

UNIVERSITY PARTNER



DISTRIBUTE AND CLOUD SYSTEMS PROGRAMMING (5CS022)


WEEK 4 WORKSHOP


| | |
|---------------------|-------------------------|
| Student Id | : 2065697 |
| Student Name | : Dhiraj Kumar Sah Kanu |
| Group | : L5CG12 |
| Submitted on | : April 17, 2022 |

Part 1 – Minimal sample program

1. Download the sample project


Go to the Canvas topic for 5CS022 Distributed and Cloud Systems Programming, and download the zip file “akka-hello(Eclipse).zip” if you are using Eclipse.


 5CS022 - Distributed and Cloud Systems Programming
Spring 2022


 **Week 4 Materials**
Sumanta Silwal • Mar 15


Dear Students,
Please go through the slides and workshop file attached herewith.


Thank you


 5CS022 Lecture 4 Akka.pdf
PDF


 5CS022 Week 4 Akka Works...
PDF


 akka-hello(Eclipse).zip
Compressed Archive

 Class comments






 akka-hello(Eclipse).zip

 Open with ZIP Extractor

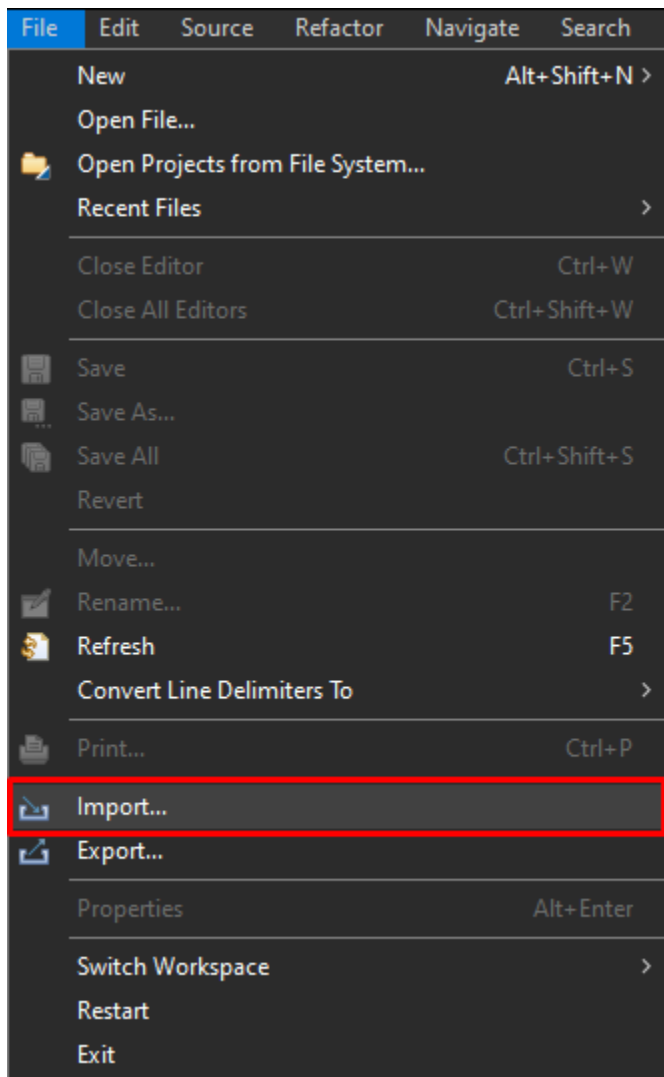
akka-hello(Eclipse).zip 1 item

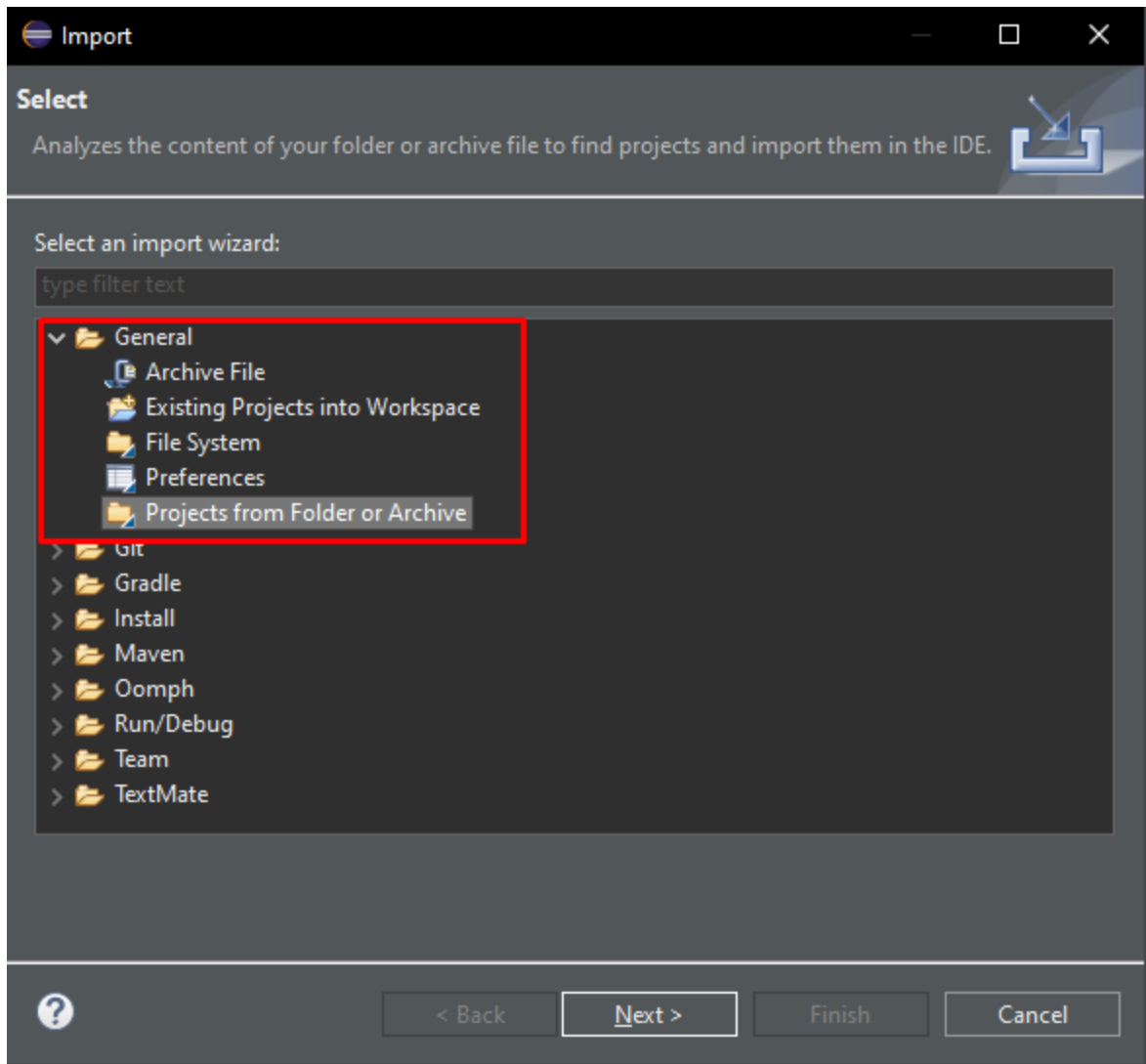
| Name | Last modified | File size |
|------------|---------------|-----------|
| akka-hello | - | - |

 akka-hello(Eclipse)...zip ^

2. Import the project into Eclipse/IntelliJ

Use the steps in the previous workshop to import the "akka-hello" project into your chosen IDE.





3. Build and run the program

Make sure that the minimal Akka program builds and runs correctly. If it does, you should see output similar to:

```
> Task :Main.main()
```

[2021-03-02 10:33:02,796] [INFO] [akka.event.slf4j.Slf4jLogger] [default-akka.actor.default-dispatcher-5]
[] - Slf4jLogger started

Press ENTER to end program.

