

Workshop 03

5CS022 Distributed System and Cloud Programming

Assessed Task

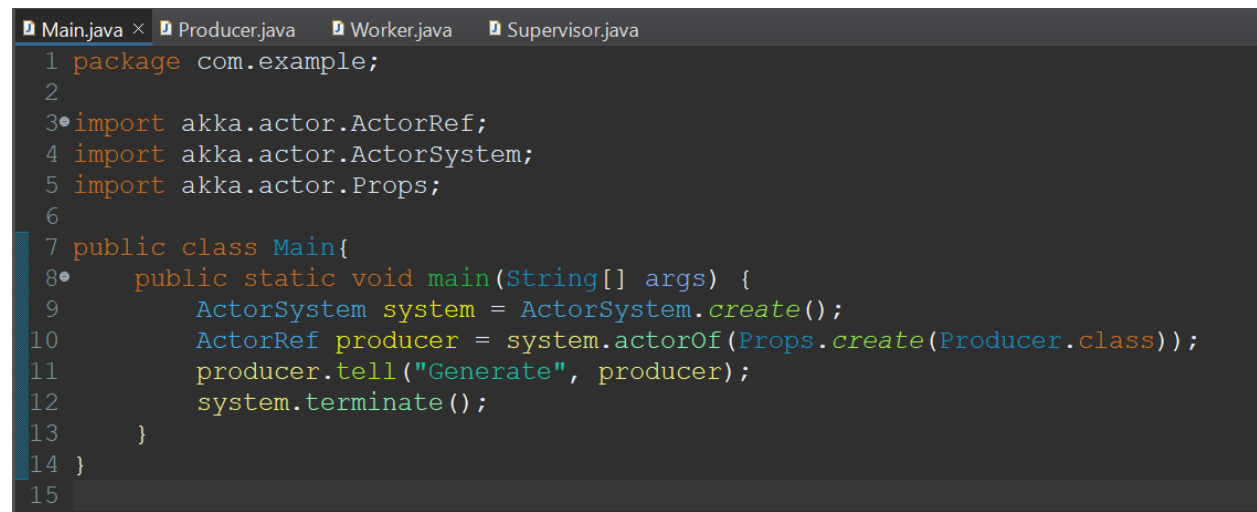
Create 3 Akka Actor classes called "Producer", "Supervisor" and "Worker". The "Producer" will generate 1000 random long integer numbers between 10000 and 100000. The "Producer" will send each number as a message to the "Supervisor".

At start-up, the Supervisor will create 10 "Worker" Actors. When the "Supervisor" receives a number from the "Producer", it will use the API `forward()` to send that message to one of the "Worker" actors, in a round-robin fashion.

The "Worker" actor will find if the number in the message is a prime number. If it is a prime number, it will then send a string/text message to the "Producer", saying that "The number XXX is a prime number." And the Producer will print out the message on the standard output.

When the 1000 numbers have been produced and checked, the "Producer" actor will end the Actor system.

Main.java:

The image shows a code editor with four tabs: Main.java, Producer.java, Worker.java, and Supervisor.java. The Main.java tab is active, displaying the following Java code:

```
1 package com.example;
2
3 import akka.actor.ActorRef;
4 import akka.actor.ActorSystem;
5 import akka.actor.Props;
6
7 public class Main{
8     public static void main(String[] args) {
9         ActorSystem system = ActorSystem.create();
10        ActorRef producer = system.actorOf(Props.create(Producer.class));
11        producer.tell("Generate", producer);
12        system.terminate();
13    }
14 }
15
```

Producer.java:

```
1 package com.example;
2
3 import akka.actor.AbstractActor;
4 import akka.actor.ActorRef;
5 import akka.actor.Props;
6 import java.util.Random;
7
8 public class Producer extends AbstractActor{
9
10     @Override
11     public Receive createReceive() {
12         return receiveBuilder()
13             .matchEquals("Generate", this::generatingRandom)
14             .match(String.class, this::showingPrimeNums)
15             .build();
16     }
17     void generatingRandom(String msg) {
18         ActorRef supervisor = getContext().actorOf(Props.create(Supervisor.class));
19         Random rand = new Random();
20         for(int i = 0; i<1000;i++) {
21             long num = 10000 +rand.nextInt(100000-10000+1);
22             supervisor.tell(num, getSelf());
23         }
24     }
25     void showingPrimeNums(String message) {
26         System.out.println(message);
27     }
28 }
```

Worker.java:

```
1 package com.example;
2
3 import akka.actor.AbstractActor;
4 import akka.actor.ActorRef;
5
6 public class Worker extends AbstractActor{
7     @Override
8     public Receive createReceive() {
9         return receiveBuilder()
10             .match(Long.class, this::primeVerification)
11             .build();
12     }
13     void primeVerification(Long num) {
14         int n;
15         for(n=2; n<= num-1; n++) {
16             if(num%n == 0)
17                 break;
18         }
19         if(n == num) {
20             ActorRef prod = getSender();
21             prod.tell(num + "is a Prime Number", getSelf());
22         }
23     }
24 }
```

Supervisor.java:

```
1 package com.example;
2
3 import akka.actor.AbstractActor;
4 import akka.actor.ActorRef;
5 import akka.actor.Props;
6
7 public class Supervisor extends AbstractActor{
8     ActorRef[] workers = new ActorRef[10];
9     int ind = 0;
10
11     public void start() {
12         for(int i = 0; i<10;i++) {
13             workers[i] = getContext().actorOf(Props.create(Worker.class));
14         }
15     }
16     @Override
17     public Receive createReceive() {
18         return receiveBuilder()
19             .match(Long.class, this::sendWorker)
20             .build();
21     }
22     void sendWorker(Long num) {
23         workers[ind].forward(num, getContext());
24         ind = (ind + 1)% workers.length;
25     }
26 }
```