***JAVA***

Message Unsafe

public class MessageUnsafe {

public static void main(String[] args){

UnsafeSender mySender = new UnsafeSender();

Thread oneMessage = new UnsafeMessageThread(mySender,"Hello");

Thread twoMessage = new UnsafeMessageThread(mySender,"World");

oneMessage.start();

twoMessage.start();

}

}

class UnsafeSender {

String message;

public void send(String msg) {

System.out.println("Sending: " + msg );

message = msg;

}

public void receive() {

System.out.println(message);

}

}

class UnsafeMessageThread extends Thread{

private UnsafeSender sender;

private String message;

public UnsafeMessageThread(UnsafeSender aSender, String aMessage) {

sender = aSender;

message = aMessage;

}

@Override

public void run() {

try {

sender.send(message);

sleep(1000);

sender.receive();

} catch (InterruptedException e) {}

}

}

Message Semaphore

import java.util.concurrent.Semaphore;

public class MessageSemaphore {

public static void main(String[] args){

SemaphoreSender mySender = new SemaphoreSender();

Semaphore mySemaphore = new Semaphore(1);

Thread oneMessage = new SemaphoreMessageThread(mySender,"Hello",mySemaphore);

Thread twoMessage = new SemaphoreMessageThread(mySender,"World",mySemaphore);

oneMessage.start();

twoMessage.start();

}

}

class SemaphoreSender {

String message;

public void send(String msg) {

System.out.println("Sending: " + msg );

message = msg;

}

public void receive() {

System.out.println("Received: " + message);

}

}

class SemaphoreMessageThread extends Thread{

private SemaphoreSender sender;

private String message;

private Semaphore semaphore;

public SemaphoreMessageThread(SemaphoreSender aSender, String aMessage, Semaphore aSemaphore) {

sender = aSender;

message = aMessage;

semaphore = aSemaphore;

}

@Override

public void run() {

try {

semaphore.acquire();

sender.send(message);

sleep(1000);

sender.receive();

semaphore.release();

} catch (InterruptedException e) {}

}

}

Message Examples

class WaitingSender {

//Do with and without wait

String message;

synchronized public void send(String msg) {

if(message != null) {

try {

wait();

} catch (InterruptedException e) {}

}

System.out.println("Sending: " + msg );

message = msg;

}

synchronized public void receive() {

System.out.println(message);

message = null;

notify();

}

}

class WaitingMessageThread extends Thread{

private WaitingSender sender;

private String message;

public WaitingMessageThread(WaitingSender aSender, String aMessage) {

sender = aSender;

message = aMessage;

}

@Override

public void run() {

try {

sender.send(message);

sleep(1000);

sender.receive();

} catch (InterruptedException e) {}

}

}

public class MessageExample {

public static void main(String[] args){

WaitingSender mySender = new WaitingSender();

Thread oneMessage = new WaitingMessageThread(mySender,"Hello");

Thread twoMessage = new WaitingMessageThread(mySender,"World");

oneMessage.start();

twoMessage.start();

}

}