[2021-03-02 10:33:03,008] [INFO] [akka.actor.ActorSystemImpl] [default-akka.actor.default-dispatcher-6]
[ActorSystemImpl(akka://default)] - Creating Actor A

[2021-03-02 10:33:03,022] [INFO] [com.example.ActorA] [default-akka.actor.default-dispatcher-5] [akka://default/user/\$a] - Actor A received Message A : 'Hello!' from
'Actor[akka://default/user/\$a#508038483]'

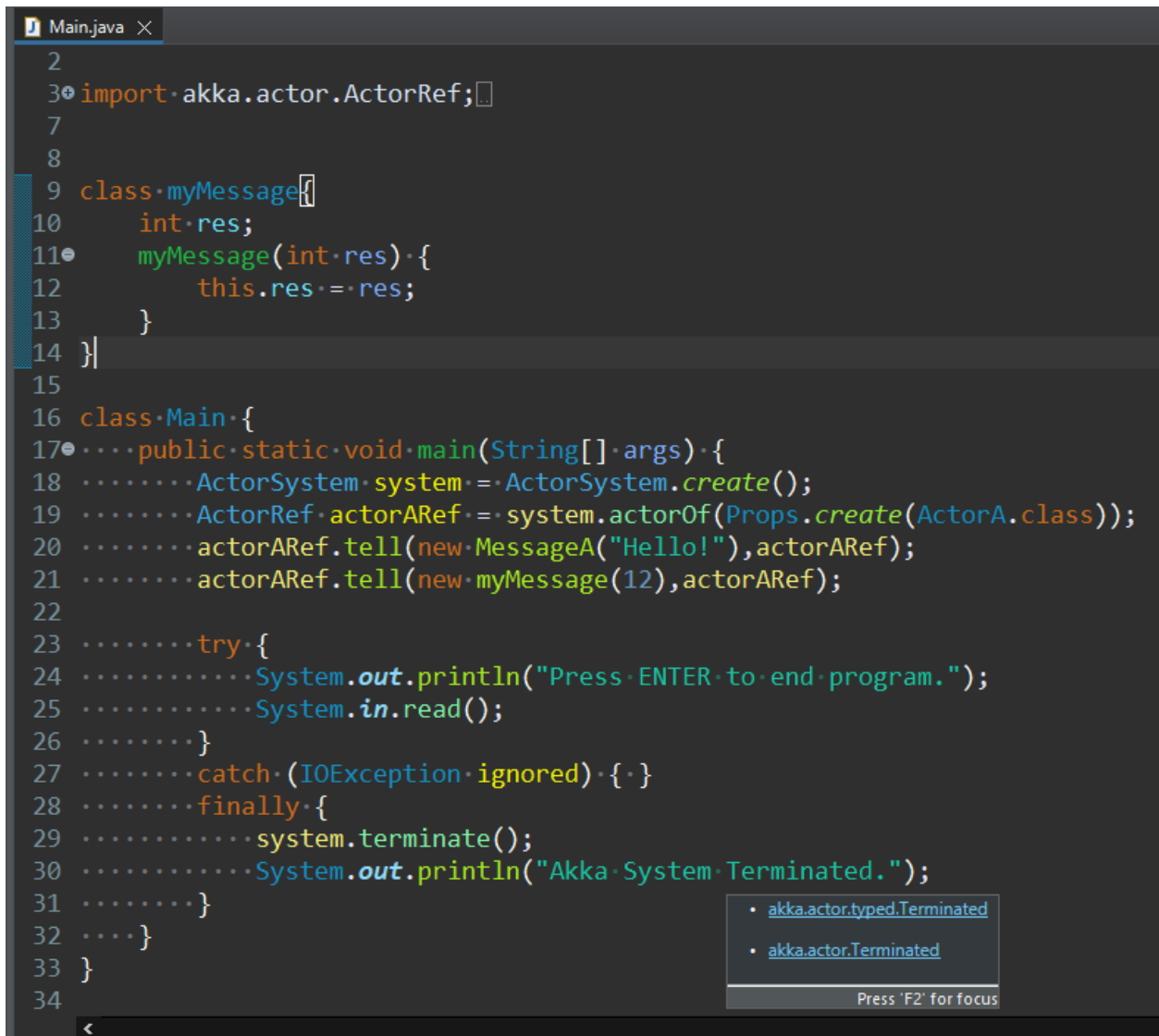
This minimal program will form the basis of all of the programs that you will work with for this workshop.

```
Main.java X
1 package com.example;
2
3 import akka.actor.ActorRef;
4
5
6
7
8 class Main {
9
10 ... public static void main(String[] args) {
11
12 ... ActorSystem system = ActorSystem.create();
13 ... ActorRef actorARef = system.actorOf(Props.create(ActorA.class));
14 ... actorARef.tell(new MessageA("Hello!"), actorARef);
15
16 ... try {
17 ... System.out.println("Press ENTER to end program.");
18 ... System.in.read();
19 ... }
20 ... catch (IOException ignored) { }
21 ... finally {
22 ... system.terminate();
23 ... System.out.println("Akka System Terminated.");
24 ... }
25 ... }
26
27 }
28
```

```
Problems Javadoc Declaration Console X
Main [Java Application] C:\Program Files\Java\jdk-18\bin\javaw.exe (Apr. 17, 2022, 9:45:53 p.m.) [pid: 2424]
[2022-04-17 21:45:54,366] [INFO] [akka.event.slf4j.Slf4jLogger] [default-akka.actor.default-dispatcher-4] [] - Slf4jLogger started
Press ENTER to end program.
Actor A received Message A : Hello! from Actor[akka://default/user/$a#-427109581]
```

