Module Title: DISTRIBUTED AND CLOUD SYSTEMS PROGRAMMING

(5CS022)

Subject Title: Workshop-03

Student Name: Nayan Raj Khanal

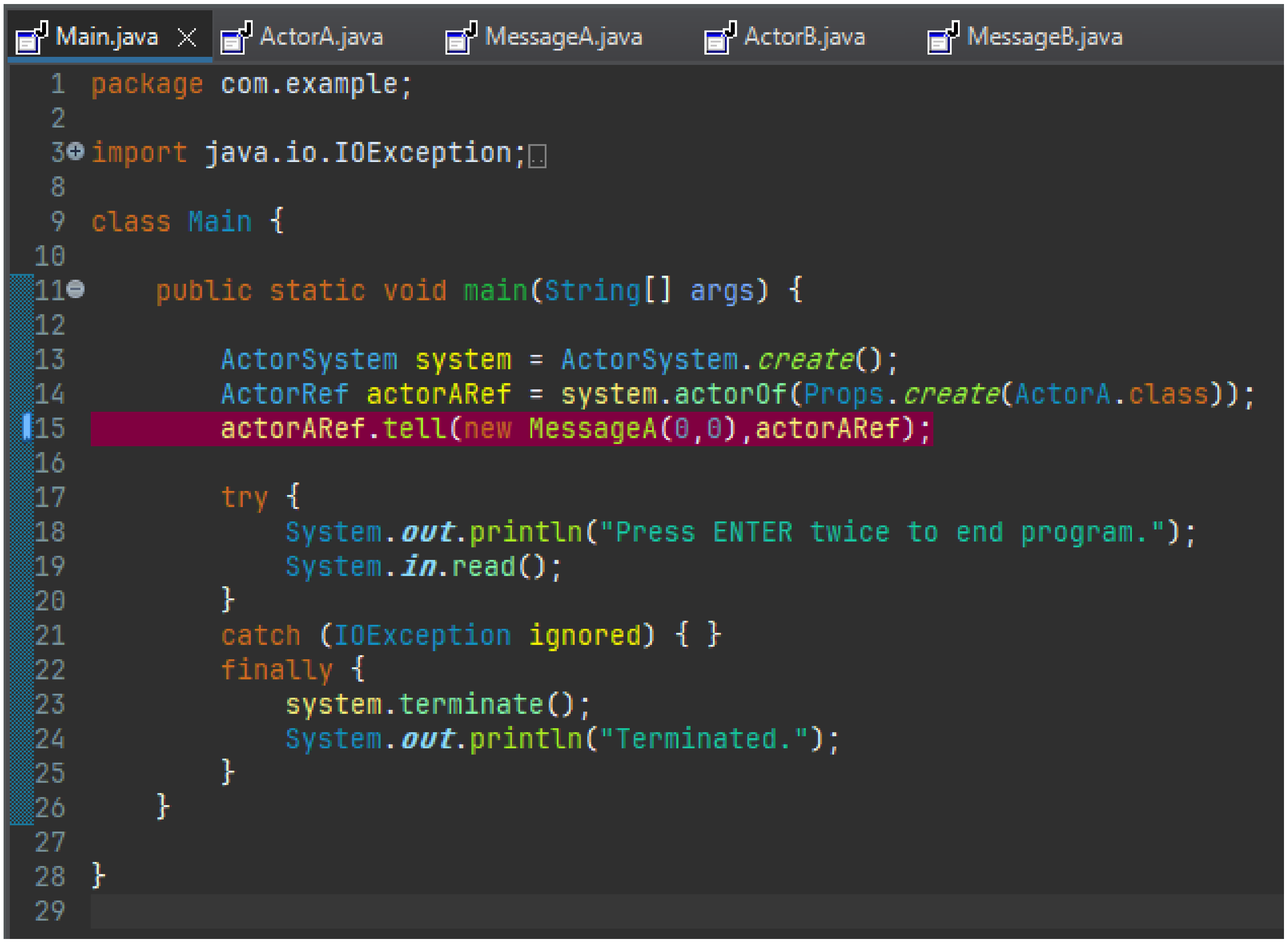
Student Code: 2227486

Instructor: Mr. Prabin Sapkota

Submitted on: 20.03.2023

1. Change the "work" done to add up numbers. Make the actors do the work before responding to their message senders. Send back the result of the work done to their senders.

