



The Moses Tool Suite Simple Simulation Components

Kim Mason, Jörn W. Janneck

Overview



- ◆ Simulation architecture (recap)
 - Model execution
 - Communication
 - Component signature

- ◆ Examples
 - Preliminaries
 - » `hades.stat.Distribution`
 - `moses.basic.Clock`
 - `moses.basic.Delay`
 - » `moses.basic.OutputWriterProcessor`

- ◆ Exercise
 - Averager

Simulation architecture

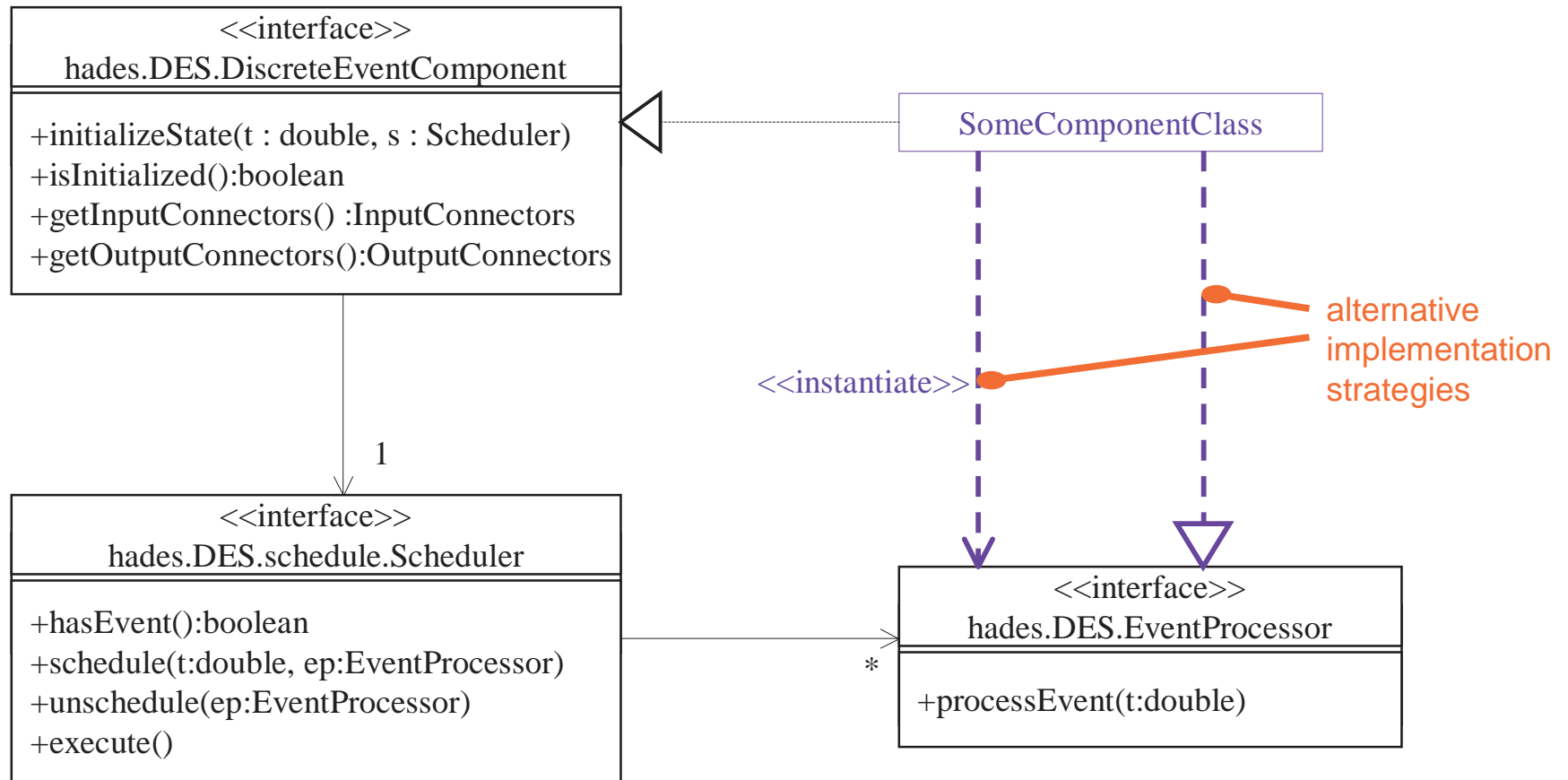
Model execution



- ◆ **DiscreteEventComponents**
 - have (named) inputs and outputs
 - » which are MessageListeners and MessageProducers, respectively
 - » can have a signature (which can be reflected)
 - have a state
 - are initialized and passed a scheduler
- ◆ **EventProcessors**
 - execute events and thus effect state change
 - can be scheduled exactly once
- ◆ **Scheduler**
 - arrange a number of EventProcessors according to their time stamp
 - initiate execution of the first EventProcessor