Part 2

1. The Actor class ActorA in the sample Akka program currently responds to only one message object –MessageA. Modify the createReceive() method so that it will also respond to any other message and print out the message on the standard output.



```
2
3 import akka.actor.ActorRef;
4
5
6
7
8
9 class myMessage {
10     int res;
11     myMessage(int res) {
12         this.res = res;
13     }
14 }
15
16 class Main {
17     public static void main(String[] args) {
18         ActorSystem system = ActorSystem.create();
19         ActorRef actorARef = system.actorOf(Props.create(ActorA.class));
20         actorARef.tell(new MessageA("Hello!"), actorARef);
21         actorARef.tell(new myMessage(12), actorARef);
22
23         try {
24             System.out.println("Press ENTER to end program.");
25             System.in.read();
26         }
27         catch (IOException ignored) {}
28         finally {
29             system.terminate();
30             System.out.println("Akka System Terminated.");
31         }
32     }
33 }
34
```

- [akka.actor.typed.Terminated](#)
- [akka.actor.Terminated](#)

Press 'F2' for focus

In the above screenshot we can see the actor system is created. Similarly, object is then created for an actor system. The Tell function was used to convey messages of the class myMessage and Message A. Error handling has been employed in this code.

```

1
2 import akka.actor.AbstractActor;
3
4 public class ActorA extends AbstractActor {
5     LoggingAdapter log = Logging.getLogger(getContext().system(), this);
6     @Override
7     public Receive createReceive() {
8         return receiveBuilder()
9             .matchAny(m -> log.info("Actor A received message-{}", m))
10            .build();
11        }
12    }
13
14    private void onMessageA(MessageA msg) {
15        System.out.println("Actor A received Message A: " + msg.text + " from " + getSender());
16    }
17 }

```

The akka.actor.AbstractActor class has been expanded to include an actor. After the actor receives a message, the createReceive() function is called. A createReceive() function assists in determining the kind of message and responding appropriately. The matchAny() function is used to match any message. The LoggingAdapter is used to trigger the matchAny method ().

```

1
2
3 public class MessageA {
4     public final String text;
5
6     public MessageA(String text) {
7         this.text = text;
8     }
9 }
10
11

```

A message is stored twice: once in the constructor and once in the class.

```
Main (0) [Java Application] C:\Program Files\Java\jdk-18\bin\javaw.exe [Apr. 17, 2022, 7:57:37 p.m.] [pid: 876]
[2022-04-17 19:57:37,995] [INFO] [akka.event.slf4j.Slf4jLogger] [default-akka.actor.default-dispatcher-4] [] - Slf4jLogger started
Press ENTER to end program.
[2022-04-17 19:57:38,084] [INFO] [first.ActorA] [default-akka.actor.default-dispatcher-4] [akka://default/user/$a] - Actor A received message first.MessageA@1dddcf4a
[2022-04-17 19:57:38,085] [INFO] [first.ActorA] [default-akka.actor.default-dispatcher-4] [akka://default/user/$a] - Actor A received message first.myMessage@98d60ff
```

The resultant output is shown in the above screenshot.