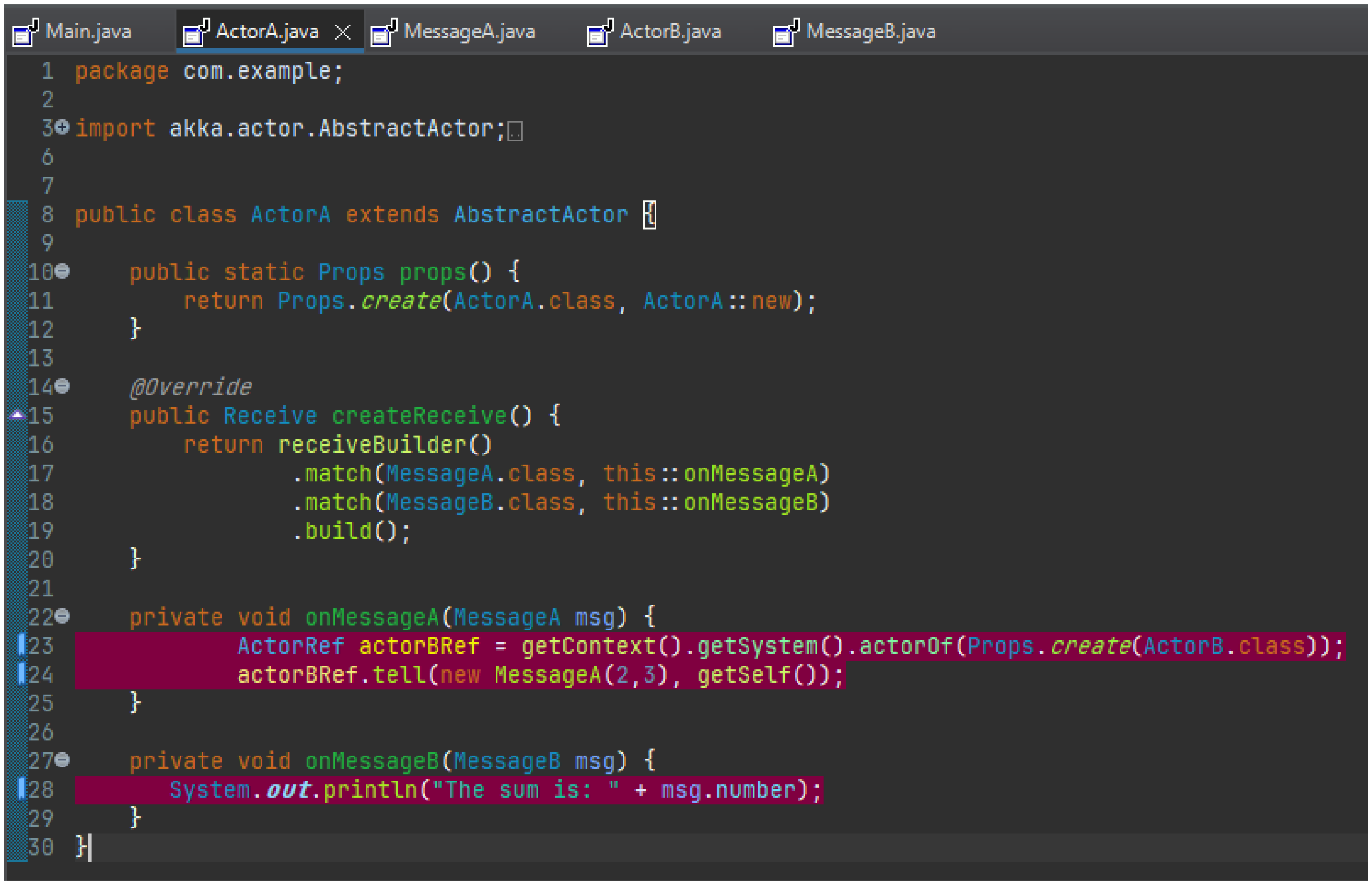
- Main:



```java
package com.example;

import java.io.IOException;

class Main {

    public static void main(String[] args) {

        ActorSystem system = ActorSystem.create();
        ActorRef actorARef = system.actorOf(Props.create(ActorA.class));
        actorARef.tell(new MessageA(0,0),actorARef);

        try {
            System.out.println("Press ENTER twice to end program.");
            System.in.read();
        }
        catch (IOException ignored) { }
        finally {
            system.terminate();
            System.out.println("Terminated.");
        }
    }

}
```

The message passed to the ActorA instance has been modified from a string message ("Starting") to an integer message (0, 0).

- ActorA:

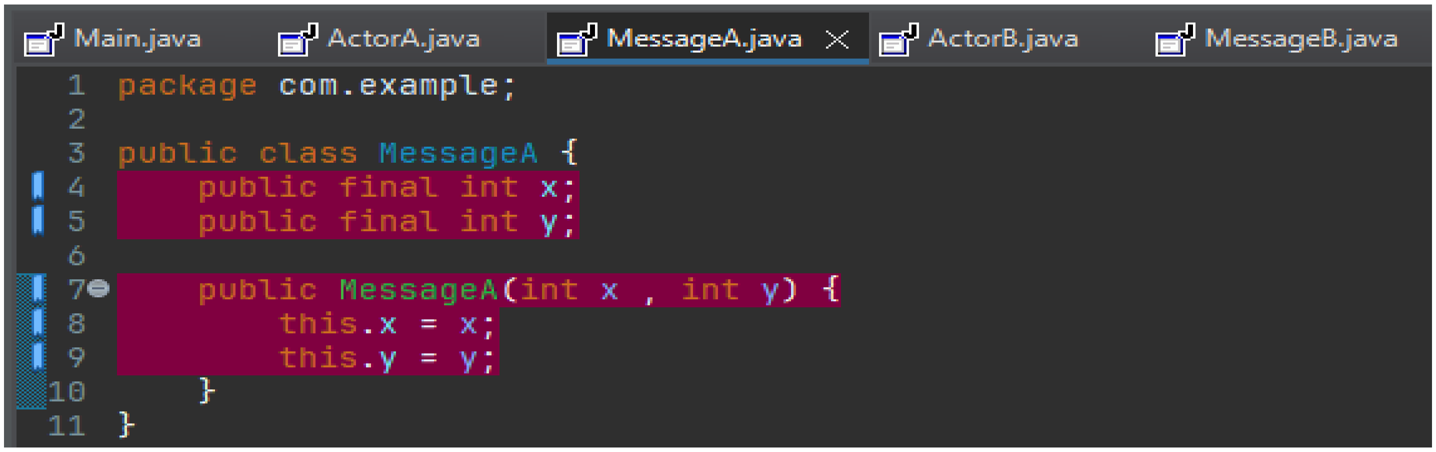


```java
package com.example;

import akka.actor.AbstractActor;


public class ActorA extends AbstractActor {

    public static Props props() {
        return Props.create(ActorA.class, ActorA::new);
    }

    @Override
    public Receive createReceive() {
        return receiveBuilder()
                .match(MessageA.class, this::onMessageA)
                .match(MessageB.class, this::onMessageB)
                .build();
    }

    private void onMessageA(MessageA msg) {
            ActorRef actorBRef = getContext().getSystem().actorOf(Props.create(ActorB.class));
            actorBRef.tell(new MessageA(2,3), getSelf());
    }

    private void onMessageB(MessageB msg) {
        System.out.println("The sum is: " + msg.number);
    }
}
```

In the onMessageA method, the logic for processing the received MessageA has been simplified, and a new ActorB instance is created and sent a new MessageA with two integer arguments, 2 and 3.

In the onMessageB method, the logic for processing the received MessageB has been modified to simply print the sum of the MessageB number value.

