Robotics and Autonomous Systems

Lecture 23: LeJOS and Jason

Richard Williams

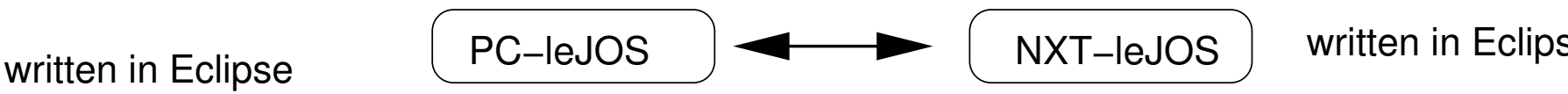
Department of Computer Science
University of Liverpool

UNIVERSITY OF LIVERPOOL

Today

- Today we will just go through an example of how to interface Jason with LeJOS
- This is essential to be able to do Assignment 2.

Jason and LeJOS



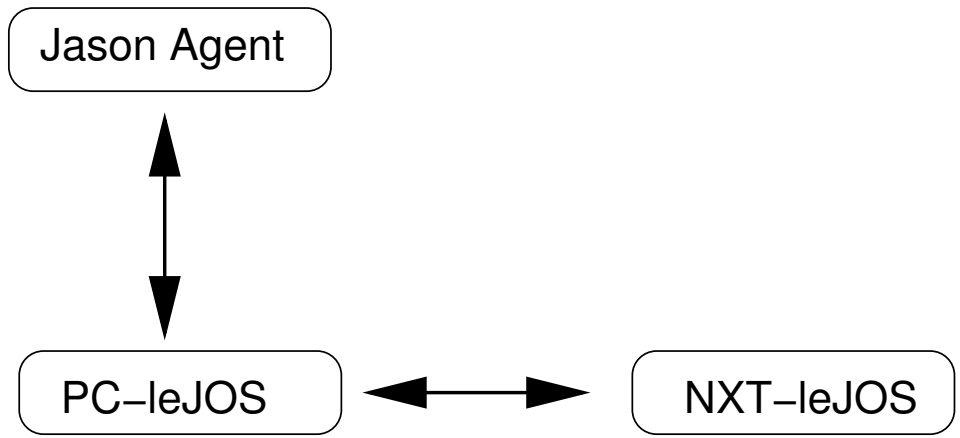
Jason and LeJOS

written in jEdit
or Eclipse

written in Eclipse

Jason and LeJOS

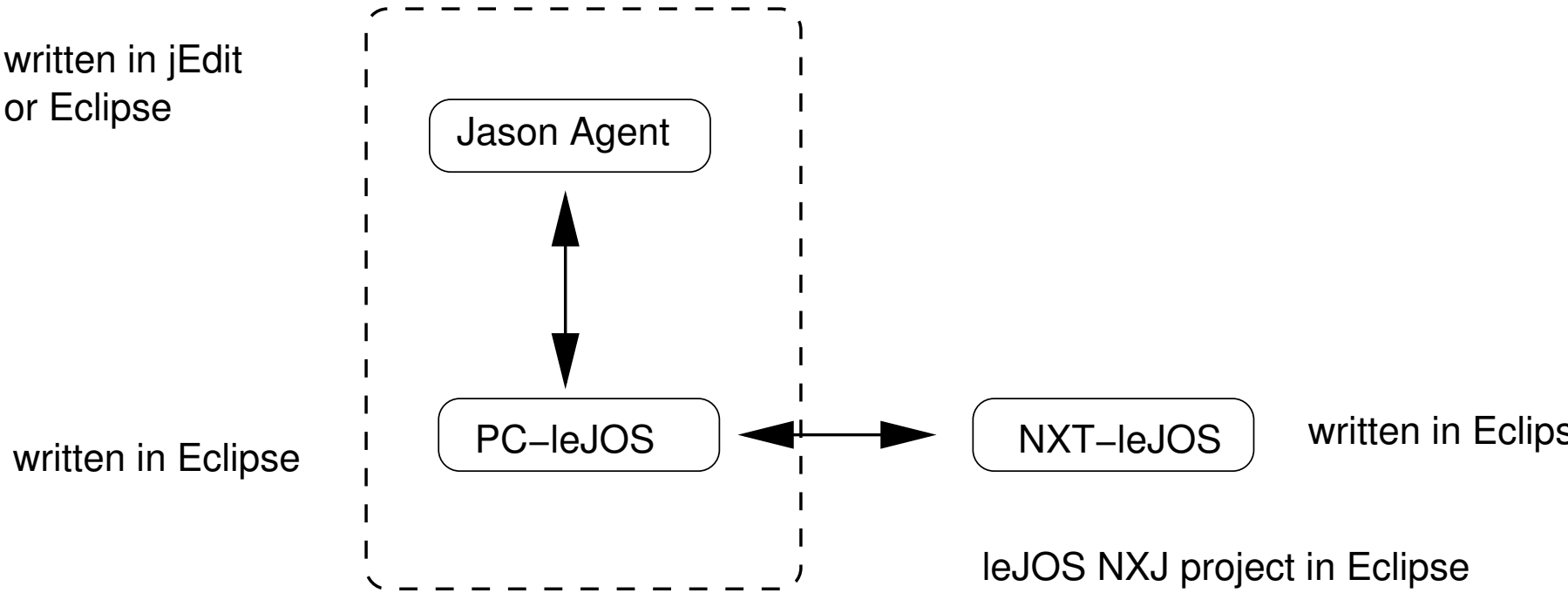
written in jEdit
or Eclipse



written in Eclipse

written in Eclips

Jason and LeJOS



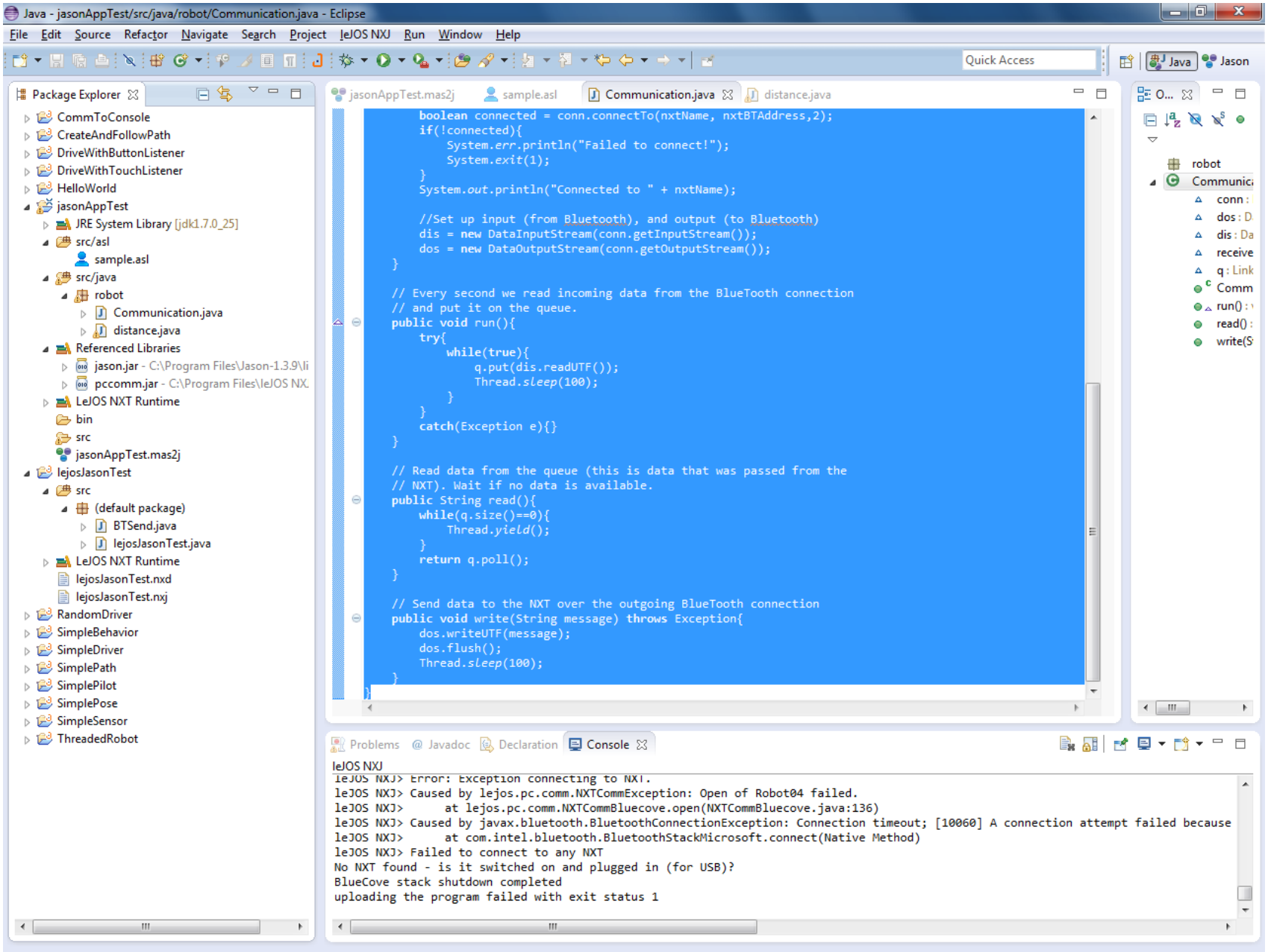
Jason Project in Eclipse containing a package with PC LeJOS code. The project loads classes for PC LeJOS