Simulation architecture

Model execution



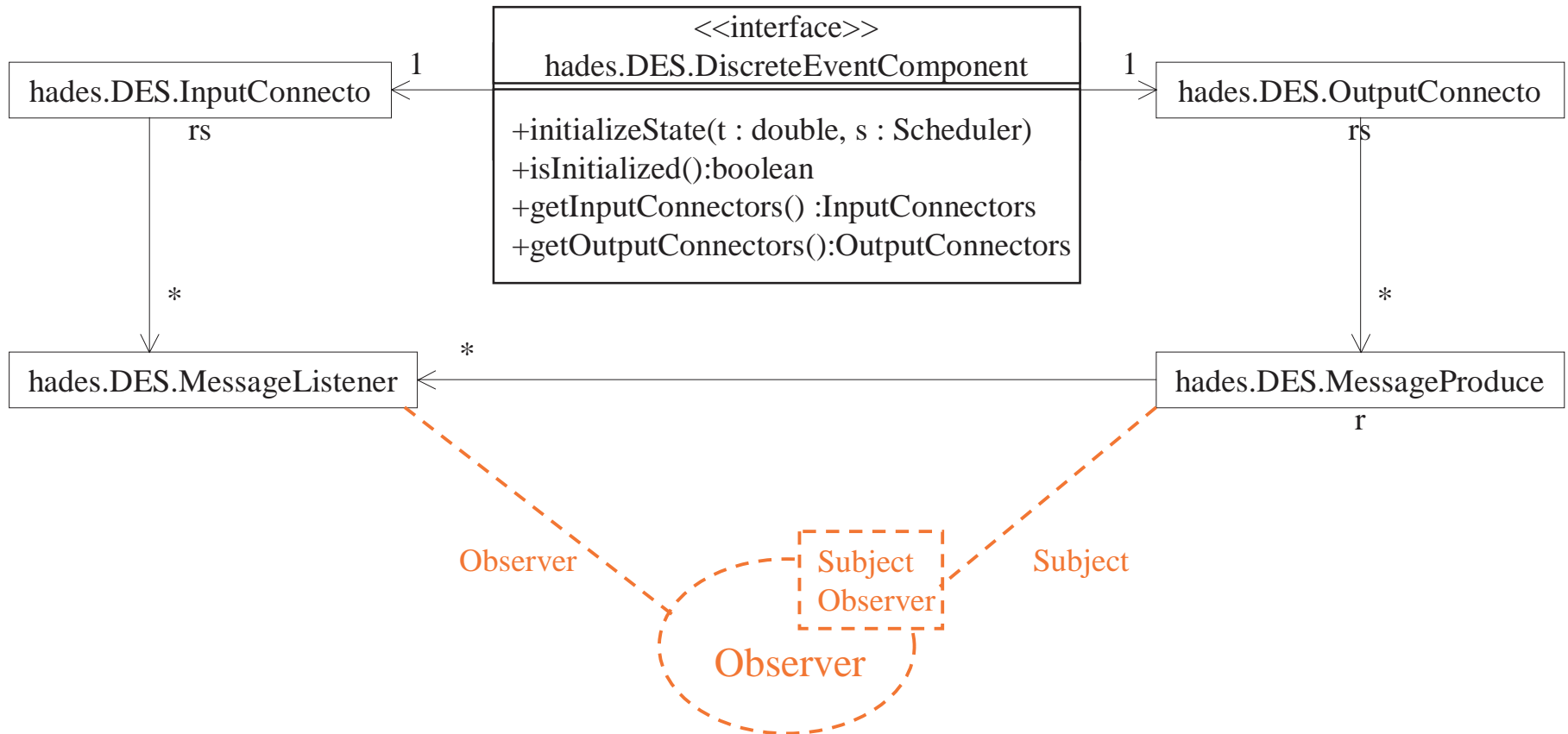


◆ Communication

- by passing MessageEvents between MessageProducers and MessageListeners
 - » events are time stamped
- producers and listeners communicate thru an observer-protocol
 - » which allows m:n-relations between them

