Producer Consumer example

a simple producer-consumer scenario where one thread (Producer) produces data and

another thread (Consumer) consumes it. The wait(), notify(), and notifyAll() methods are used

for communication between the threads.

*class* Data {
 *private* String packet;
 *private boolean* transfer = *true*;

 *// Synchronized method to send data
 public synchronized void* send(String packet) {
 *while* (!transfer) {
 *try* {
 wait();
 } *catch* (InterruptedException e) {
 Thread.*currentThread*().interrupt();
 System.*out*.println("Thread interrupted");
 }
 }
 transfer = *false*;
 *this*.packet = packet;
 notifyAll();
 }

 *// Synchronized method to receive data
 public synchronized* String receive() {
 *while* (transfer) {
 *try* {
 wait();
 } *catch* (InterruptedException e) {
 Thread.*currentThread*().interrupt();
 System.*out*.println("Thread interrupted");
 }
 }
 transfer = *true*;
 notifyAll();
 *return* packet;
 }
}

*class* Sender *implements Runnable* {
 *private* Data data;

 *public* Sender(Data data) {
 *this*.data = data;
 }

 @Override
 *public void* run() {
 String packets[] = {
 "First packet",
 "Second packet",
 "Third packet",
 "Fourth packet",
 "End"
 };

 *for* (String packet : packets) {
 data.send(packet);
 System.*out*.println("Sending " + packet);
 *try* {
 Thread.*sleep*(500);
 } *catch* (InterruptedException e) {
 Thread.*currentThread*().interrupt();
 System.*out*.println("Thread interrupted");
 }
 }
 }
}

*class* Receiver *implements Runnable* {
 *private* Data data;

 *public* Receiver(Data data) {
 *this*.data = data;
 }

 @Override
 *public void* run() {
 *for* (String receivedMessage = data.receive();
 !"End".equals(receivedMessage);
 receivedMessage = data.receive()) {

 System.*out*.println("Received: " + receivedMessage);

 *try* {
 Thread.*sleep*(1000);
 } *catch* (InterruptedException e) {
 Thread.*currentThread*().interrupt();
 System.*out*.println("Thread interrupted");
 }
 }
 }
}

*public class* ProducerConsumer {
 *public static void* main(String[] args) {
 Data data = *new* Data();
 Thread sender = *new* Thread(*new* Sender(data));
 Thread receiver = *new* Thread(*new* Receiver(data));

 sender.start();
 receiver.start();
 }
}

Received: First packet

Sending First packet

Sending Second packet

Received: Second packet

Sending Third packet

Received: Third packet

Sending Fourth packet

Received: Fourth packet

Sending End