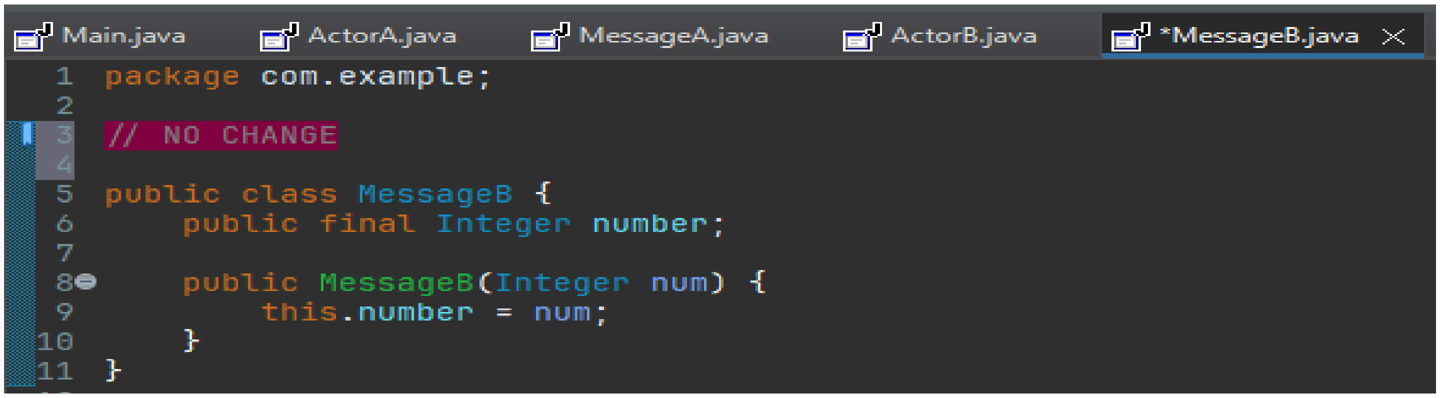
- MessageA:



```java
package com.example;

public class MessageA {
    public final int x;
    public final int y;

    public MessageA(int x , int y) {
        this.x = x;
        this.y = y;
    }
}
```

The MessageA class has been modified to have two integer fields, x and y, instead of a single String field, text. The constructor for MessageA now takes two integer arguments, x and y, instead of a single String argument.

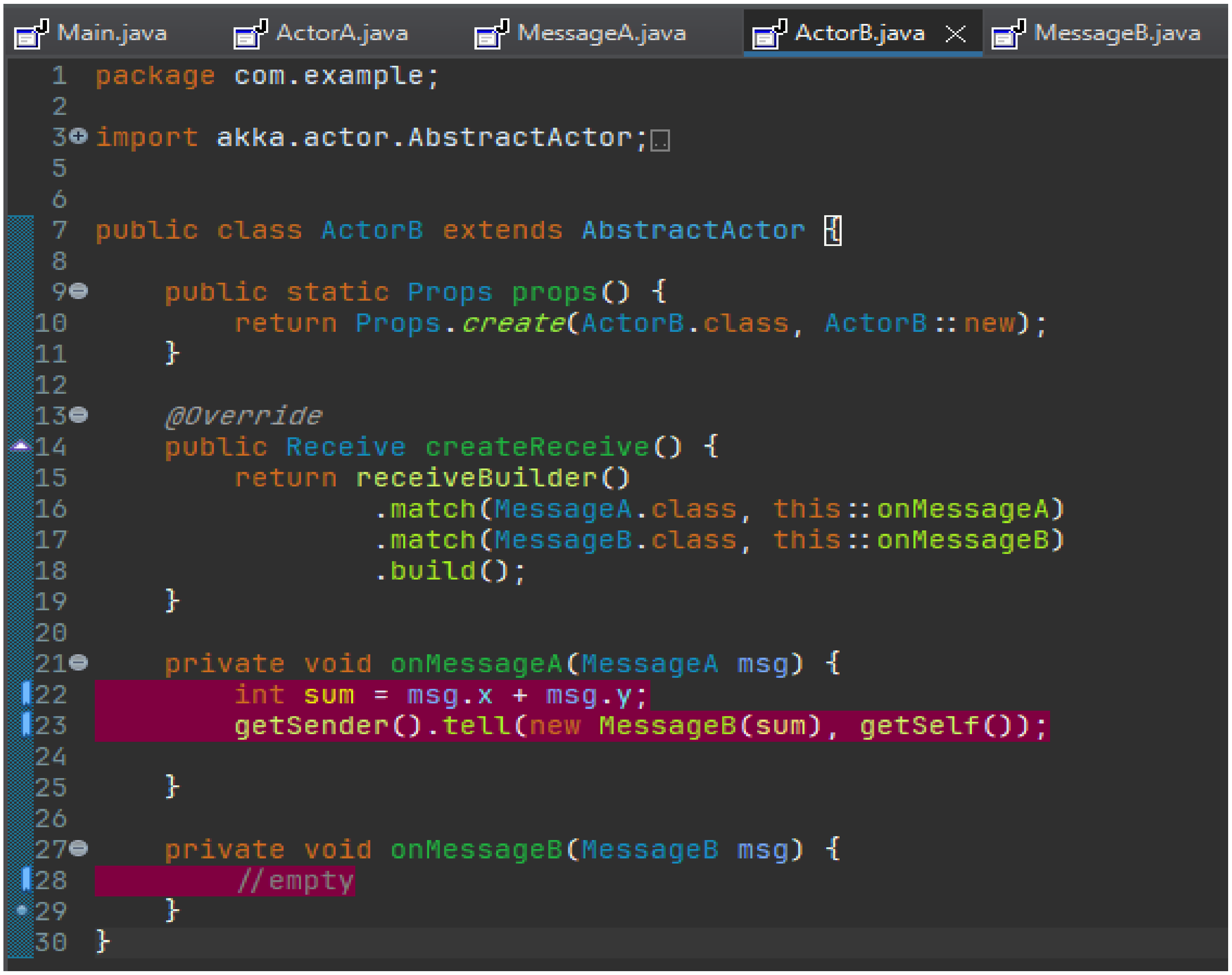
- MessageB:



```java
package com.example;

// NO CHANGE

public class MessageB {
    public final Integer number;

    public MessageB(Integer num) {
        this.number = num;
    }
}
```

Nothing was changed in this code.