Simulation architecture

Communication



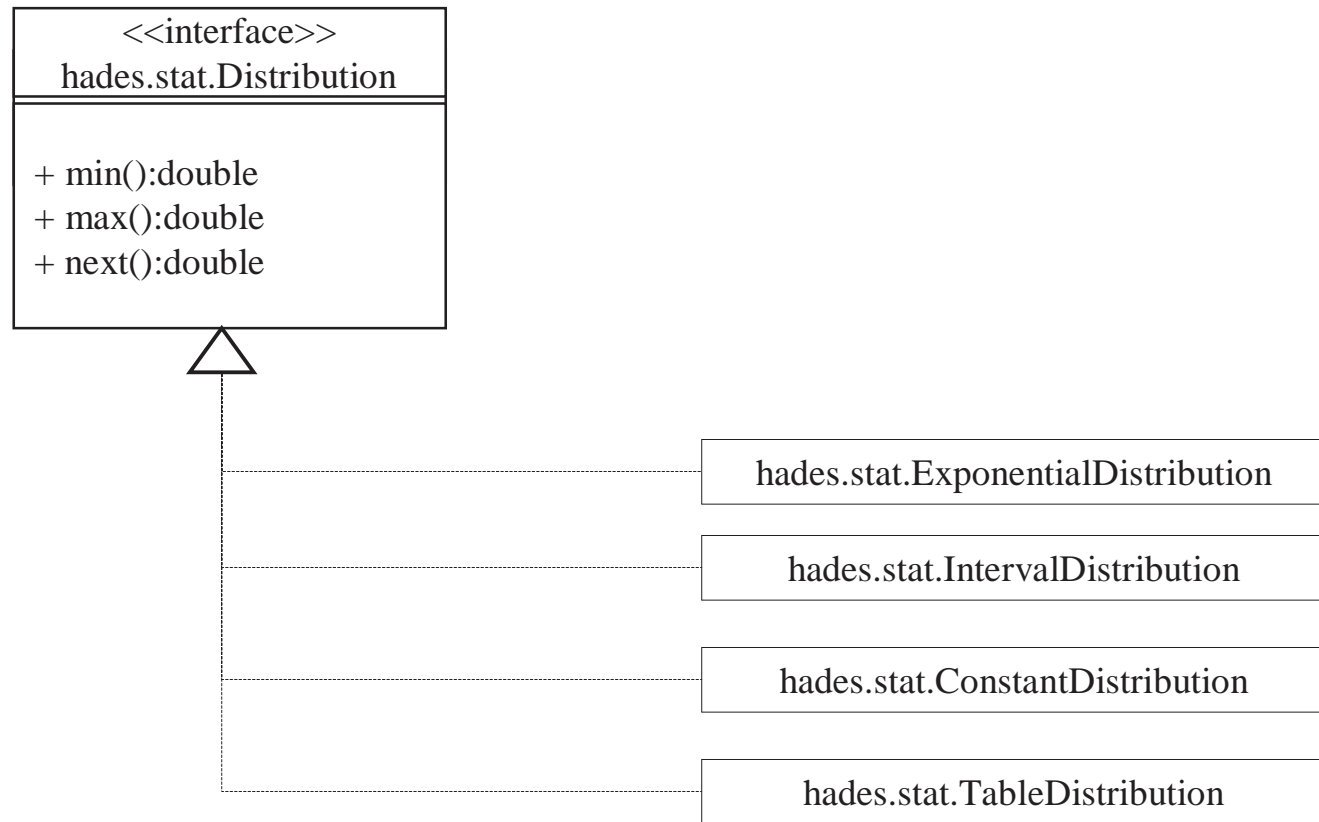
Component signature



- ◆ component description

- needed to extract I/O signature etc. at design time, without instantiating a component
- retrieved using reflection
- `hades.DES.components.ComponentDescriptor`

Preliminaries



Preliminaries



