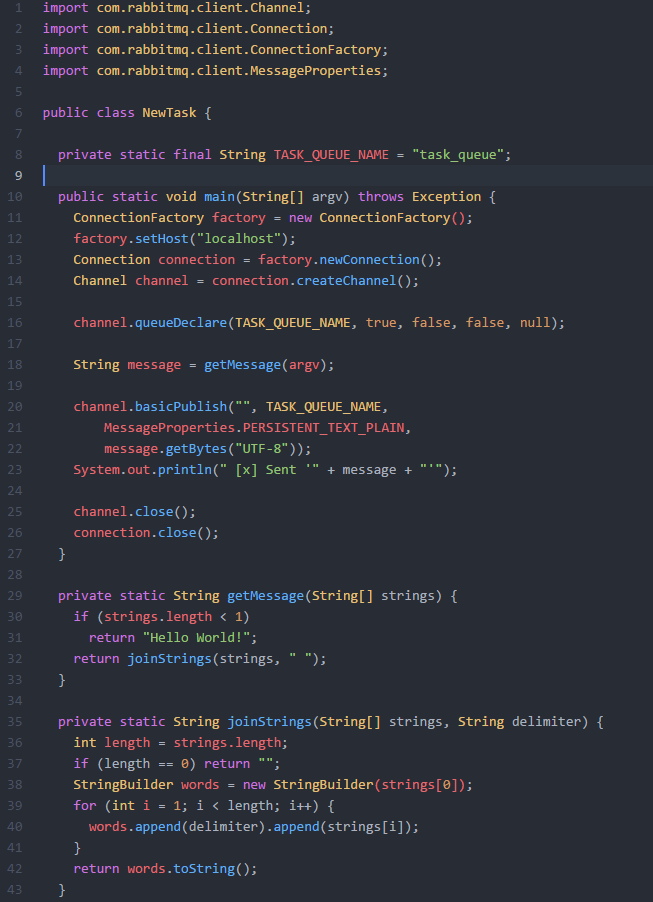
Pada tutorial ke-2 ini akan dijelaskan konsep work queue. Konsep utama work queue adalah mendistribusikan pekerjaan (task) kepada beberapa pekerja. Kita menyamarkan task sebagai pesan dan mengirimkan pesan ke queue. Jika ada pekerja yang sedang menganggur maka, ia akan menerima pesan dan mengerjakan task yang telah diberikan.



Kita akan mensimulasikan suatu pekerjaan (task) dengan menggunakan fungsi sleep(). Kita akan merepresentasikan kekomplekan task dengan dot (.), semakin banyak dot maka task yang dikerjakan akan semakin lama. Contoh: ketika pekerja menerima pesan dengan 3 dot yaitu: hello … maka pekerja akan sleep() selama 3 detik dan tidak bisa mengerjakan task lain.

**NewTask**

Kita akan mengubah sedikit source code Send.java menjadi NewTask.java untuk melakukan simulasi diatas.. NewTask akan menerima input parameter berupa string mengubahnya menjadi pesan dan mengirimkan pesan tersebut. Contoh: “java NewTask task …” maka pesan yang dikirim adalah hello task … dan pekerja yang menerima pesan tersebut akan sleep() selama 3 detik.

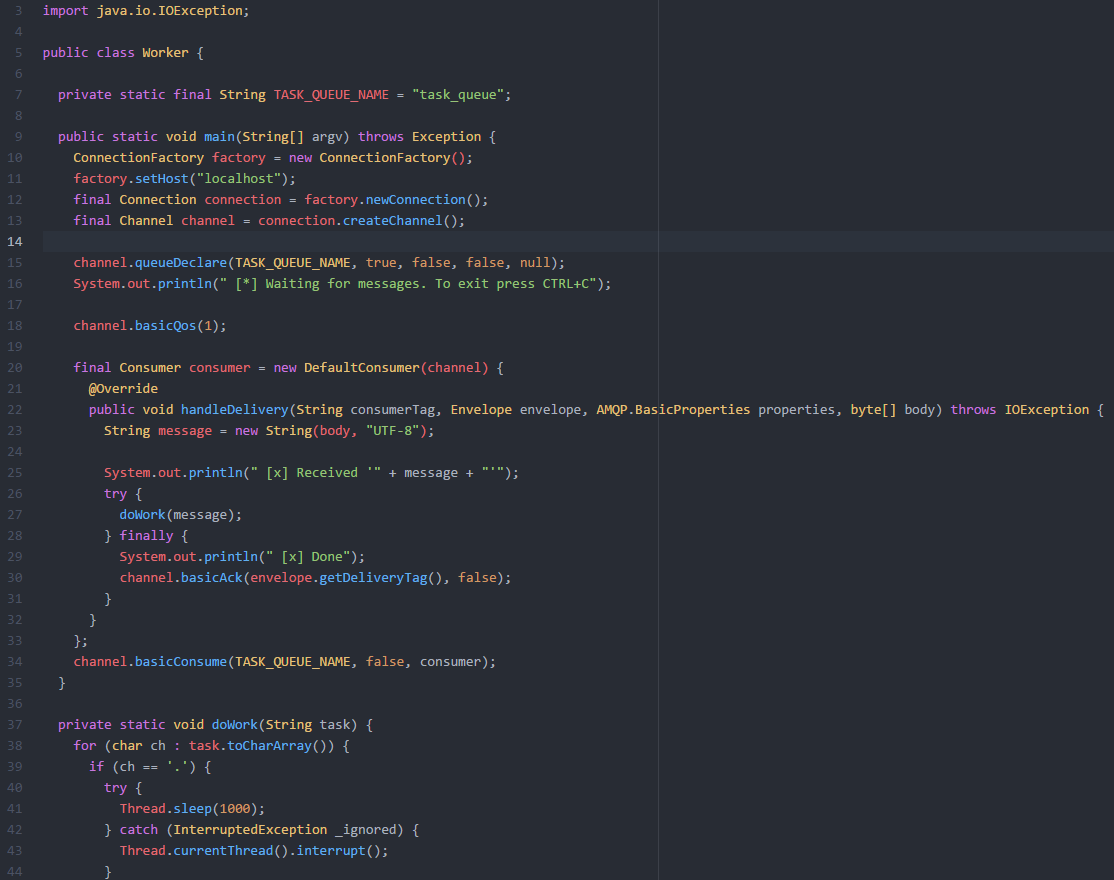


Penjelasan:

* Baris 18: NewTask.java sangat mirip dengan Send.java. Send.java hanya mengirim *hello world* tetapi NewTask.java dapat mengirimkan pesan apapun yang diberikan oleh user melalui *command prompt.* Hal tersebut dilakukan pada baris 18. Fungsi getMessage() akan dipanggil, getMessage() kemudian memanggil fungsi joinString() yang mengambil parameter dari *command prompt.*

**Worker**

Kita juga akan mengubah sedikit Recv.java menjadi Worker.java sehingga jika pekerja menerima pesan maka ia akan sleep sebanyak dot yang ia terima.



Penjelasan:

* Baris 27: Worker.java mirip dengan Recv.java dengan perbedaan utama yaitu worker akan sleep() sesuai dengan banyaknya dot yang worker terima dalam pesan. Hal ini dilakukan pada baris 27. Fungsi doWork(), baris 37-44, adalah menghitung jumlah dot (baris 38-39) dan memanggil fungsi sleep (baris 41). Ketika worker sleep() maka worker tidak bisa mengkonsumsi queue sehingga worker lain yang akan mengkonsumsi.

**Message Acknowledgment**

Memproses task membutuhkan waktu. Apa yang terjadi jika pada saat worker sedang memproses pesan/task tiba-tiba worker mati (komputer crash)? Pesan/task akan hilang. Akan tetapi kita tidak menginginkan adanya pesan/task yang hilang. Jika worker mati maka ada worker lain yang akan menggantikannya. Untuk memastikan tidak ada pesan yang hilang maka RabbitMQ menggunakan **acknowledgment(ack)**. ACK akan dikirim dari consumer ke server RabbitMQ jika consumer telah selesai memproses task tersebut. Setelah menerima ack maka server RabitMQ akan menghapus pesan. Jika worker/consumer mati (koneksi tertutup/hilang, channel tertutup) tanpa mengirimkan ack maka server rabbitMQ akan menganggap bahwa worker mati dan melakukan enqueue ulang pesan.

**Message Durability**

Bagaimana jika server RabbitMQ yang crash? Tanpa adanya setup yang sesuai maka pesan dan antrian akan hilang jika server mati. Untuk memastikan bahwa pesan dan antrian tidak hilang maka perlu disetting agar pesan dan queue sebagai *durable*.

* Membuat queue/antrian durable

**boolean durable = true;**

channel.queueDeclare("hello", **durable**, false, false, null);

* Membuat pesan durable

channel.basicPublish("", "task\_queue",

MessageProperties.**PERSISTENT\_TEXT\_PLAIN**,

message.getBytes());

**Running**

1. Compile Newtask.java pada komputer A

javac -cp.:commons-io-2.5.jar:slf4j-api-1.7.21.jar:slf4j-simple-1.7.21.jar:amqp-client-4.0.0.jar NewTask.java

1. Compile Worker.java pada komputer B

javac -cp.:commons-io-2.5.jar:slf4j-api-1.7.21.jar:slf4j-simple-1.7.21.jar:amqp-client-4.0.0.jar Worker.java

1. Compile Worker.java pada komputer C

javac -cp.:commons-io-2.5.jar:slf4j-api-1.7.21.jar:slf4j-simple-1.7.21.jar:amqp-client-4.0.0.jar Worker.java

1. Jalankan Worker pada komputer C

java -cp.:commons-io-2.5.jar:slf4j-api-1.7.21.jar:slf4j-simple-1.7.21.jar:amqp-client-4.0.0.jar Worker

1. Jalankan Worker pada komputer B

java -cp.:commons-io-2.5.jar:slf4j-api-1.7.21.jar:slf4j-simple-1.7.21.jar:amqp-client-4.0.0.jar Worker

1. Jalankan NewTask pada komputer A

java -cp.:commons-io-2.5.jar:slf4j-api-1.7.21.jar:slf4j-simple-1.7.21.jar:amqp-client-4.0.0.jar NewTask

**Tugas**

1. Compile tutorial diatas dan pastikan semua berjalan!
2. Buatlah Worker1.java dan NewTask1.java. NewTask1.java akan mengirimkan pesan pada queue bernama “task\_queue” dan Worker1.java akan mengkonsumsi pesan dari “task\_queue”. Jika terdapat pesan berisi “sistem\_terdistribusi” maka Worker1.java akan mengirimkan pesan “sistem ok” ke NewTask1.java menggunakan queueu “result\_queue”