2. Every message in Akka is a Java object. However, not all messages need to have a custom Java class created for them. Convert the sample program so that it can respond to messages that contain the primitive data types such as byte, short, int, long, float, double, boolean, and char, without having to create custom Java classes.

```
2
3 import akka.actor.ActorRef;
7
8 class myMessage{
9     int res;
10 myMessage(int res){
11     this.res = res;
12 }
13 }
14
15 class Main{
16 public static void main(String[] args){
17     ActorSystem system = ActorSystem.create();
18     ActorRef actorARef = system.actorOf(Props.create(ActorA.class));
19     actorARef.tell("Hello!", actorARef);
20     actorARef.tell(12, actorARef);
21     actorARef.tell(13, actorARef);
22     actorARef.tell(999, actorARef);
23     actorARef.tell(20.0, actorARef);
24     actorARef.tell(233.322, actorARef);
25     actorARef.tell(123456789, actorARef);
26     actorARef.tell(true, actorARef);
27
28     try{
29         System.out.println("Press ENTER to end program.");
30         System.in.read();
31     }
32     catch(IOException ignored){}
33     finally{
34         system.terminate();
35         System.out.println("Akka System Terminated.");
36     }
37 }
38 }
39
```

At the start of this main program, an actor system is constructed. The actor is then given an item. To transmit a message, the tell method is utilized. In this software, try catch was employed. Individual classes have not been created in this. A built-in function was employed.


```

1
2
3 import akka.actor.AbstractActor;
4
5
6
7
8
9 public class ActorA extends AbstractActor {
10
11     LoggingAdapter log = Logging.getLogger(getContext().system(), this);
12
13     ... @Override
14     ... public Receive createReceive() {
15         ... return receiveBuilder()
16         ... .matchAny(m->log.info("Actor A received message {}", m))
17         ... .build();
18     }
19 }

```

After receiving a message, matchAny matches the messages. Various sorts of data are matched. The LoggingAdapter is used to execute the matchAny method.

```

1
2
3 public class MessageA {
4     ... public final String text;
5
6     ... public MessageA(String text) {
7         ... this.text = text;
8     }
9 }
10

```

```

Main (1) [Java Application] C:\Program Files\Java\jdk-18\bin\javaw.exe (Apr. 17, 2022, 8:02:24 p.m.) [pid: 3588]
[2022-04-17 20:02:25,568] [INFO] [second.ActorA] [default-akka.actor.default-dispatcher-4] [akka://default/user/$a] - Actor A received message Hello!
[2022-04-17 20:02:25,569] [INFO] [second.ActorA] [default-akka.actor.default-dispatcher-4] [akka://default/user/$a] - Actor A received message 12
[2022-04-17 20:02:25,569] [INFO] [second.ActorA] [default-akka.actor.default-dispatcher-4] [akka://default/user/$a] - Actor A received message 13
[2022-04-17 20:02:25,569] [INFO] [second.ActorA] [default-akka.actor.default-dispatcher-4] [akka://default/user/$a] - Actor A received message 999
[2022-04-17 20:02:25,569] [INFO] [second.ActorA] [default-akka.actor.default-dispatcher-4] [akka://default/user/$a] - Actor A received message 20.0
[2022-04-17 20:02:25,569] [INFO] [second.ActorA] [default-akka.actor.default-dispatcher-4] [akka://default/user/$a] - Actor A received message 233.322
[2022-04-17 20:02:25,569] [INFO] [second.ActorA] [default-akka.actor.default-dispatcher-4] [akka://default/user/$a] - Actor A received message 123456789
[2022-04-17 20:02:25,569] [INFO] [second.ActorA] [default-akka.actor.default-dispatcher-4] [akka://default/user/$a] - Actor A received message true