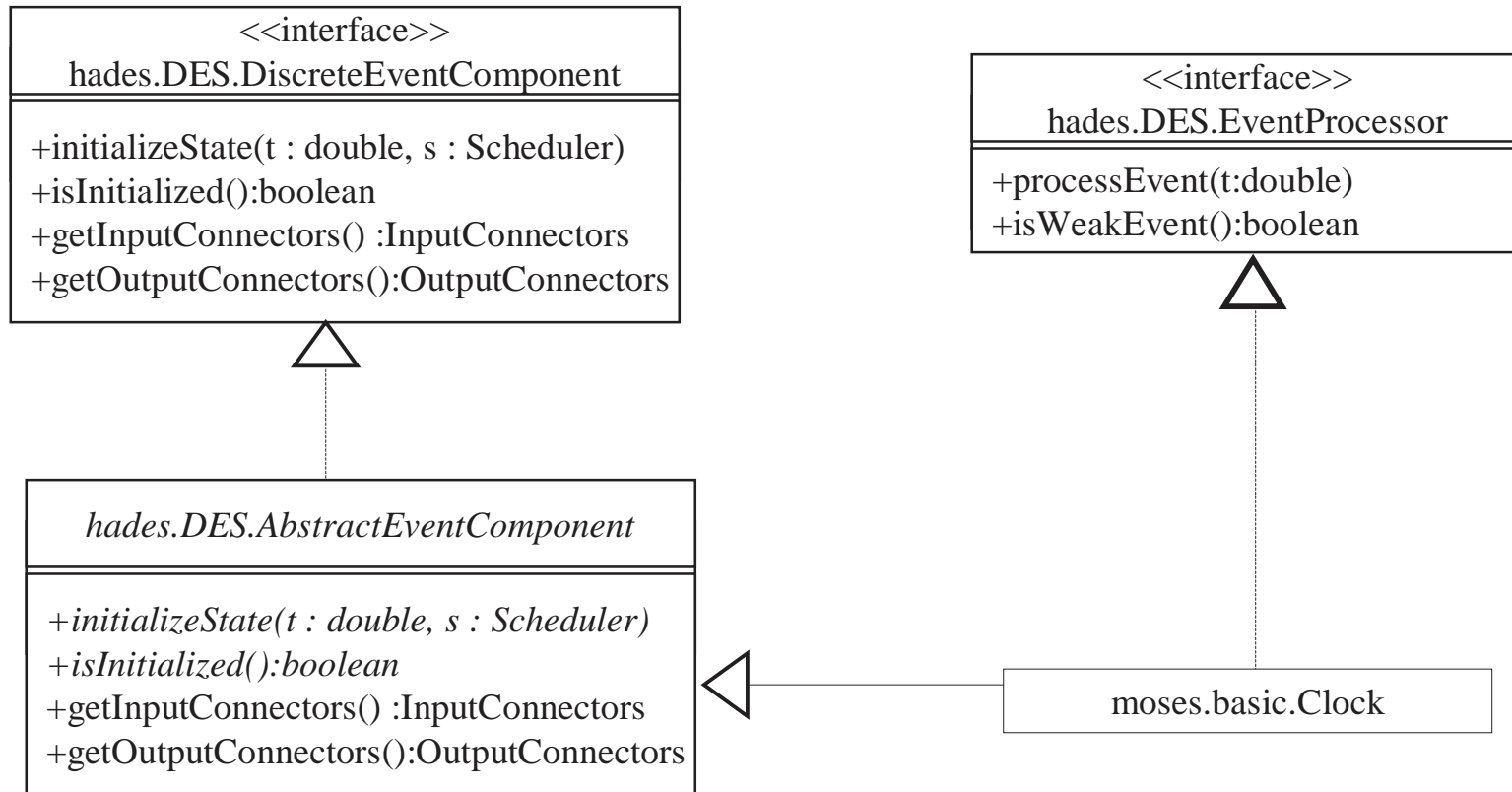
```
public class MessageEvent
extends java.util.EventObject
implements java.io.Serializable {

    public double    time;
    public Object    value;

    public MessageEvent(Object source,
                        double tm, Object val) {
        super(source);
        time = tm;
        value = val;
    }
}
```