Jason and LeJOS

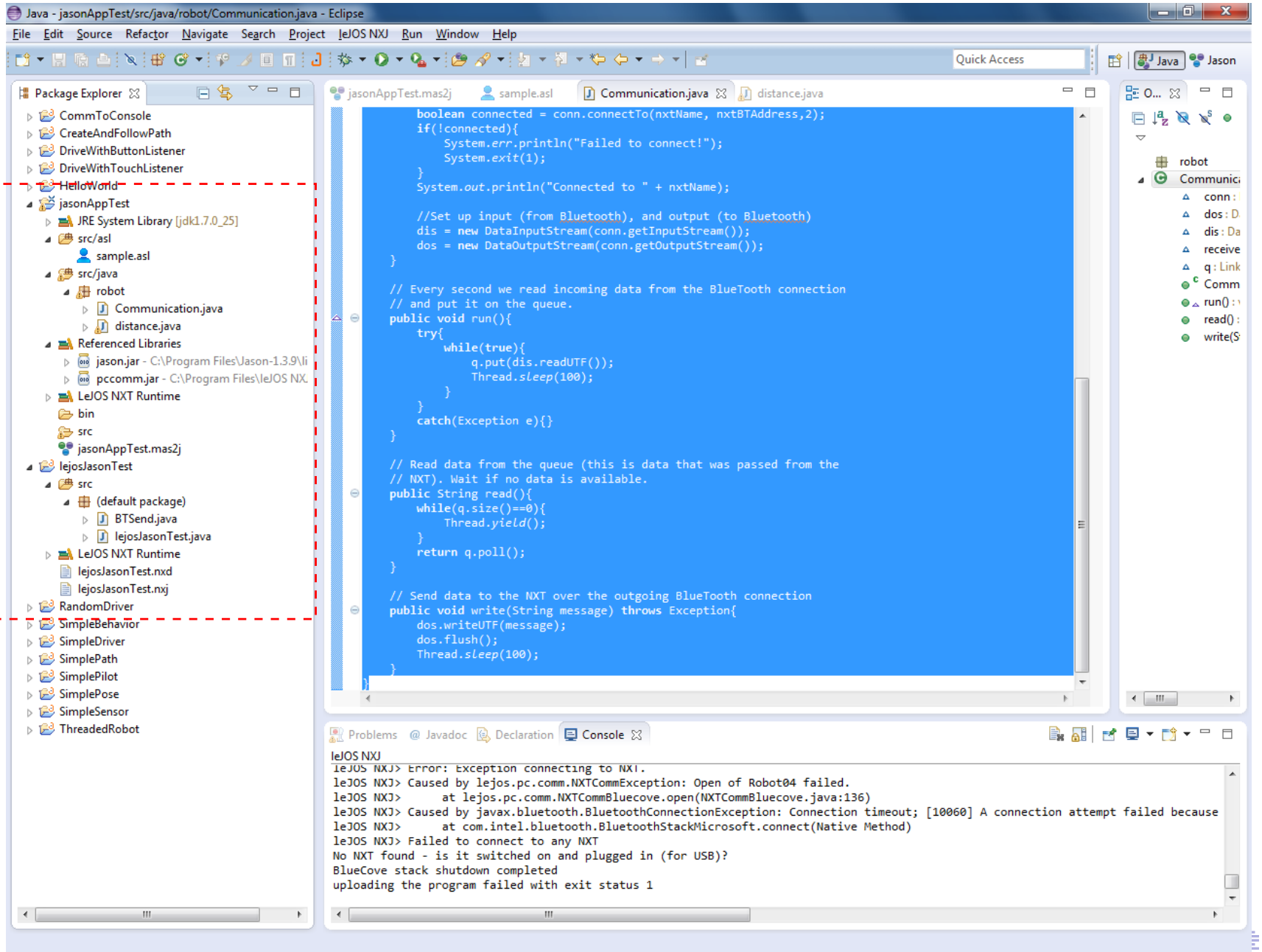
- A Jason agent is interfaced with a LeJOS PC program
  - The robot provides the environment for the agent.

  which in turn communicates with an NXT brick.
- The Jason agent uses an action `distance` to obtain information about the distance read by the US sensor on the NXT brick.
- The action is hooked to a method of a LeJOS PC program, which obtains the reading through BT from the NXT brick.
- The program is run as a Jason project