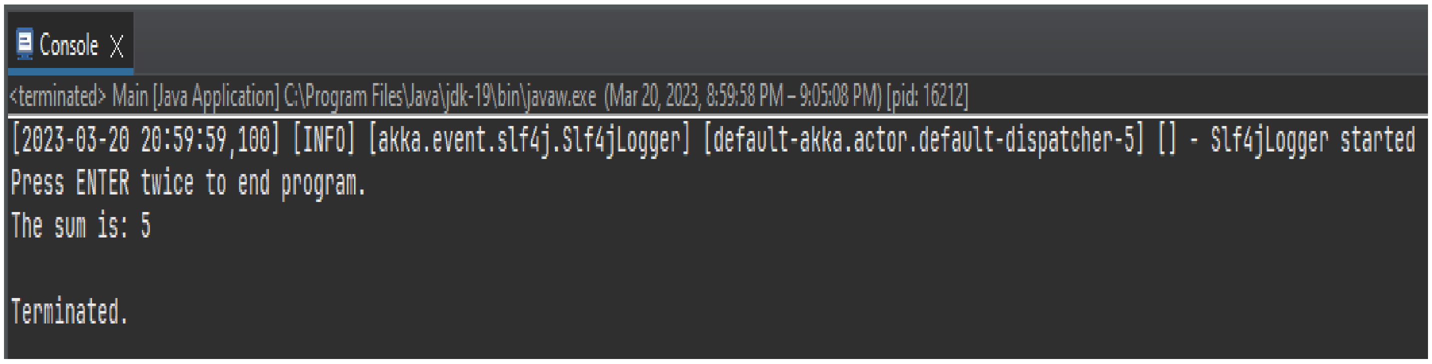
- ActorB:



```java
package com.example;

import akka.actor.AbstractActor;


public class ActorB extends AbstractActor {

    public static Props props() {
        return Props.create(ActorB.class, ActorB::new);
    }

    @Override
    public Receive createReceive() {
        return receiveBuilder()
                .match(MessageA.class, this::onMessageA)
                .match(MessageB.class, this::onMessageB)
                .build();
    }

    private void onMessageA(MessageA msg) {
        int sum = msg.x + msg.y;
        getSender().tell(new MessageB(sum), getSelf());

    }

    private void onMessageB(MessageB msg) {
        //empty
    }
}
```

ActorB now simply calculates the sum of MessageA's x and y values and sends a MessageB containing the result back to the sender (ActorA). onMessageB method is now empty because it is no longer necessary to check for a specific value in the MessageB.

- Output:

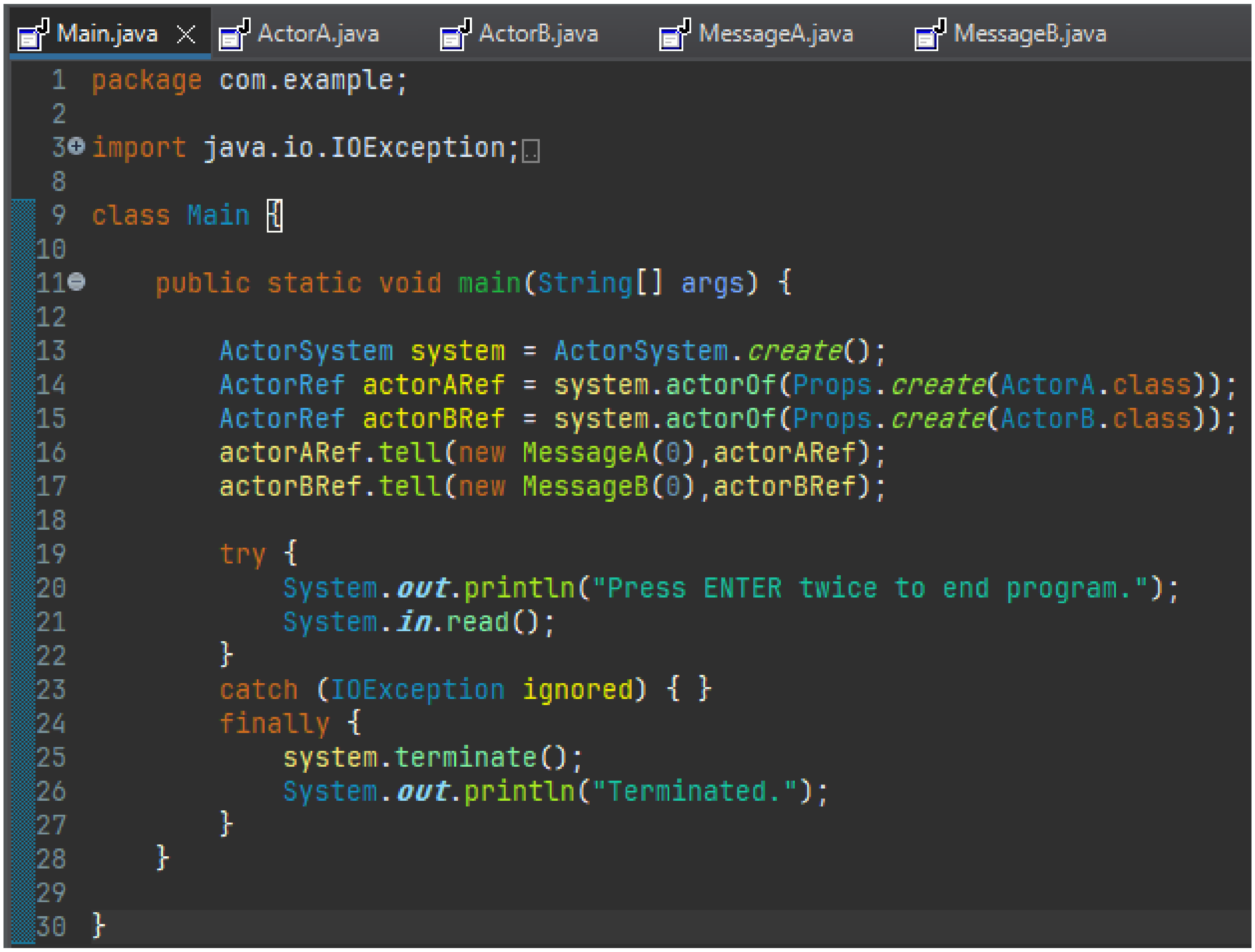


Console

```
<terminated> Main [Java Application] C:\Program Files\Java\jdk-19\bin\javaw.exe  (Mar 20, 2023, 8:59:58 PM – 9:05:08 PM) [pid: 16212]
[2023-03-20 20:59:59,100] [INFO] [akka.event.slf4j.Slf4jLogger] [default-akka.actor.default-dispatcher-5] [] - Slf4jLogger started
Press ENTER twice to end program.
The sum is: 5

Terminated.
```