Example

moses.basic.Clock



Local attributes



```
private Distribution distr;  
  
private Scheduler scheduler;  
  
private MessageProducer out;
```

Model execution - EventProcessor



```
public boolean    processEvent(double tm) {  
  
    out.notifyMessage(new MessageEvent(this, tm, null));  
  
    scheduler.schedule(tm + distr.next(), this);  
    return true;  
}  
  
public boolean    isWeakEvent() { return false; }
```

Execution - DEC



```
public void initializeState(double t, Scheduler s) {
    scheduler = s;
    scheduler.schedule(t, this);
}

public boolean isInitialized() {
    return scheduler != null;
}
```

moses.basic.Clock Constructor



```
public Clock(Distribution distr) {  
    this.distr = distr;  
  
    out = new BasicMessageProducer();  
    outputs.addConnector("Clk", out);  
}
```

this attribute is protected in
`hades.DES.AbstractDiscreteEventComponent`

Signature



```
public Object outputClk() { return null; }
```

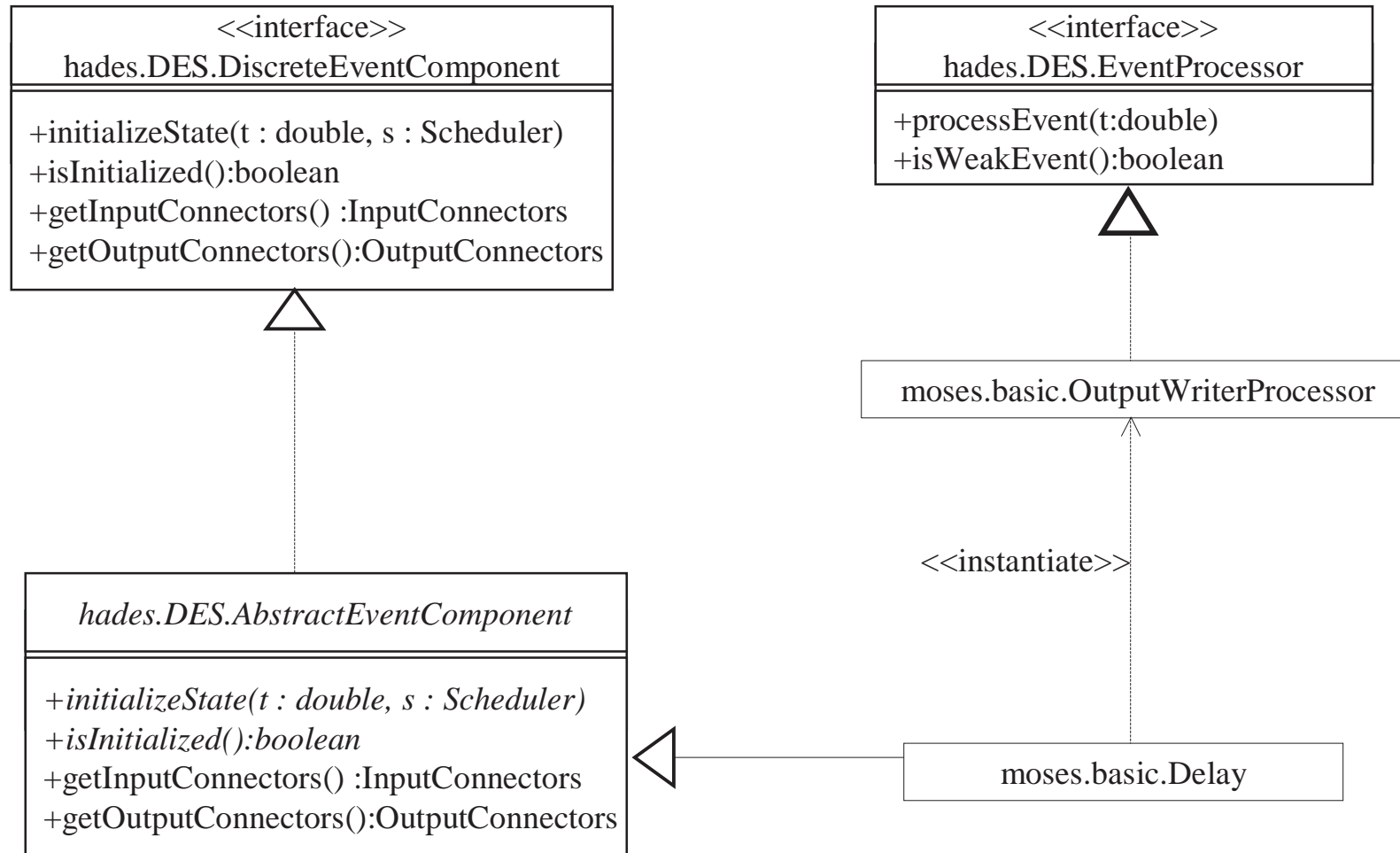
token class

connector name
(prefixed with 'output')