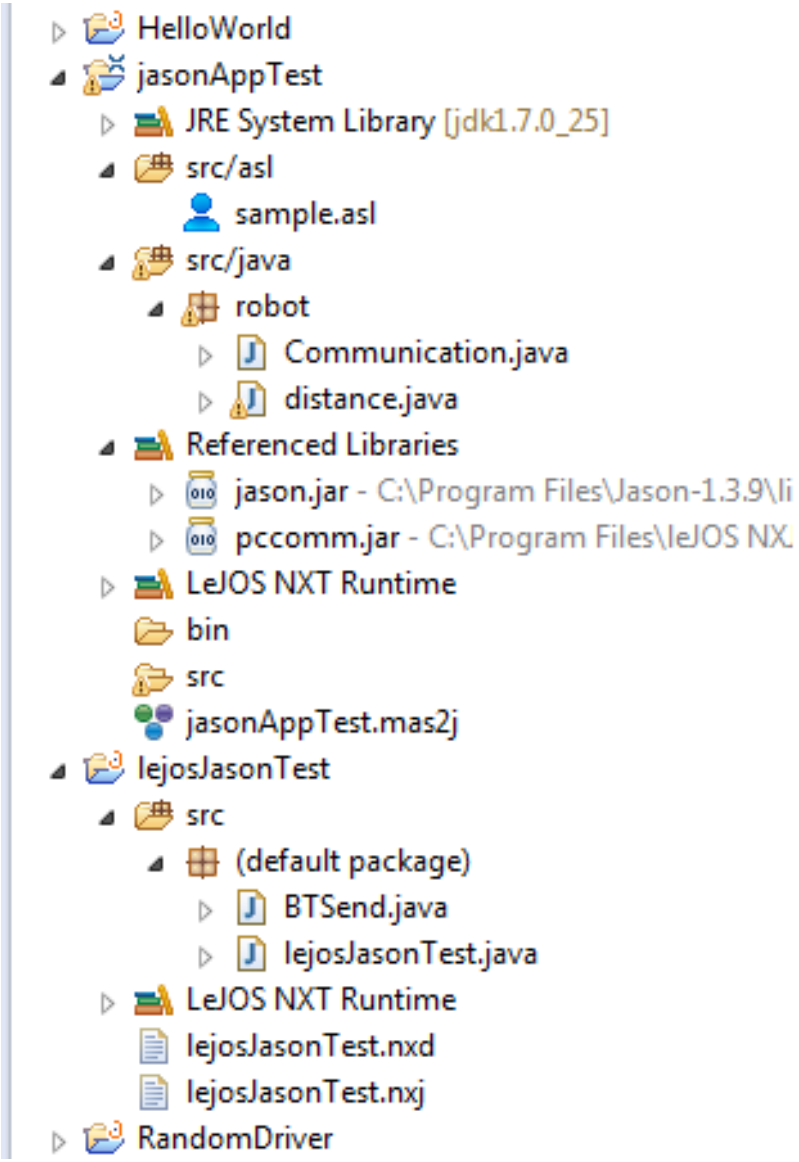
Jason and LeJOS



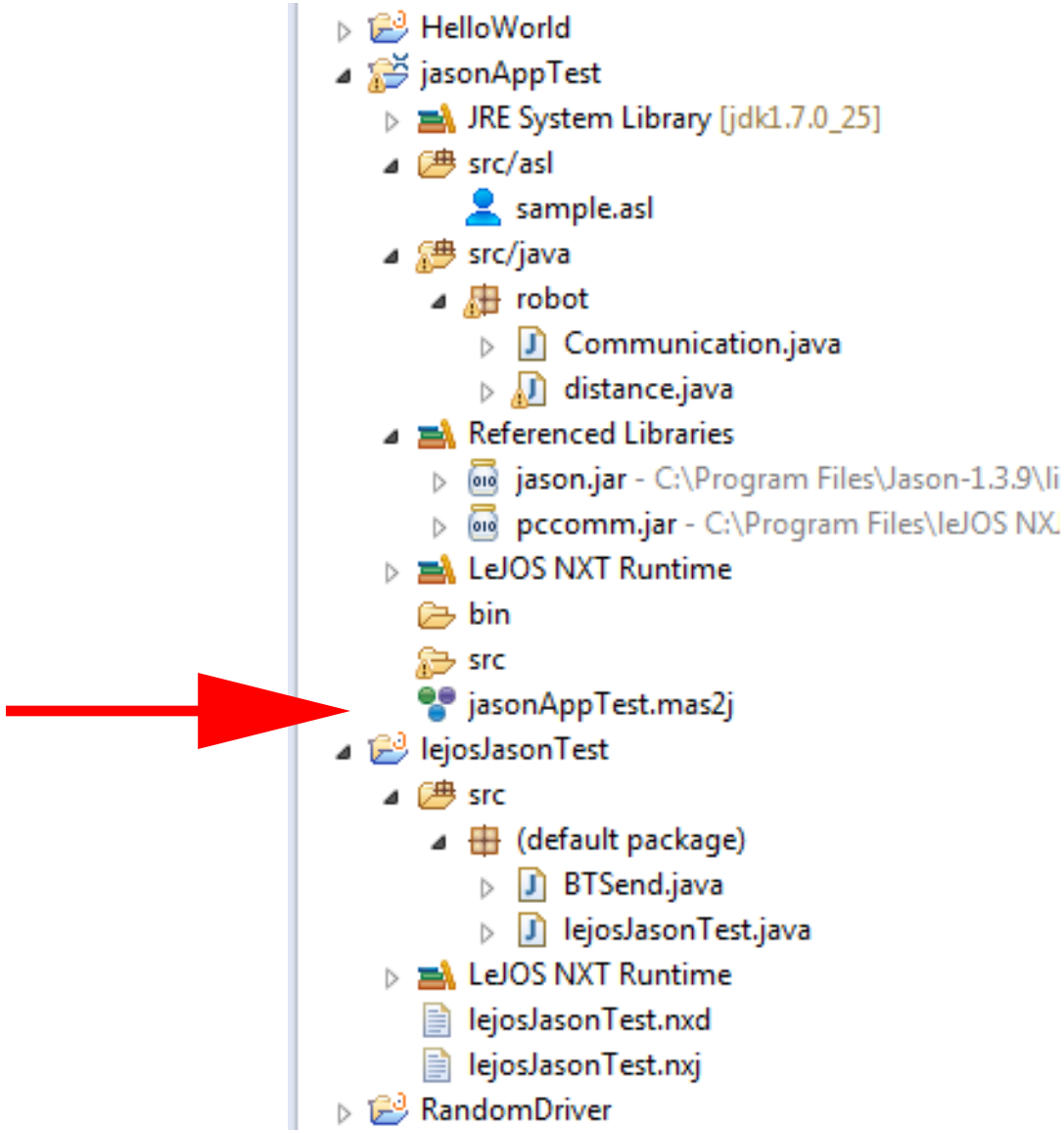
Jason and LeJOS



Jason and LeJOS



Jason and LeJOS



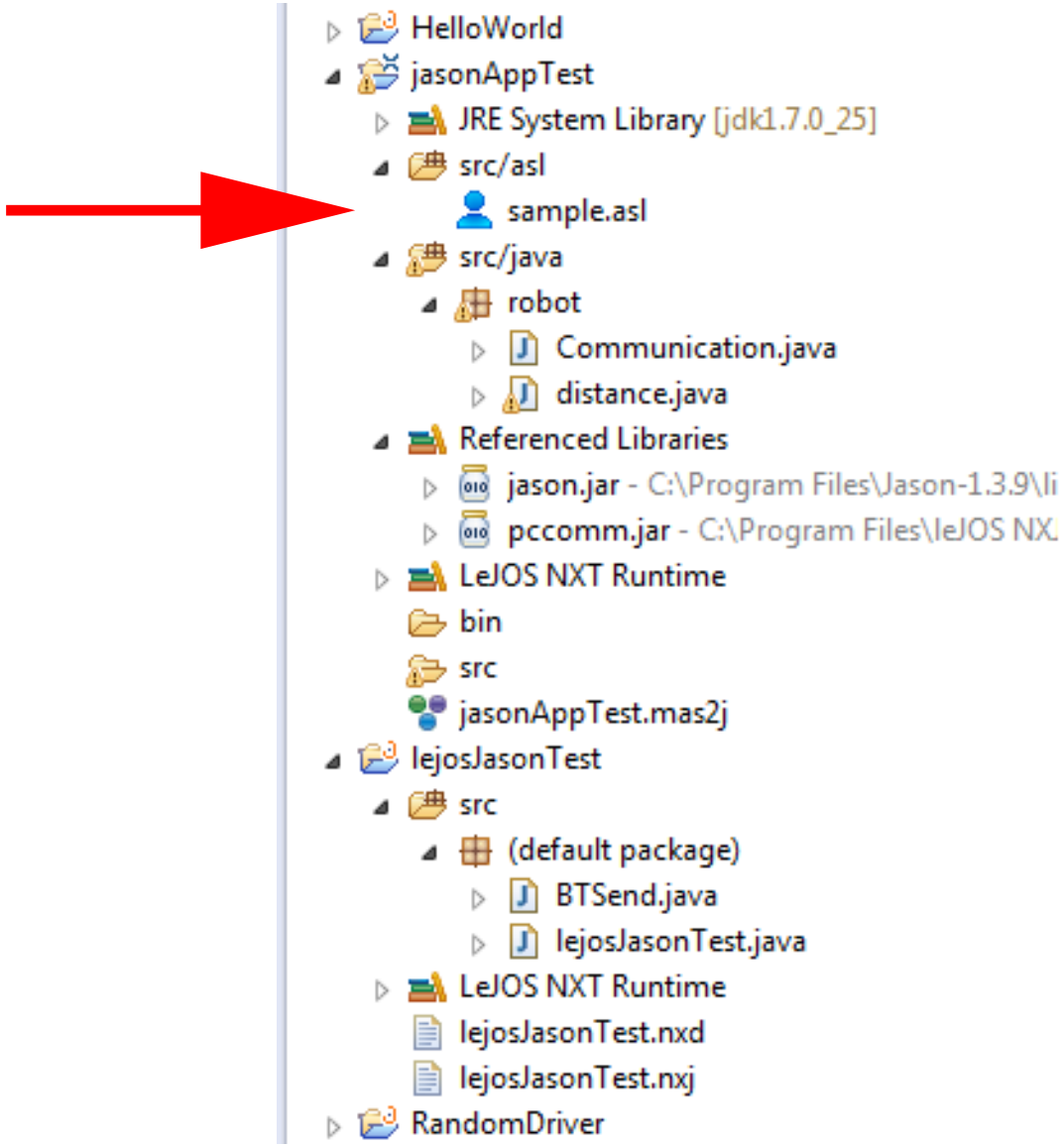
The MAS file

```
MAS jasonAppTest {

  infrastructure: Centralised

  agents:
    agent1 sample;

  classpath:
    "C:/Program Files/leJOS NXJ/lib/pc/*.jar";
    "C:/Program Files/leJOS NXJ/lib/pc/3rdparty/*.jar";

  aslSourcePath:
    "src/asl";
}
```

The MAS file

- Note the classpath to the leJOS PC library.
- This tells Jason where these files are.
- The compilation of the Jason part is carried out by Jason, invoked by Eclipse.
- Since this compilation involves compiling leJOS/PC code, Jason needs to know where the library is.
- (Since Jason is written in Java, a Jason program could include any other Java library also).

The MAS file


HelloWorld
jasonAppTest
JRE System Library [jdk1.7.0_25]
src/asl
sample.asl
src/java
robot
Communication.java
distance.java
Referenced Libraries
jason.jar - C:\Program Files\Jason-1.3.9\li
pccomm.jar - C:\Program Files\leJOS NX
LeJOS NXT Runtime
bin
src
jasonAppTest.mas2j
lejosJasonTest
src
(default package)
BTSend.java
lejosJasonTest.java
LeJOS NXT Runtime
lejosJasonTest.nxd
lejosJasonTest.nxj
RandomDriver

The agent

```
/* Initial beliefs and rules */

/* Initial goals */

!start.

/* Plans */

+!start : true
    <- robot.distance(Object);
    .print("Distance is", Object);
    .wait(1000);
    !start.
```

The agent

- Begin with the (aptly named) goal `!start`
- This calls the actions:

```
robot.distance
.print
.wait
```

- Then reinvokes `!start`
- Jason equivalent of an infinite loop.

The agent

- Where does `robot.distance` come from?

The agent

- Where does `robot.distance` come from?
- It is part of the environment.

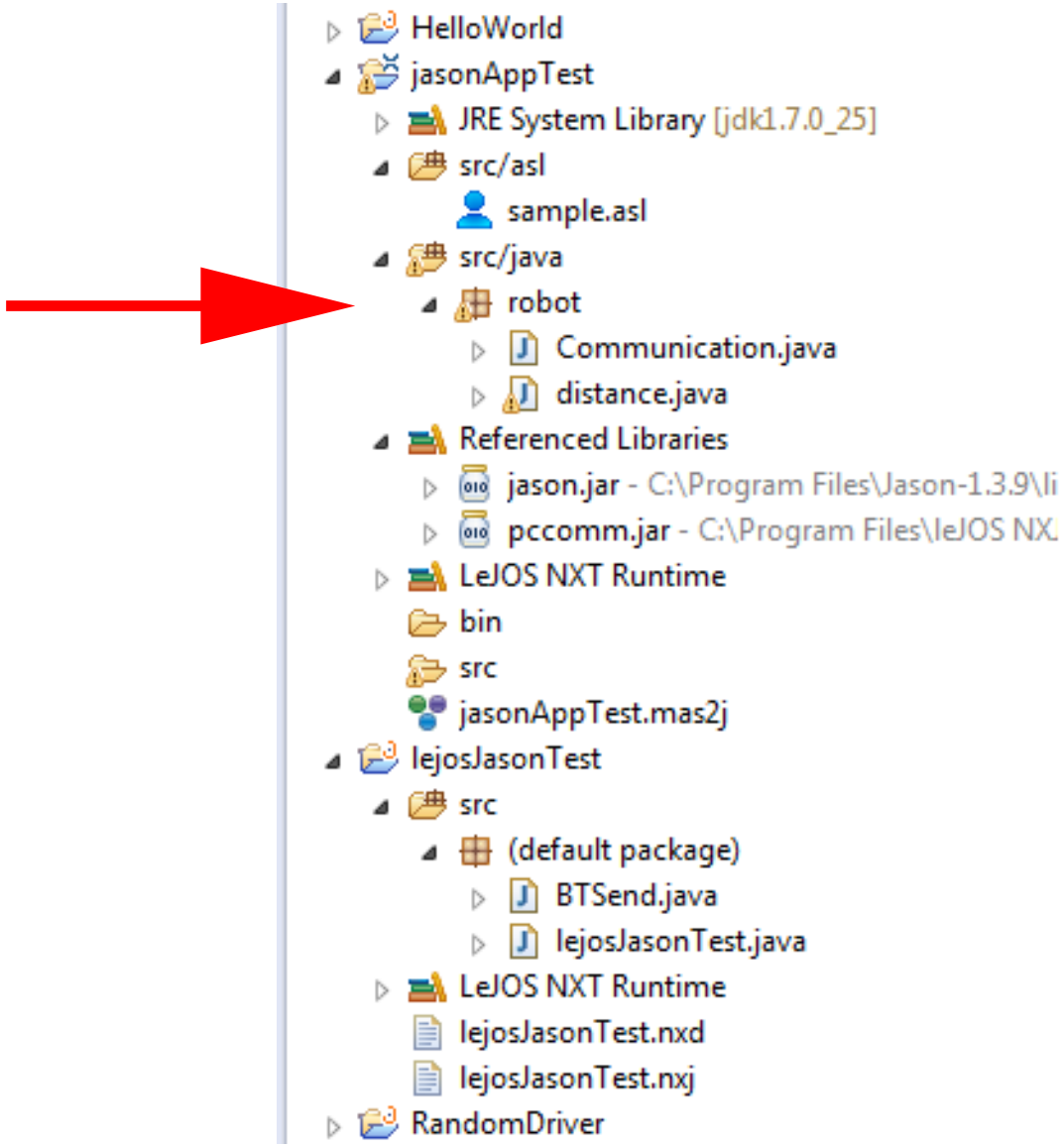
The agent

- Where does `robot.distance` come from?
- It is part of the environment.
- Which in this case is provided by the robot.

The agent

- Where does `robot.distance` come from?
- It is part of the environment.
- Which in this case is provided by the robot.
- It is a command to the robot to do something.

The agent


HelloWorld
jasonAppTest
JRE System Library [jdk1.7.0_25]
src/asl
sample.asl
src/java
robot
Communication.java
distance.java
Referenced Libraries
jason.jar - C:\Program Files\Jason-1.3.9\li
pccomm.jar - C:\Program Files\leJOS NX
LeJOS NXT Runtime
bin
src
jasonAppTest.mas2j
lejosJasonTest
src
(default package)
BTSend.java
lejosJasonTest.java
LeJOS NXT Runtime
lejosJasonTest.nxd
lejosJasonTest.nxj
RandomDriver

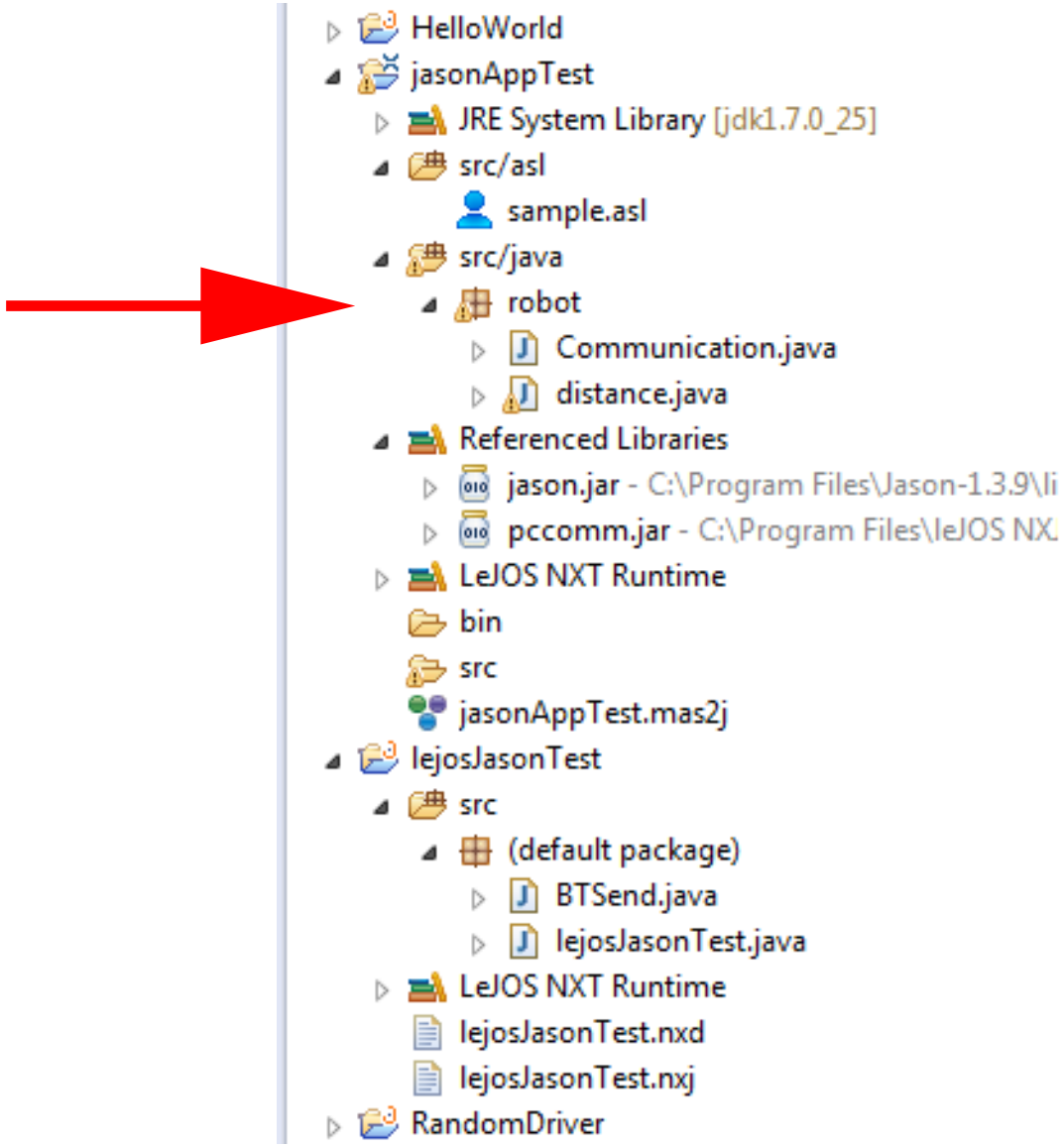
The agent

- This is the package that provides the “environment” in which the `sample.asl` agent runs.
- It is a Java program that runs on the PC.
  - Provides the `distance` command.
- As far as Jason is concerned, that is all it needs to know.