2. Rewrite the "prime numbers" task from Week 1 MPI workshop to use Actors instead.

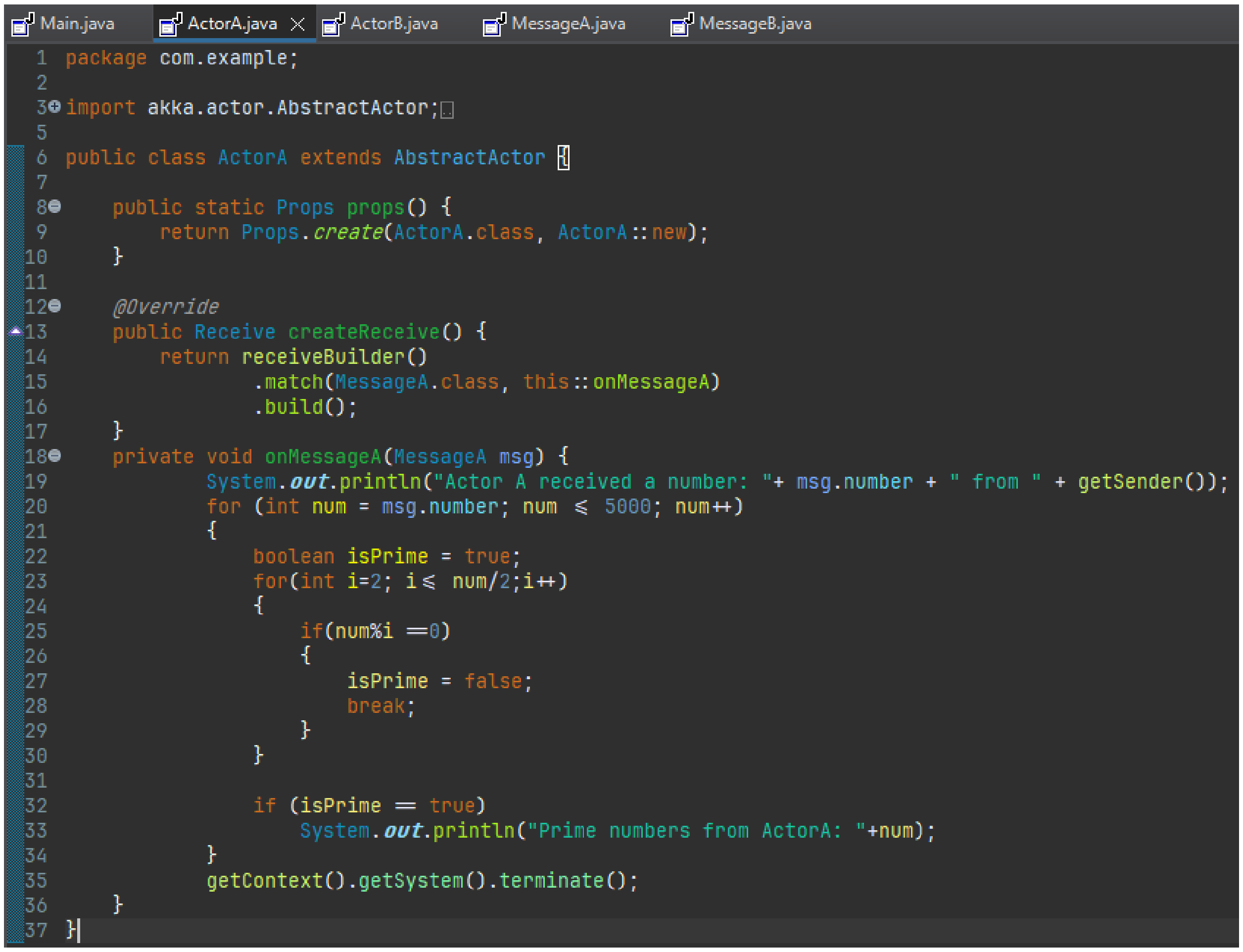
    - Main:



```java
package com.example;

import java.io.IOException;

class Main {

    public static void main(String[] args) {

        ActorSystem system = ActorSystem.create();
        ActorRef actorARef = system.actorOf(Props.create(ActorA.class));
        ActorRef actorBRef = system.actorOf(Props.create(ActorB.class));
        actorARef.tell(new MessageA(0),actorARef);
        actorBRef.tell(new MessageB(0),actorBRef);

        try {
            System.out.println("Press ENTER twice to end program.");
            System.in.read();
        }
        catch (IOException ignored) { }
        finally {
            system.terminate();
            System.out.println("Terminated.");
        }
    }

}
```

The code creates an ActorSystem, instantiates two actors ActorA and ActorB, and sends them each a message. It then waits for the user to press enter before terminating the ActorSystem.

- ActorA:

