An example with basic rules

An identifier for the monitor

Declaration of any variables used

A rule: Event \\ Condition –X Action

–X signifies that if the rule fires, the assertion has failed

```
Rule.create("withdrawLimit",
      new String[] {"double amount"},
      new String[] {"withdraw(amount)\\amount>1000 -X
      System.out.println(\"Withdraw limit exceeded: \"+ amount);"});
```

An example with finite state machines

Import any relevant classes

For each unique instance of this variable, a copy of this assertion is spawned

start always signifies the initial state

```
FSM.create("accountLifeCycle",
      "import tutorial._9_fsm.accounting.*;",
      "Account acc",
      "start    [acc.new()]                    new",
      "new      [deleteAccount(acc)]           end",
      "start    [deleteAccount(acc)]           bad",
      "start    [acc.deposit]                  bad",
      "start    [acc.withdraw]                 bad",
      "new      [acc.new()]                    bad");
```

A transition (similar to a rule): Event \\ Condition >> Action [this example shows only events]

bad signifies the assertion has failed

An example with regular expression

The regular expression can either be to match correct or incorrect behaviour

Symbol definition in terms of events

The regular expression itself

```
RE.create("regular_expressions",
      "import tutorial._6_rules.accounting.*;",
      "Account acc",
      RE.Matching.EXPECTED_BEHAVIOUR,
      new String[] {
      "n = acc.new()",
      "x = deleteAccount(acc)",
      "d = acc.deposit",
      "w = acc.withdraw"},
      "n(d|w)*x");
```