The agent

- This is the package that provides the “environment” in which the `sample.asl` agent runs.
- It is a Java program that runs on the PC.
  - Provides the `distance` command.
- As far as Jason is concerned, that is all it needs to know.
- In reality this program talks to an NXT robot.

The agent


HelloWorld
jasonAppTest
JRE System Library [jdk1.7.0_25]
src/asl
sample.asl
src/java
robot
Communication.java
distance.java
Referenced Libraries
jason.jar - C:\Program Files\Jason-1.3.9\li
pccomm.jar - C:\Program Files\leJOS NX
LeJOS NXT Runtime
bin
src
jasonAppTest.mas2j
lejosJasonTest
src
(default package)
BTSend.java
lejosJasonTest.java
LeJOS NXT Runtime
lejosJasonTest.nxd
lejosJasonTest.nxj
RandomDriver

Communication

- Two bits:
  - `Communication.java`
  - `distance.java`

Communication.java

```
package robot;

import java.io.*;
import java.util.concurrent.LinkedBlockingQueue;

import lejos.pc.comm.NXTConnector;

public class Communication implements Runnable{

  NXTConnector conn;
  DataOutputStream dos;
  DataInputStream dis;

  String received;
  LinkedBlockingQueue<String> q;
```

Communication.java

- This is the communication infrastructure object.
- `NXTConnector` to provide the connection.
- Two `streams`, for passing data over the connection.
- A `LinkedBlockingQueue<String>` for buffering data for transport.
- The whole thing is `Runnable` to allow it to be easily threaded.

Communication.java

```
public Communication(String nxtName, String nxtBTAddress){
  received=null;
  q = new LinkedBlockingQueue<String>();

  // Open up a BlueTooth connection
  conn = new NXTConnector();
  boolean connected = conn.connectTo(nxtName, nxtBTAddress,2);
  if(!connected){
    System.err.println("Failed to connect!");
    System.exit(1);
  }
  System.out.println("Connected to " + nxtName);

  //Set up input (from Bluetooth), and output (to Bluetooth)
  dis = new DataInputStream(conn.getInputStream());
  dos = new DataOutputStream(conn.getOutputStream());
}
```

Communication.java

- Pretty standard communication stuff.
- Exploits the Java/LeJOS infrastructure.
- Key line is
  `conn.connectTo(nxtName, nxtBTAddress,2);`
  which opens the connection.

Communication.java

- Then we add:

```
public void run(){
  try{
    while(true){
    q.put(dis.readUTF());
    Thread.sleep(100);
    }
  }
  catch(Exception e){}
}
```

Communication.java

- The “main” of the `Runnable`
  - Is kicked off when the `Runnable` is popped into a thread.
- Every tenth of a second it grabs from the input stream.
- Puts what it gets into the `q`.

Communication.java

- And:

```
public String read(){
  while(q.size()==0){
    Thread.yield();
  }
  return q.poll();
}
```

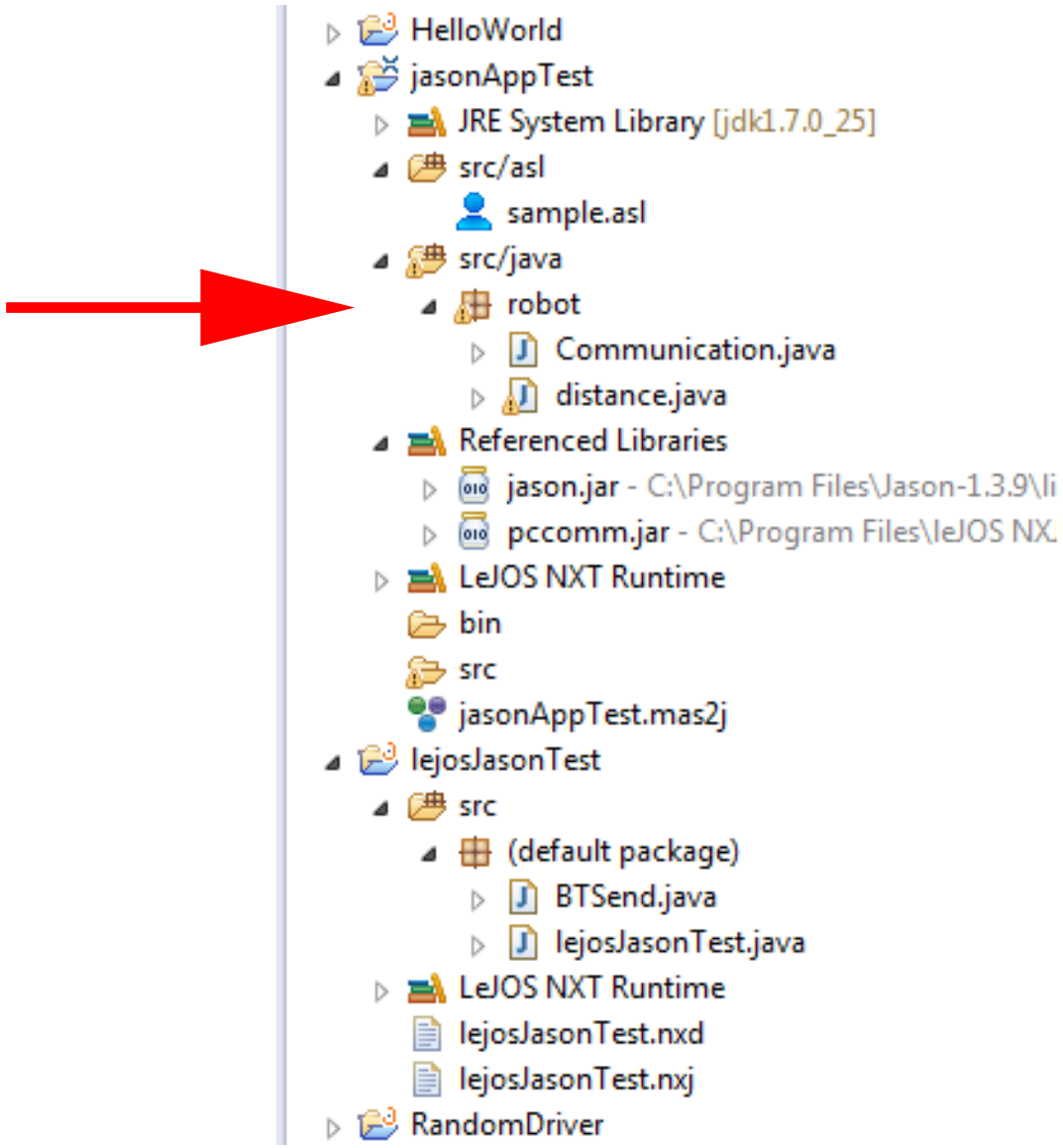
Communication.java

- Main interface function — read from the q.
- The reason for the q is to make this asynchronous
- Communication happens when it happens, and the results are stored in the q.
  Happens when the NXT is ready.
- The Jason program causes the q to be read when it is ready.
- q handles the slack.
- Being a blocking queue, it can't be over-filled, and it causes a wait if it is empty.

`Communication.java`

- `Communication.java` also provides a `write` method, but the example doesn't use it.

What else?


