**University of Wolverhampton**

**School of Mathematics and Computer Science**

**5CS022 Distribute and Cloud Systems Programming**

**Workshop 4 The Akka Framework Part 2**

**Overview**

This workshop continues the exploration of the Akka Framework from the previous workshop.

1. In the sample Akka program, the Actor class ActorA currently responds to only one message object-MessageA. Modify the createReceive() method to respond to any other message and print the message to standard output.

Text

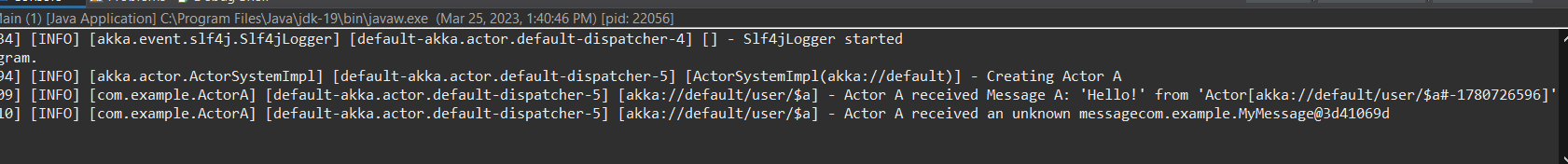
Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated



1. In Akka, every message is a Java object. However, not all messages require the development of a custom Java class. Convert the sample program to respond to messages containing primitive datatypes like byte, short, int, long, float, double, boolean, and char without the need for custom Java classes.Text

   Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

A picture containing text

Description automatically generated

3. In traditional multithreading programs such as Pthread programs, there can be issues with shared-resource contention, such as when multiple threads try to access a global variable simultaneously. This requires the use of mutexes and critical sections to ensure that only one thread can access the resource at a time. However, in Akka Actor, this is not an issue. To demonstrate this, we can create a "Counter" Actor class that keeps track of a global counter and create 20 instances of "ActorA" objects that send messages to the "Counter" actor to increment the global counter.

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Graphical user interface, text

Description automatically generated

4. Create two Akka Actor classes named "ActorA" and "ActorB" to illustrate the usage of Akka's API method "setReceiveTimeout()". The ActorA will repeatedly generate a random integer between 1 to 5, 100 times in a loop, and then send it as a message to ActorB. ActorB will then sleep for the number of seconds equal to the received integer. Set the receive timeout to 2 seconds, and if the timeout occurs, send a "stop()" message to ActorB and then create a new instance of ActorB to process the next message.

A screenshot of a computer

Description automatically generated with medium confidence

Text

Description automatically generated

Text

Description automatically generated

A screenshot of a computer

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

5. Create three Akka Actor classes named "Producer", "Supervisor", and "Worker". The "Producer" Actor will generate 1000 random long integer numbers between 10000 and 100000, and send each number as a message to the "Supervisor". At startup, the "Supervisor" will create 10 "Worker" Actors. When the "Supervisor" receives a number from the "Producer", it will forward that message to one of the "Worker" Actors using the forward() API method, in a round-robin fashion. The "Worker" Actor will then determine whether the number in the message is a prime number or not. If it is a prime number, it will send a text message to the "Producer" saying "The number XXX is a prime number". The "Producer" will print out the message on the standard output. Once all 1000 numbers have been produced and checked, the "Producer" Actor will terminate the Actor system.

A screenshot of a computer

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated