Module	Assessment Type
Distributed and Cloud Systems Programming	Individual Report

Workshop 7 I

Student Id : 2049867
Student Name : Roshan Parajuli
Section : L5CG3
Module Leader : Rupak Koirala
Lecturer /Tutor : Saroj Sharma
Submitted on : 2021-05-14

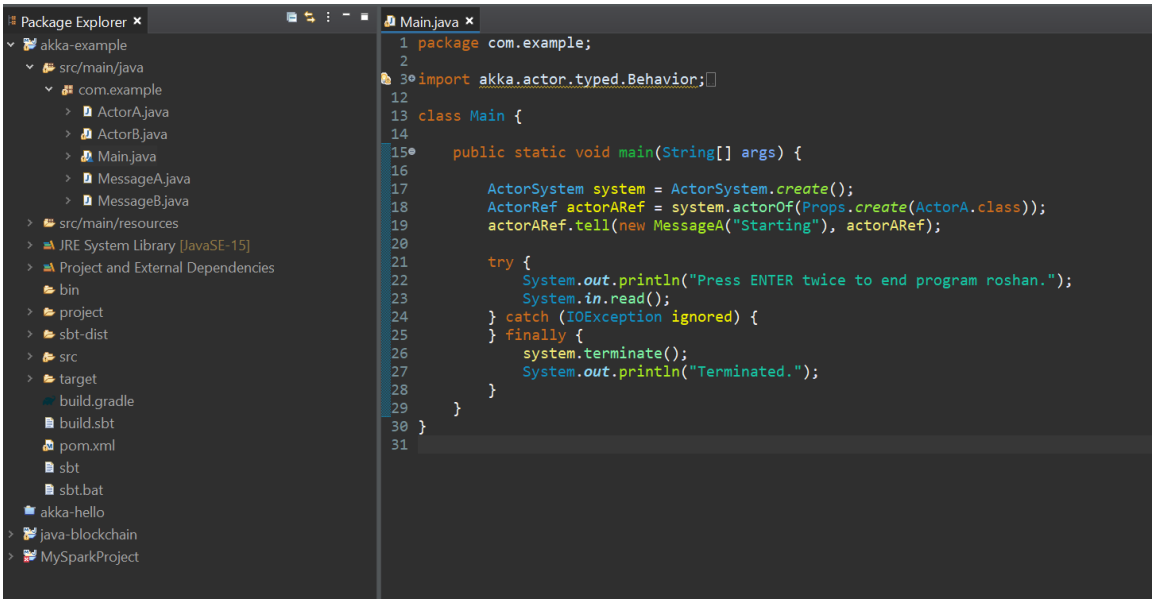
Introduction

Akka is a free and open-source toolkit and runtime simplifying the construction of concurrent and distributed applications on the JVM. It supports multiple programming models for concurrency. In this workshop, a simple concurrent program is imported and run in the VSCode IDE.

Workshop

Steps:

1. The sample project is sample project.
2. The project is imported into Eclipse.
3. The Main.java file was run as it contained the main function for the calling of actor A with the message to start the program.



4. First of all, the akka system is created using which the references for other classes are created.

```
ActorSystem system = ActorSystem.create();
ActorRef actorARef = system.actorOf(Props.create(ActorA.class));
```

5. Reference of Actor A is created.

```
ActorSystem system = ActorSystem.create();
ActorRef actorARef = system.actorOf(Props.create(ActorA.class));
actorARef.tell(new MessageA("Starting"), actorARef);
```

After the reference is created, tell method is used to send a new Message (namely Message A) to the actor A.

6. After actor B, receives the message sent from actor A, the value of 42 is returned as message B through it. Message A is mapped to the onMessageA function in the createReceive method in the actor.

```
private void onMessageA(MessageA msg) {
    System.out.println("Actor B received Message A : " + msg.text + " from " + getSender());
    for (int i = 0; i < 10; i++) {
        num++;
        System.out.println("Actor B doing work " + i);
    }
    getSender().tell(new MessageB(num), getSelf());
}
```

7. In a similar way, Actor A receives message B and replies another message B with the value of 999.
8. After actor B receives the message B of 999, it replies with a Message A of “GoodBye!”.

```
if (msg.number == 999) {
    getSender().tell(new MessageA("Goodbye!"), getSelf());

} else {
    getSender().tell(new MessageB(num), getSelf());
}
```

9. When ActorA receives the message A of “Goodbye!”, the whole akka system is terminated.

```
if (msg.text.equalsIgnoreCase("Goodbye!")) {
    getContext().getSystem().terminate();
} else {
    ActorRef actorBRef = getContext().getSystem().actorOf(Props.create(ActorB.class));
    actorBRef.tell(new MessageA("Hello!"), getSelf());
}
```

Changing the work done to add up the numbers

Since message B is an integer, it is sent along the actors rather than the random integer like the value 42 above. To make the actors do the work before responding to the message, the loop is shifted above all the tell methods.

```
private void onMessageA(MessageA msg) {
    System.out.println("Actor A received Message A : " + msg.text + " from " + getSender());
    for (int i = 0; i < 10; i++) {
        System.out.println("Actor A doing work " + i);
    }
    if (msg.text.equalsIgnoreCase("Goodbye!")) {
        getContext().getSystem().terminate();
    } else {
        ActorRef actorBRef = getContext().getSystem().actorOf(Props.create(ActorB.class));
        actorBRef.tell(new MessageA("Hello!"), getSelf());
    }
}
```

In the actor A, on receiving message B, the loop is executed and number is increased every

```
private void onMessageB(MessageB msg) {
    System.out.println("Actor A received Message B : " + msg.number + " from " + getSender());
    int numb = msg.number;

    for (int i = 0; i < 10; i++) {
        numb++;
        System.out.println("Actor A doing more work " + numb);
    }

    getSender().tell(new MessageB(numb), getSelf());

}
```

time the loop runs and this function returns the current count or the current value that the number holds to the sender (Previously, it was 42 and now is replaced by the value).
After making such changes in Actor A, Actor B was modified accordingly.

The following output can be observed: (due to some errors on eclipse, inellij was used to compile and run the final version of the code)

```
Actor A doing more work 995
Actor A doing more work 996
Actor A doing more work 997
Actor A doing more work 998
Actor A doing more work 999
Actor A doing more work 1000
Actor B received Message B : 1000 from Actor[akka://default/user/$a#-760993399]
Actor B doing more work 1001
Actor B doing more work 1002
Actor B doing more work 1003
Actor B doing more work 1004
Actor B doing more work 1005
Actor B doing more work 1006
Actor B doing more work 1007
Actor B doing more work 1008
Actor B doing more work 1009
Actor B doing more work 1010
Actor A received Message A : Goodbye! from Actor[akka://default/user/$b#-925338883]
Actor A doing work 0
Actor A doing work 1
Actor A doing work 2
Actor A doing work 3
Actor A doing work 4
Actor A doing work 5
Actor A doing work 6
Actor A doing work 7
Actor A doing work 8
Actor A doing work 9


Terminated
```

Conclusion

In this workshop, simple akka system is observed where message was passed to and fro between all the actors.