CSCI 344/715
Lecturer: Simina Fluture

Topics: Java Message Passing

Synchronous Message Passing
Asynchronous Message Passing
MessagePassing with pipes

Readings: **Lecture on the web**
Class notes
[SH] 6.2.1, 6.2.2

We will consider the following definitions: (Hartley, page 169)

Send( )

1. Blocking: waits for receiver or receiver is already waiting
2. Buffered, non-blocking: message is buffered but the receiver has not necessarily gotten it yet.
3. Non-buffered, non-blocking: returns an error if no receiver is ready or waiting, returns OK if message is sent and received.

Receive( )

1. Blocking: waits for message, sender to send
2. Buffered, non-blocking: returns an error if no message is waiting, else returns OK
3. Non-buffered, non-blocking: returns an error if no sender is ready or waiting, returns OK if message is sent and received.

(2/3, 3/2, 3/3 are not possible. Explain why)

All receivers are blocking. The receiving process does not have to use busy waiting until the message arrives. The receiver is delayed until at least one message exists.
In **synchronous** message passing, **send** is **blocking** primitive.
In **asynchronous** message passing, the **send** is **buffered / non-blocking**.

The threads can be in the same JVM or in different JVMs.

Synchronous Message Passing

The synchronous message-passing example uses 2 binary semaphores **senderIn** and **ReceiverIn.**

send(Object m) object receive( )

Note: one sender synchronizes with one receiver at a time.

CSCI 344/715
Lecturer: Simina Fluture

Asynchronous Message Passing

The asynchronous message passing example uses a vector as a buffer to retain sent but not yet received messages.

Asynchronous Message Passing

```
send(Object m) {
        If (m=null) {
                System.err.println("null message passed to send( )");
                Throw new NullPointerEception( );
        }
        numMessages++;
        messages.addElement(m);
        if (numMessages < = 0) notify( );
}

object receive( ) {
numMessages--;
if (numMessages < 0) {
   while(true) {
        try {
                wait( );
                break;
        }
        catch (InterruptedException e) {
                System.err.println
                ("…………………………..");
                if (numMessages >=0) break;
                else continue;                          }
         }
    }
}
receivedMessage = messages.firstElement( );
messages.removeElementAt(0);
return receivedMessage;
}
```

In the synchronization package used by Java, both synchronous and asynchronous message passing are implemented.

Message Passing with Pipes

Bring printed copies of the posted programs with you:

ConnectionManager.java, Message,java, Receiver.java, Sender.java