@DroolsSession("classpath:/org/droolsassert/logicalEvents.drl")

**public** **class** LogicalEventsTest {

@Rule

**public** DroolsAssert drools = **new** DroolsAssert();

@Before

**public** **void** **before**() {

drools.setGlobal("stdout", System.*out*);

}

@Test

**public** **void** **testCallsConnectAndDisconnectLogic**() {

Dialing caller1Dial = **new** Dialing("11111", "22222");

drools.insertAndFire(caller1Dial);

drools.assertRetracted(caller1Dial);

CallInProgress call = drools.getObject(CallInProgress.**class**);

*assertEquals*("11111", call.callerNumber);

drools.advanceTime(5, *MINUTES*);

Dialing caller3Dial = **new** Dialing("33333", "22222");

drools.insertAndFire(caller3Dial);

drools.assertExist(caller3Dial);

drools.advanceTime(5, *SECONDS*);

drools.assertExist(call, caller3Dial);

drools.advanceTime(5, *SECONDS*);

drools.assertExist(call);

drools.assertRetracted(caller3Dial);

drools.advanceTime(1, *HOURS*);

drools.assertRetracted(call);

drools.assertAllRetracted();

}

@Test

**public** **void** **testCallsConnectAndDisconnectLogicStickToEvents**() {

Dialing caller1Dial = **new** Dialing("11111", "22222");

drools.insertAndFire(caller1Dial);

drools.assertActivated("input call");

drools.assertRetracted(caller1Dial);

CallInProgress call = drools.getObject(CallInProgress.**class**);

*assertEquals*("11111", call.callerNumber);

Dialing caller3Dial = **new** Dialing("33333", "22222");

drools.insertAndFire(caller3Dial);

drools.assertActivated();

drools.assertExist(call, caller3Dial);

drools.awaitFor("drop dial-up if callee is talking");

drools.assertActivated("drop dial-up if callee is talking", "input call dropped");

drools.assertExist(call);

drools.assertRetracted(caller3Dial);

drools.awaitFor("drop the call if caller is talking more than permitted time");

drools.assertActivatedCount(

1, "drop the call if caller is talking more than permitted time",

1, "call in progress dropped");

drools.assertRetracted(call);

drools.assertNoScheduledActivations();

drools.assertAllRetracted();

}

@Test

@TestRules(expected = "input call")

**public** **void** **testAssertActivations**() {

drools.insertAndFire(**new** Dialing("11111", "22222"));

}

@Test

@TestRules(expected = {

"input call",

"drop the call if caller is talking more than permitted time",

"call in progress dropped"

}, checkScheduled = **true**)

**public** **void** **testAssertScheduledActivations**() {

drools.insertAndFire(**new** Dialing("11111", "22222"));

}

Logical events story

Given imports

org.droolsassert

org.droolsassert.LogicalEventsTest

Given drools session classpath:/org/droolsassert/logicalEvents.drl

Given global stdout is System.out

!-- test calls connect and disconnect logic

Given new session for scenario

Given variable caller1Dial is new Dialing('11111', '22222')

When insert and fire caller1Dial

Then retracted caller1Dial

Given variable call as CallInProgress object from the session

Then assert call.callerNumber is '11111'

When advance time for 5 minutes

Given variable caller3Dial as new Dialing('33333', '22222')

When insert and fire caller3Dial

Then exist caller3Dial

When advance time for 5 seconds

Then exist call, caller3Dial

When advance time for 5 seconds

Then exist call

Then retracted caller3Dial

When advance time for 1 hour

Then retracted call

Then retracted all facts

!-- test calls connect and disconnect logic stick to events

Given new session for scenario

Given variable caller1Dial as new Dialing('11111', '22222')

When insert and fire caller1Dial

Then activated input call

Then retracted caller1Dial

Given variable call as CallInProgress object from the session

Then assert call.callerNumber equals '11111'

Given variable caller3Dial as new Dialing('33333', '22222')

When insert and fire caller3Dial

Then activated no rules

Then exist call, caller3Dial

When await for 'drop dial-up if callee is talking'

Then activated 'drop dial-up if callee is talking', 'input call dropped'

Then exist call

Then retracted caller3Dial

When await for 'drop the call if caller is talking more than permitted time'

Then count of activated are

1 drop the call if caller is talking more than permitted time

1 call in progress dropped

Then retracted call

Then there are no scheduled activations

Then retracted all facts

!-- test assert activations

Given new session for scenario

Given variable dial as Dialing from yaml {

callerNumber: '11111',

calleeNumber: '22222'

}

When insert and fire dial

Then all activations are 'input call'

!-- test assert scheduled activations

Given new session for scenario

Given variable dial as new Dialing('11111', '22222')

When insert and fire dial

Then all activations and scheduled are

input call

drop the call if caller is talking more than permitted time

call in progress dropped