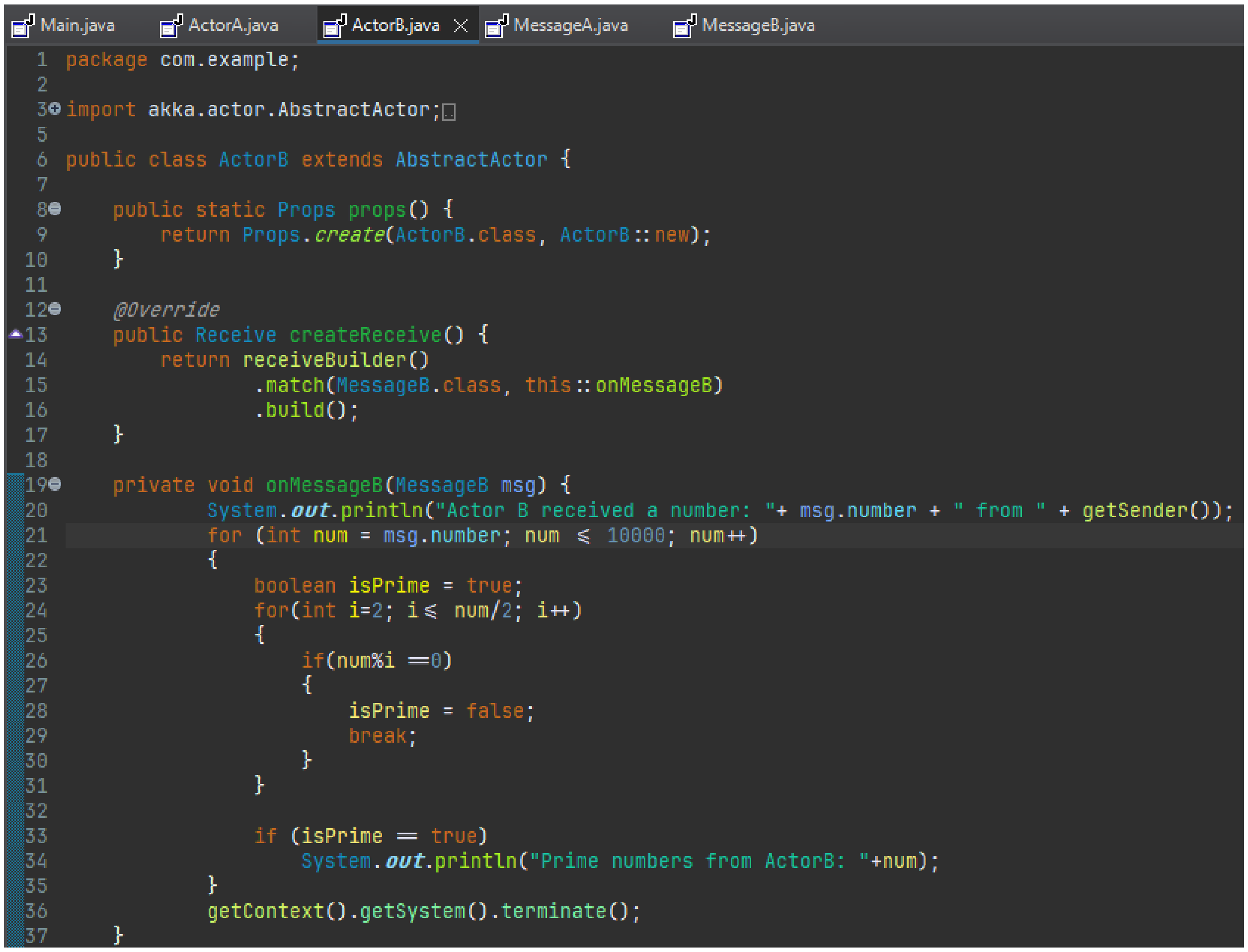


Main.java ActorA.java ActorB.java MessageA.java MessageB.java

```java
package com.example;

import akka.actor.AbstractActor;

public class ActorA extends AbstractActor {

    public static Props props() {
        return Props.create(ActorA.class, ActorA::new);
    }

    @Override
    public Receive createReceive() {
        return receiveBuilder()
                .match(MessageA.class, this::onMessageA)
                .build();
    }
    private void onMessageA(MessageA msg) {
            System.out.println("Actor A received a number: "+ msg.number + " from " + getSender());
            for (int num = msg.number; num <= 5000; num++)
            {
                boolean isPrime = true;
                for(int i=2; i<= num/2;i++)
                {
                    if(num%i ==0)
                    {
                        isPrime = false;
                        break;
                    }
                }

                if (isPrime == true)
                    System.out.println("Prime numbers from ActorA: "+num);
            }
            getContext().getSystem().terminate();
    }
}
```

The code defines an ActorA class that extends AbstractActor. The ActorA class defines a createReceive method that listens for messages of type MessageA and calls an onMessageA method when a message is received. The onMessageA method prints a message indicating that it received a number and then calculates prime numbers up to 5000. Finally, the onMessageA method terminates the ActorSystem.

- ActorB:

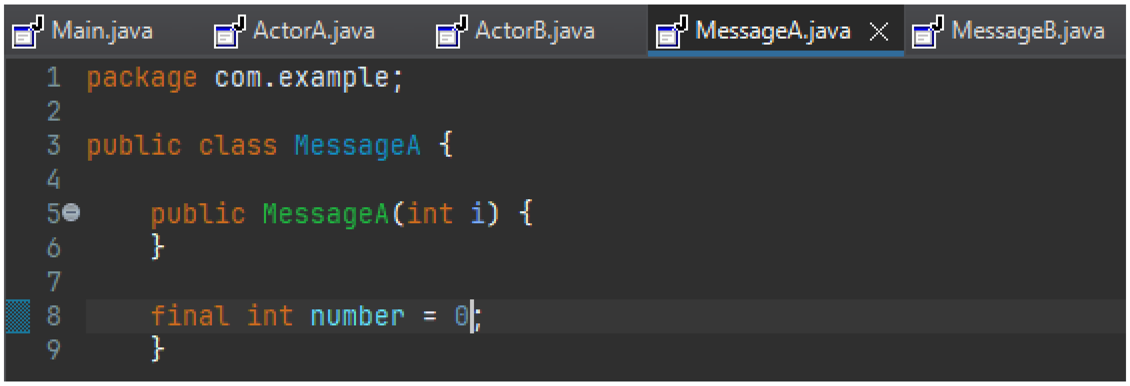


Main.java ActorA.java ActorB.java MessageA.java MessageB.java

```java
package com.example;

import akka.actor.AbstractActor;

public class ActorB extends AbstractActor {

    public static Props props() {
        return Props.create(ActorB.class, ActorB::new);
    }

    @Override
    public Receive createReceive() {
        return receiveBuilder()
                .match(MessageB.class, this::onMessageB)
                .build();
    }

    private void onMessageB(MessageB msg) {
            System.out.println("Actor B received a number: "+ msg.number + " from " + getSender());
            for (int num = msg.number; num <= 10000; num++)
            {
                boolean isPrime = true;
                for(int i=2; i<= num/2; i++)
                {
                    if(num%i ==0)
                    {
                        isPrime = false;
                        break;
                    }
                }

                if (isPrime == true)
                    System.out.println("Prime numbers from ActorB: "+num);
            }
            getContext().getSystem().terminate();
    }
```

The code defines an ActorB class that extends AbstractActor. The ActorB class defines a createReceive method that listens for messages of type MessageB and calls an onMessageB method when a message is received. The onMessageB method prints a message indicating that it received a number and then calculates prime numbers up to 10000. Finally, the onMessageB method terminates the ActorSystem.

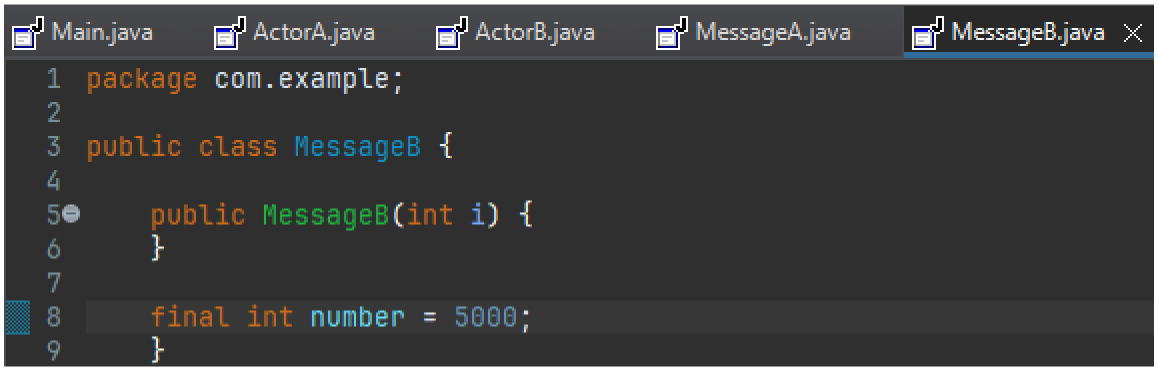
- MessageA:



```java
package com.example;

public class MessageA {

    public MessageA(int i) {
    }

    final int number = 0;
    }
```

This is a Java class for a message object, MessageA, which contains an integer number 0.