HelloWorld
jasonAppTest
JRE System Library [jdk1.7.0_25]
src/asl
sample.asl
src/java
robot
Communication.java
distance.java
Referenced Libraries
jason.jar - C:\Program Files\Jason-1.3.9\li
pccomm.jar - C:\Program Files\leJOS NX
LeJOS NXT Runtime
bin
src
jasonAppTest.mas2j
lejosJasonTest
src
(default package)
BTSend.java
lejosJasonTest.java
LeJOS NXT Runtime
lejosJasonTest.nxd
lejosJasonTest.nxj
RandomDriver

distance.java

```
package robot;

import jason.*;
import jason.asSemantics.*; // to do unification
import jason.asSyntax.*;    // to handle AgentSpeak
                            // syntax.

public class distance
          extends DefaultInternalAction {

  Communication comm;
  Thread commThread;
```

distance.java

- This is the connection to Jason.
- We create a Jason action by extending its default code for an action.
- This action will be a `read` from a `Communication` object.

distance.java

```
public distance(){
  comm = new Communication("NXT", "00:16:53:1a:a7:11");
  commThread = new Thread(comm);
  commThread.start();
}
```

distance.java

- The constructor opens up a channel using the `Communication` object.
- Note that this relies on the address of the specific NXT that you are connecting to.
- Not only is it paired, but it is coded to the robot.

distance.java

```
public Object execute(TransitionSystem ts,
                      Unifier un, Term[] args)
  throws Exception{

  String message = comm.read();
  StringTerm result = new StringTermImpl(message);

  return un.unifies(result,  args[0]);
}
```

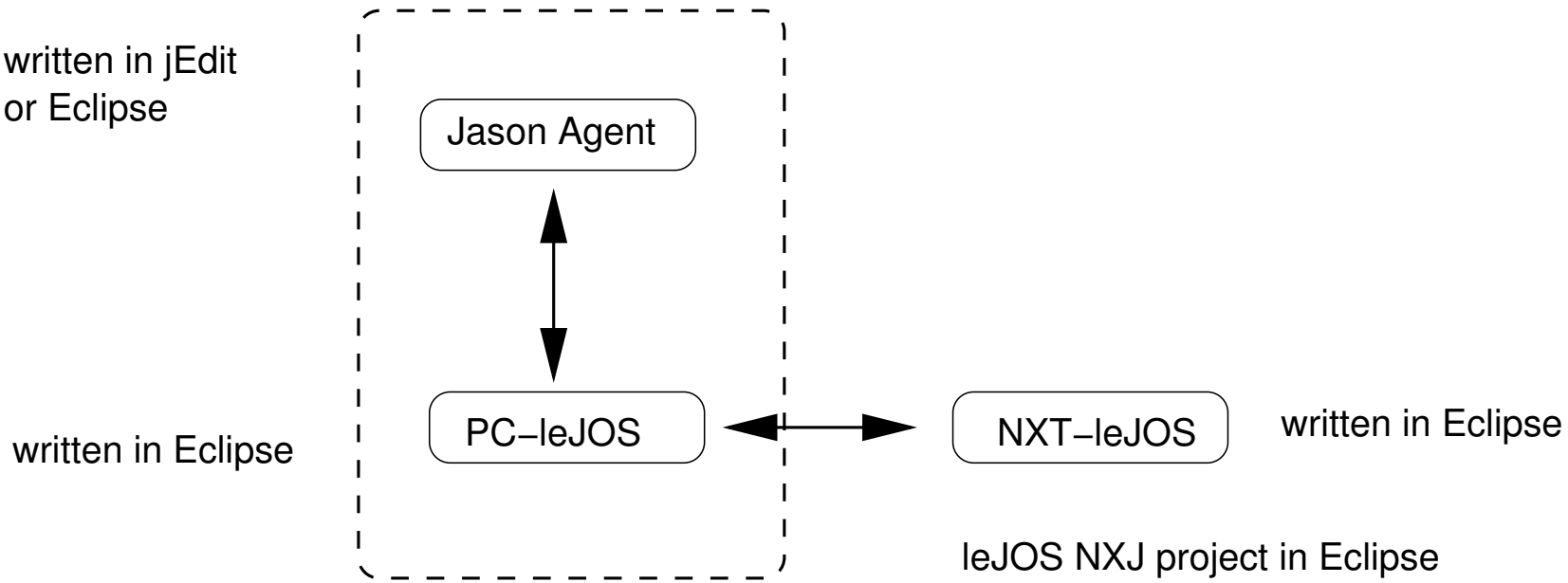
distance.java

- This is what gets called when Jason executes `robot.distance`
- `ts` is the state when the action is called.
  - So actions can refer to the agent state.
- `un` captures the unification
  - Gives you the access to the values of the variables in the plan which are referenced by the action.
- `args` are the arguments of the action.

distance.java

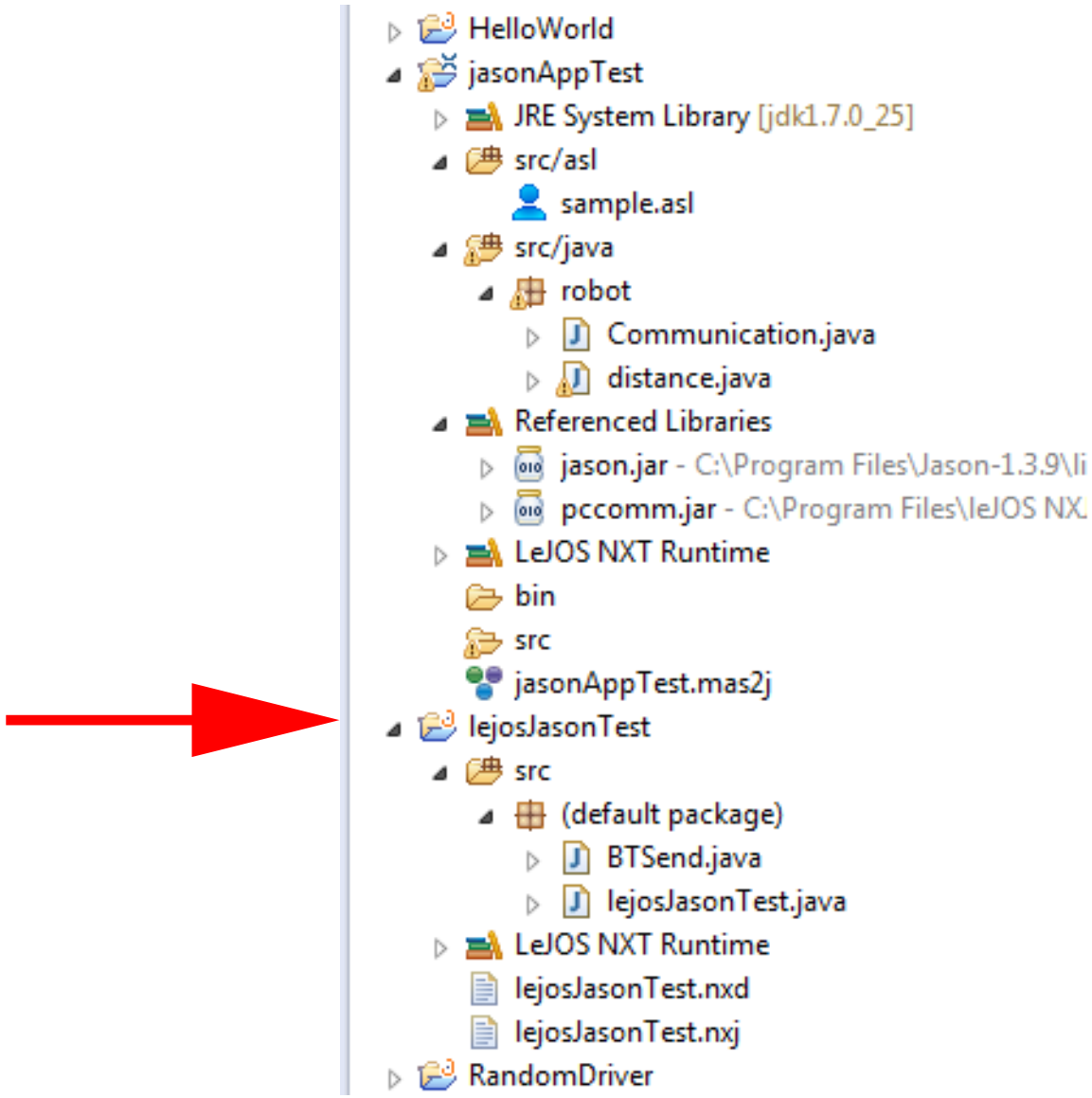
- This winds up what happens on the Jason side.
- More now to discuss what happens on the LeJOS side.
- This is just a regular LeJOS program.

On the LeJOS side



Jason Project in Eclipse containing a package with PC LeJOS code. The project loads classes for PC LeJOS

On the LeJOS side



BTSend.java

```
public class BTSend implements Runnable{
  Queue<String> q;
  NXTConnection conn;
  DataOutputStream out;

  public BTSend(NXTConnection conn){
    q = new Queue<String>();
    this.conn=conn;
    out=conn.openDataOutputStream();
  }
```

BTSend.java

- Looks a lot like `Communication.java`
- Uses `NXTConnection` to provide a connection over BlueTooth.
- Uses a (regular) `q` to load up data to send.
- Only has an output stream.

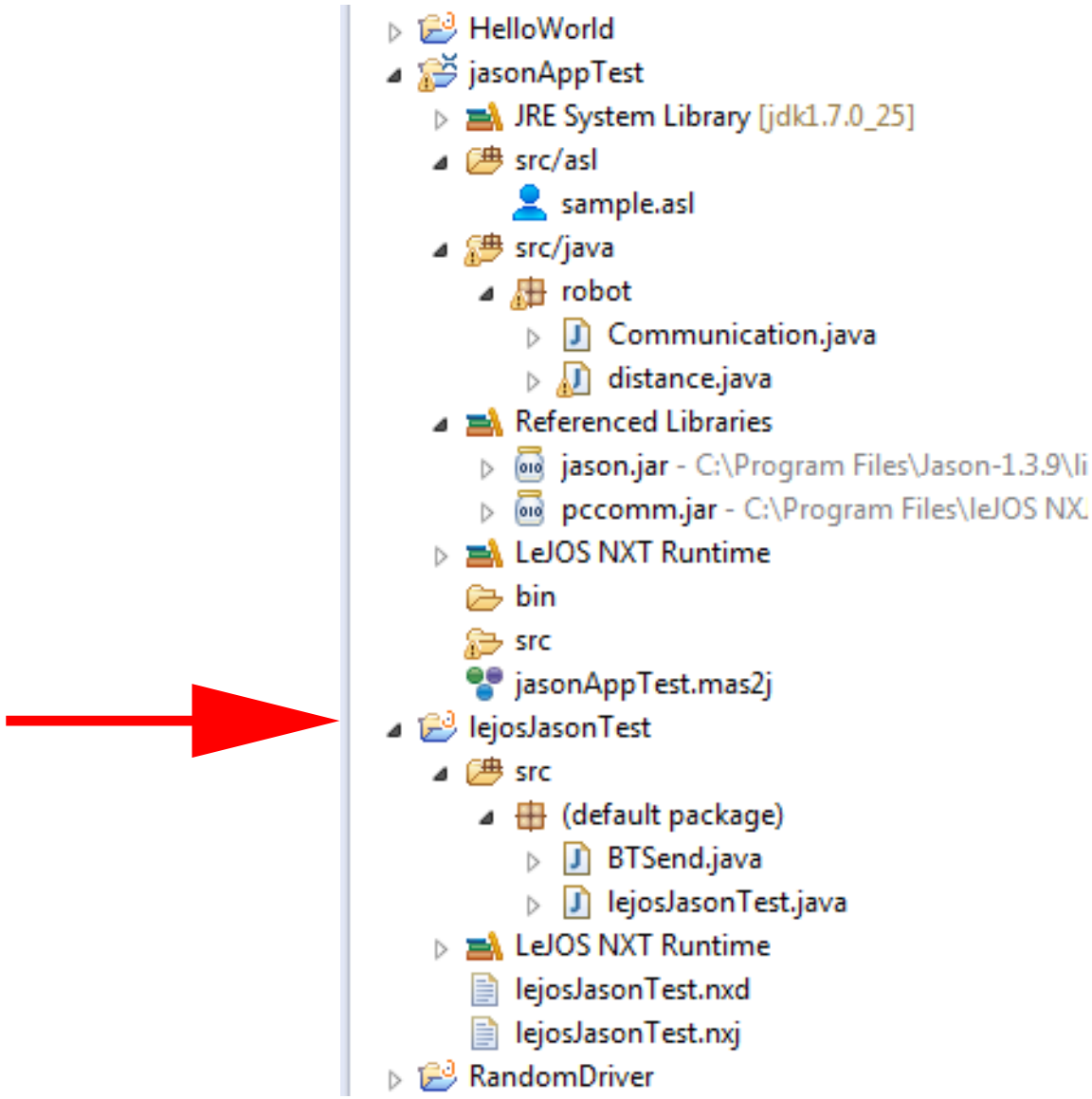
BTSend.java

```
public void run(){
  try{
    while(true){
      while(q.empty()){
        Thread.yield();
      }
      out.writeUTF((String)q.pop());
      out.flush();
    }
  }
  catch (Exception e) {}
}
```

BTSend.java

- Do nothing if q is empty.
- Else pop an item off the q and send it to the output stream.

On the LeJOS side



lejosJasonTest.java

```
import lejos.nxt.LCD;
import lejos.nxt.SensorPort;
import lejos.nxt.UltrasonicSensor;
import lejos.nxt.comm.Bluetooth;
import lejos.nxt.comm.NXTConnection;

public class lejosJasonTest {
  static UltrasonicSensor us;
  static NXTConnection conn;
  static BTSend sender;
  static Thread senderThread;
```

- Make use of `BTSend` to communicate over the `NXTConnection`

lejosJasonTest.java

```
public static void main(String[] args) throws Exception{
  us= new UltrasonicSensor(SensorPort.S3);
  int distance;

  System.out.println("Waiting");
  conn = Bluetooth.waitForConnection();
  System.out.println("Connected");

  sender = new BTSend(conn);
  senderThread = new Thread(sender);
  senderThread.setDaemon(true);
  senderThread.start();

  while(true){
    distance = us.getDistance();
    LCD.clear();
    LCD.drawString(distance+"", 0, 0);
    sender.write(distance+"");
    Thread.sleep(3000);
    }
  }
}
```

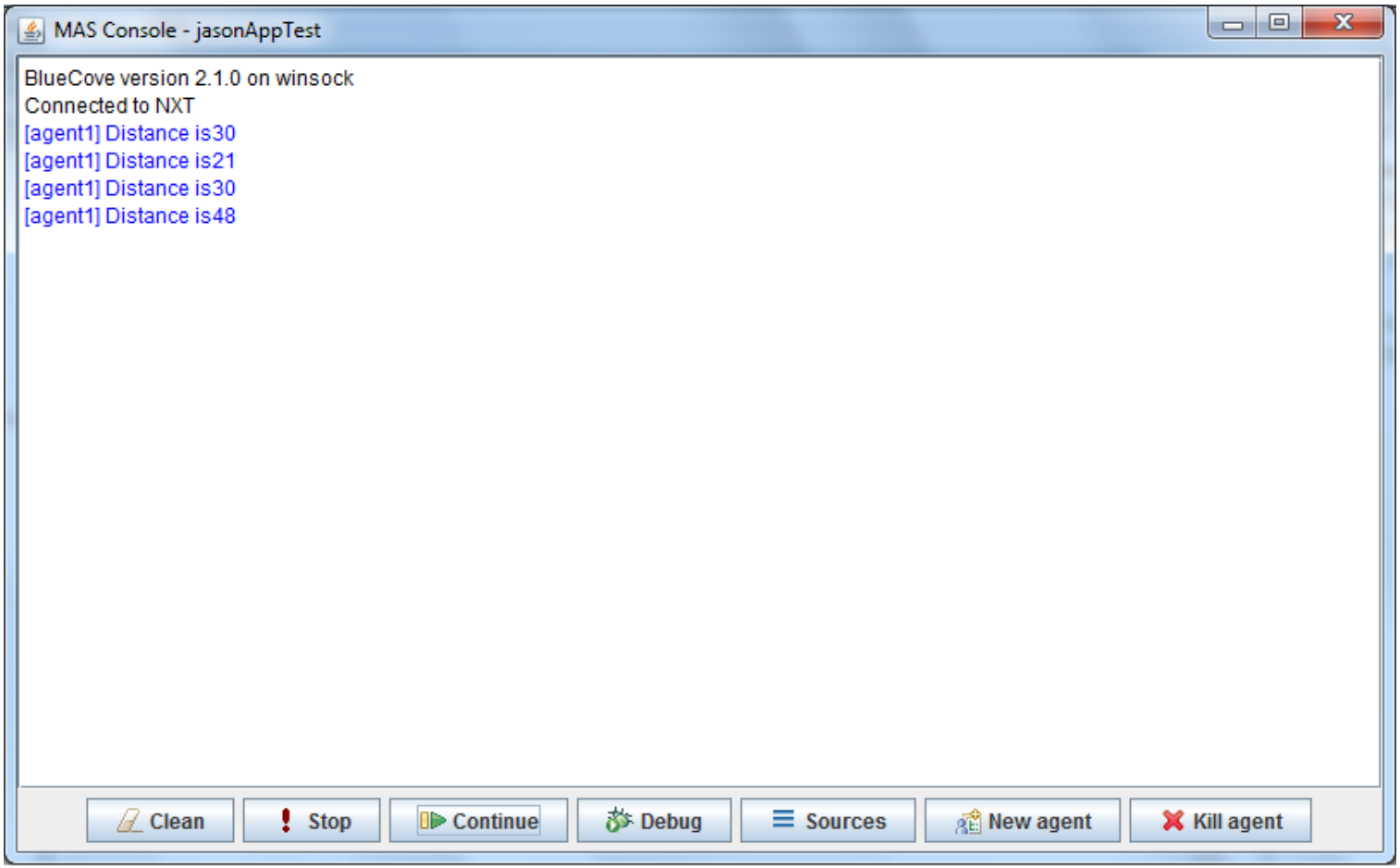
lejosJasonTest.java

- Open a Bluetooth connection.
- Use this to instantiate the `BTSend` object (which gives communication over the link)...
- ... and kick that communication off in its own thread.
- (That is what the `start` does.)
- Then, every three seconds, read the ultrasound sensor and send the result over the connection.

Next

- Now we can run it.

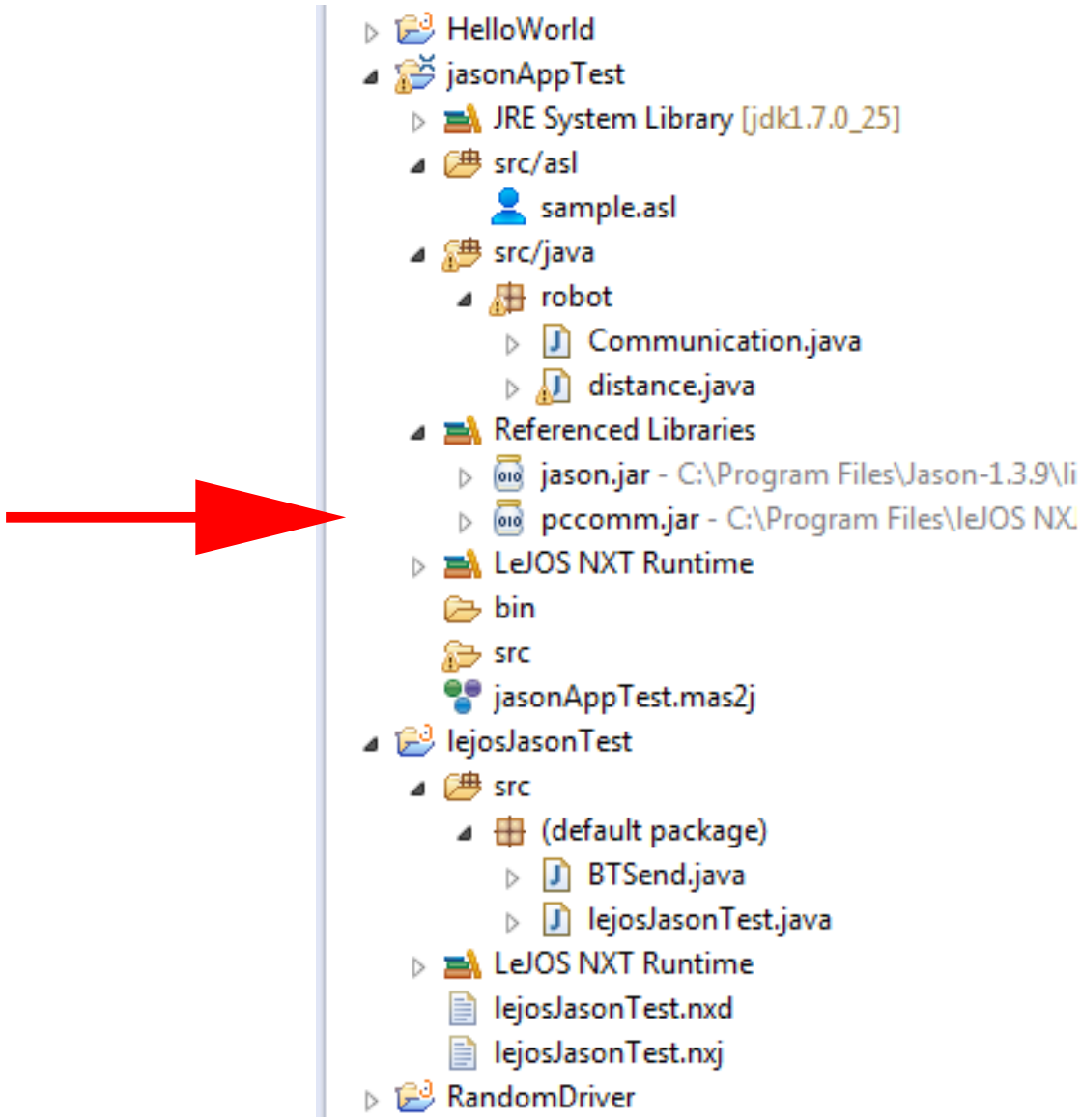
This is what you should see



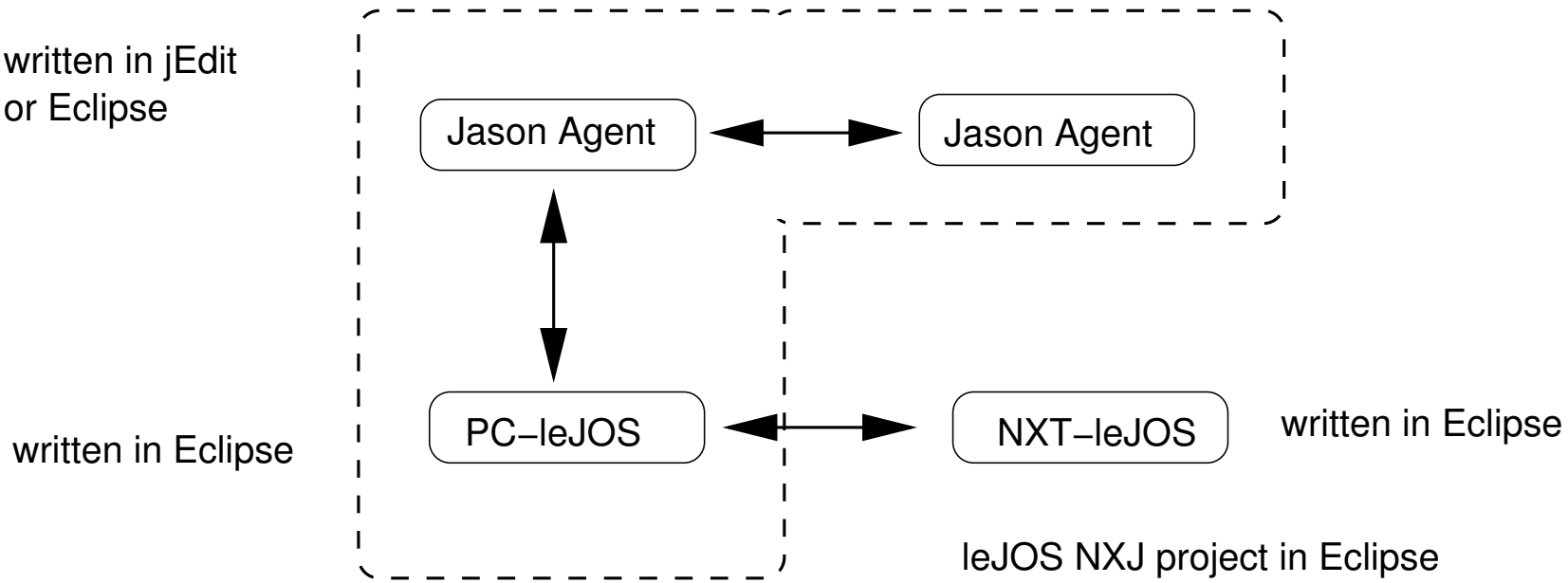
Note

- Even with these notes you will find it tricky getting things set up.

Need to include this



How the second assignment might look



Jason Project in Eclipse containing a package with PC LeJOS code. The project loads classes for PC LeJOS

Summary

- This lecture looked at interfacing LeJOS to Jason.
- This allows a Jason agent to execute commands that have an effect on the NXT.
- (Note that this is not what happens in the example.)
- This general structure will likely be helpful in Assignment 2.