```

3. In standard multithreading programs (for example Pthread programs), shared-resource contention (e.g. global variables) can be an issue and requires the use of mutexes and critical sections. Demonstrate that with Akka Actor, this is not an issue, by creating a "Counter" Actor class to keep track of a global counter, and lots of instances (20) of "ActorA" object to send messages to "Counter" to increment the global counter.

```
1 |
2 |
3 | import akka.actor.ActorRef;
4 |
5 |
6 |
7 |
8 |
9 | class Main {
10 |
11 | ... public static void main(String[] args) {
12 |
13 | ... ActorSystem system = ActorSystem.create();
14 | ... ActorRef counterARef = system.actorOf(Props.create(Counter.class));
15 | ...
16 | ...
17 | ... for(int i = 0; i < 10; i++) {
18 | ...
19 | ... ActorRef actorRef = system.actorOf(Props.create(ActorA.class));
20 | ... actorRef.tell(new MessageA("Hello"), counterARef);
21 | ...
22 | ... }
23 | ...
24 |
25 | ... try {
26 | ... System.out.println("Press ENTER to end program.");
27 | ... System.in.read();
28 | ... }
29 | ... catch (IOException ignored) { }
30 | ... finally {
31 | ... system.terminate();
32 | ... System.out.println("Akka System Terminated.");
33 | ... }
34 | ... }
35 |
36 | }
```

```

1
2
3 import akka.actor.AbstractActor;
8
9 public class ActorA extends AbstractActor {
10
11     LoggingAdapter log = Logging.getLogger(getContext().system(), this);
12
13     @Override
14     public Receive receive() {
15         return receiveBuilder()
16             .match(MessageA.class, this::onMessageA)
17             .build();
18     }
19
20     private void onMessageA(MessageA msg) {
21         ActorRef counter = getSender();
22         counter.tell(new MessageA("Message"), getSelf());
23     }
24 }
25
26
27 }

```

LoggingAdapter third.ActorA.log

Press 'F2' for focus

```

1 |
2
3 import akka.actor.AbstractActor;
4
5
6 public class Counter extends AbstractActor{
7
8     private int count;
9
10
11     Counter(){
12         count = 0;
13     }
14
15     @Override
16     public Receive createReceive() {
17         .....return receiveBuilder()
18             .....match(MessageA.class, this::onMessageA)
19             .....build();
20         ....}
21
22     private void onMessageA(MessageA msg) {
23         count++;
24         System.out.printf("Counter: %d\n", count);
25     }
26 }
27
28 }

```

```

1 |
2
3 public class MessageA {
4     ....public final String text;
5
6     ....public MessageA(String text) {
7         .....this.text = text;
8     }
9 }
10

```

```
Main (2) [Java Application] C:\Program Files\Java\jdk-18\bin\javaw.exe (Apr. 17, 2022, 8:04:01 p.m.) [pid: 2200]
[2022-04-17 20:04:02,254] [INFO] [akka.event.slf4j.Slf4jLogger] [default-akka.actor.default-dispatcher-5] [] - Slf4jLogger started
Press ENTER to end program.
Counter: 1
Counter: 2
Counter: 3
Counter: 4
Counter: 5
Counter: 6
Counter: 7
Counter: 8
Counter: 9
Counter: 10
```

4. Create 2 Akka Actor classes "ActorA" and "ActorB" to demonstrate the Akka API `setReceiveTimeout()`. ActorA will generate a random integer number from 1 to 5 in a loop for 100 times, and send this integer as a message to ActorB, which would then call `Thread.sleep()` that many seconds. Set the receive timeout to 2 seconds, and when the timeout triggers, send ActorB a `stop ()` message, and then create a new instance of ActorB to process the next number.

Assessed Task

1. Create 3 Akka Actor classes call "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 "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 forward that message to one of the "Worker" actors, in a round-robin fashion.

The "Worker" actor will determine 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 terminate the Actor system.

Export your project as a Zip file (either IntelliJ or Eclipse) and submit this program as "workshoptask3.zip" as part of your final portfolio submission. You can also upload it to the formative submission point for formative feedback.