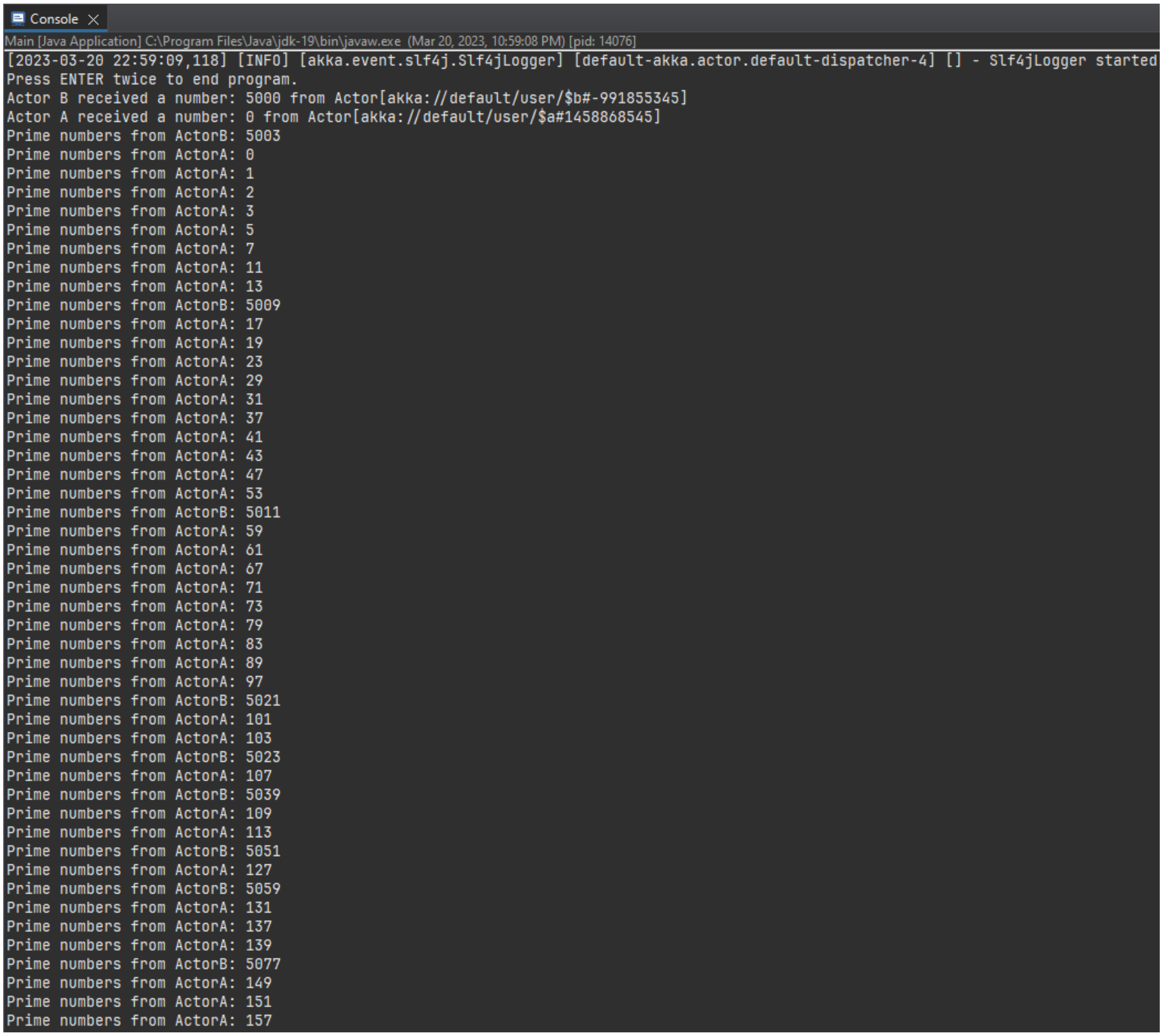
- MessageB:



```java
package com.example;

public class MessageB {

    public MessageB(int i) {
    }

    final int number = 5000;
    }
```

This is a Java class for a message object, MessageB, which contains an integer number 5000.

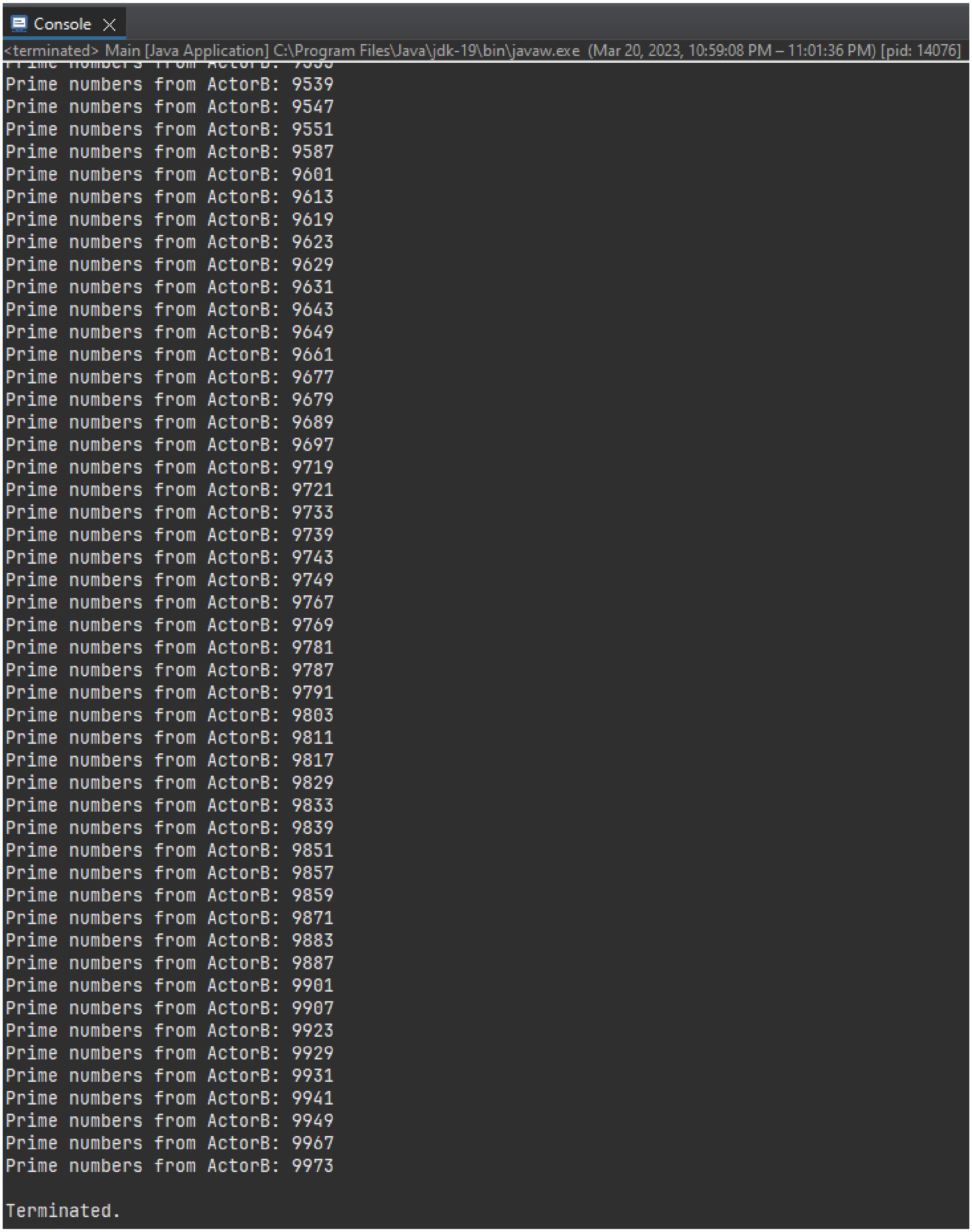
- Output:



Console

Main [Java Application] C:\Program Files\Java\jdk-19\bin\javaw.exe (Mar 20, 2023, 10:59:08 PM) [pid: 14076]

```
[2023-03-20 22:59:09,118] [INFO] [akka.event.slf4j.Slf4jLogger] [default-akka.actor.default-dispatcher-4] [] - Slf4jLogger started
Press ENTER twice to end program.
Actor B received a number: 5000 from Actor[akka://default/user/$b#-991855345]
Actor A received a number: 0 from Actor[akka://default/user/$a#1458868545]
Prime numbers from ActorB: 5003
Prime numbers from ActorA: 0
Prime numbers from ActorA: 1
Prime numbers from ActorA: 2
Prime numbers from ActorA: 3
Prime numbers from ActorA: 5
Prime numbers from ActorA: 7
Prime numbers from ActorA: 11
Prime numbers from ActorA: 13
Prime numbers from ActorB: 5009
Prime numbers from ActorA: 17
Prime numbers from ActorA: 19
Prime numbers from ActorA: 23
Prime numbers from ActorA: 29
Prime numbers from ActorA: 31
Prime numbers from ActorA: 37
Prime numbers from ActorA: 41
Prime numbers from ActorA: 43
Prime numbers from ActorA: 47
Prime numbers from ActorA: 53
Prime numbers from ActorB: 5011
Prime numbers from ActorA: 59
Prime numbers from ActorA: 61
Prime numbers from ActorA: 67
Prime numbers from ActorA: 71
Prime numbers from ActorA: 73
Prime numbers from ActorA: 79
Prime numbers from ActorA: 83
Prime numbers from ActorA: 89
Prime numbers from ActorA: 97
Prime numbers from ActorB: 5021
Prime numbers from ActorA: 101
Prime numbers from ActorA: 103
Prime numbers from ActorB: 5023
Prime numbers from ActorA: 107
Prime numbers from ActorB: 5039
Prime numbers from ActorA: 109
Prime numbers from ActorA: 113
Prime numbers from ActorB: 5051
Prime numbers from ActorA: 127
Prime numbers from ActorB: 5059
Prime numbers from ActorA: 131
Prime numbers from ActorA: 137
Prime numbers from ActorA: 139
Prime numbers from ActorB: 5077
Prime numbers from ActorA: 149
Prime numbers from ActorA: 151
Prime numbers from ActorA: 157
```


Console

<terminated> Main [Java Application] C:\Program Files\Java\jdk-19\bin\javaw.exe (Mar 20, 2023, 10:59:08 PM – 11:01:36 PM) [pid: 14076]

```
Prime numbers from ActorB: 9539
Prime numbers from ActorB: 9547
Prime numbers from ActorB: 9551
Prime numbers from ActorB: 9587
Prime numbers from ActorB: 9601
Prime numbers from ActorB: 9613
Prime numbers from ActorB: 9619
Prime numbers from ActorB: 9623
Prime numbers from ActorB: 9629
Prime numbers from ActorB: 9631
Prime numbers from ActorB: 9643
Prime numbers from ActorB: 9649
Prime numbers from ActorB: 9661
Prime numbers from ActorB: 9677
Prime numbers from ActorB: 9679
Prime numbers from ActorB: 9689
Prime numbers from ActorB: 9697
Prime numbers from ActorB: 9719
Prime numbers from ActorB: 9721
Prime numbers from ActorB: 9733
Prime numbers from ActorB: 9739
Prime numbers from ActorB: 9743
Prime numbers from ActorB: 9749
Prime numbers from ActorB: 9767
Prime numbers from ActorB: 9769
Prime numbers from ActorB: 9781
Prime numbers from ActorB: 9787
Prime numbers from ActorB: 9791
Prime numbers from ActorB: 9803
Prime numbers from ActorB: 9811
Prime numbers from ActorB: 9817
Prime numbers from ActorB: 9829
Prime numbers from ActorB: 9833
Prime numbers from ActorB: 9839
Prime numbers from ActorB: 9851
Prime numbers from ActorB: 9857
Prime numbers from ActorB: 9859
Prime numbers from ActorB: 9871
Prime numbers from ActorB: 9883
Prime numbers from ActorB: 9887
Prime numbers from ActorB: 9901
Prime numbers from ActorB: 9907
Prime numbers from ActorB: 9923
Prime numbers from ActorB: 9929
Prime numbers from ActorB: 9931
Prime numbers from ActorB: 9941
Prime numbers from ActorB: 9949
Prime numbers from ActorB: 9967
Prime numbers from ActorB: 9973

Terminated.
```