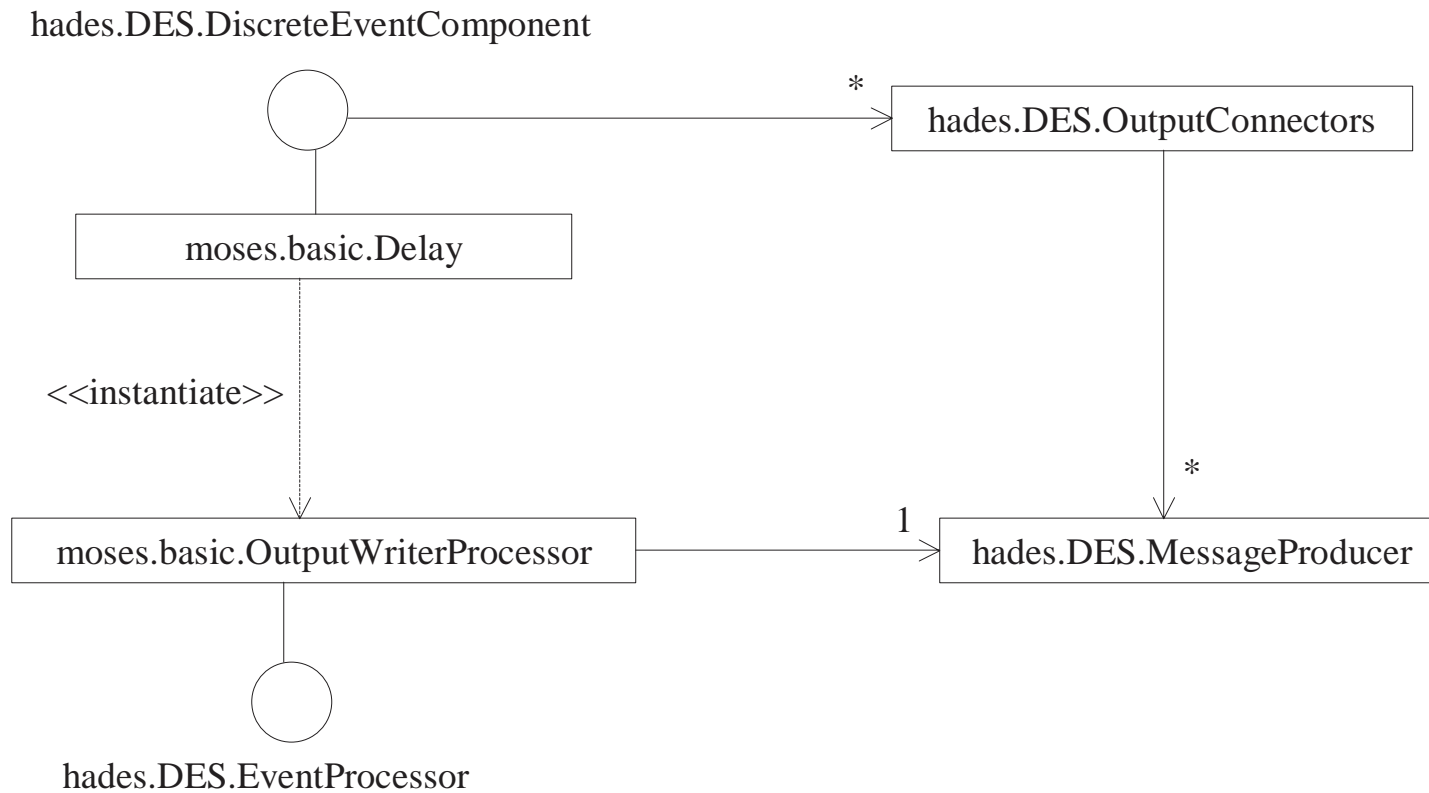
NB: This method doesn't do anything, and isn't supposed to.
It's never called - it sits there only to be reflected.

Example

moses.basic.Delay



moses.basic.Delay OutputWriterProcessor



OutputWriterProcessor



```
private Object          src;
private MessageProducer out;
private Object          val;

//
//  EventProcessor
//

public boolean processEvent(double tm) {
    out.notifyMessage(new MessageEvent(src, tm, val));
    return true;
}
```

Local attributes



```
private Distribution distr;  
  
private Scheduler scheduler;  
  
private MessageProducer out;
```

Execution - DEC



```
public void initializeState(double t, Scheduler s) {
    scheduler = s;
}

public boolean isInitialized() {
    return scheduler != null;
}
```

moses.basic.Delay

Handling input



```
class Input extends AbstractMessageListener {  
  
    public void message(MessageEvent evt) {  
        scheduler.schedule(evt.time + distr.next(),  
            new OutputWriterProcessor(this,  
                out, evt.value));  
    }  
}
```

Here the event processor is created.

NB: This is an inner class to moses.basic.Delay. It has therefore access to its private fields

moses.basic.Delay Constructor



```
public Delay(Distribution d) {  
    this.distr = d;  
  
    inputs.addConnector("A", new Input());  
  
    out = new BasicMessageProducer();  
    outputs.addConnector("B", out);  
}
```

these attributes are protected in
hades.DES.AbstractDiscreteEventComponent

Signature



```
public Object inputA() { return null; }  
public Object outputB() { return null; }
```