Computational Tree Logic

EECS 4315

www.eecs.yorku.ca/course/4315/

Syntax

The *state formulas* are defined by

$$f ::= a \mid f \wedge f \mid \neg f \mid \exists g \mid \forall g$$

The *path formulas* are defined by

$$g ::= \bigcirc f \mid f \,\mathsf{U}\, f$$

Syntax

The *state formulas* are defined by

$$f ::= a \mid f \wedge f \mid \neg f \mid \exists g \mid \forall g$$

The *path formulas* are defined by

$$g ::= \bigcirc f \mid f \,\mathsf{U}\, f$$

The *formulas* are defined by

$$f ::= a \mid f \wedge f \mid \neg f \mid \exists \bigcirc f \mid \exists (f \,\mathsf{U}\, f) \mid \forall \bigcirc f \mid \forall (f \,\mathsf{U}\, f)$$

Semantics of CTL

$$
\begin{array}{rcl}
s \models a & \text{iff} & a \in \ell(s) \\
s \models f_1 \wedge f_2 & \text{iff} & s \models f_1 \wedge s \models f_2 \\
s \models \neg f & \text{iff} & s \not\models f \\
s \models \exists g & \text{iff} & \exists p \in \mathit{Paths}(s) : p \models g \\
s \models \forall g & \text{iff} & \forall p \in \mathit{Paths}(s) : p \models g
\end{array}
$$

and

$$
\begin{array}{rcl}
p \models \bigcirc f & \text{iff} & p[1] \models f \\
p \models f_1 \,\mathsf{U}\, f_2 & \text{iff} & \exists i \geq 0 : p[i] \models f_2 \wedge \forall 0 \leq j < i : p[j] \models f_1
\end{array}
$$

Model checking CTL

Problem

Given a transition system TS and a CTL formula f, check whether $TS \models f$.

Model checking CTL

Problem

Given a transition system TS and a CTL formula f, check whether $TS \models f$.

Definition

The *satisfaction set* $Sat(f)$ is defined by

$$Sat(f) = \{ s \in S \mid s \models f \}.$$

Model checking CTL

Problem

Given a transition system TS and a CTL formula f, check whether $TS \models f$.

Definition

The *satisfaction set* $Sat(f)$ is defined by

$$Sat(f) = \{ s \in S \mid s \models f \}.$$

Basic idea

Compute $Sat(f)$ by recursion on the structure of f.

Model checking CTL

Problem

Given a transition system TS and a CTL formula f, check whether $TS \models f$.

Definition

The *satisfaction set* $Sat(f)$ is defined by

$$Sat(f) = \{ s \in S \mid s \models f \}.$$

Basic idea

Compute $Sat(f)$ by recursion on the structure of f.

$TS \models f$ iff $I \subseteq Sat(f)$.

Model checking CTL

Problem

Given a transition system TS and a CTL formula f, check whether $TS \models f$.

Definition

The *satisfaction set* $Sat(f)$ is defined by

$$Sat(f) = \{ s \in S \mid s \models f \}.$$

Basic idea

Compute $Sat(f)$ by recursion on the structure of f.

$TS \models f$ iff $I \subseteq Sat(f)$.

Alternative view

Label each state with the subformulas of f that it satisfies.

Model checking CTL

The *formulas* are defined by

$$f ::= a \mid f \wedge f \mid \neg f \mid \exists \bigcirc f \mid \forall \bigcirc f \mid \exists(f \,\mathsf{U}\, f) \mid \forall(f \,\mathsf{U}\, f)$$

Question

What is $Sat(\forall(f \,\mathsf{U}\, g))$?

Model checking CTL

$s \in Sat(\forall(f \,\mathsf{U}\, g))$

iff $s \models \forall(f \,\mathsf{U}\, g)$

iff $\forall p \in Paths(s) : p \models f \,\mathsf{U}\, g$

iff $\forall p \in Paths(s) : \exists i \geq 0 : p[i] \models g \wedge \forall 0 \leq j < i : p[j] \models f$

iff $\forall p \in Paths(s) : p[0] \models g \vee (\exists i \geq 1 : p[i] \models g \wedge \forall 0 \leq j < i : p[j] \models f)$

iff $\forall p \in Paths(s) : p[0] \models g \vee$

$(p[0] \models f \wedge \exists i \geq 1 : p[i] \models g \wedge \forall 1 \leq j < i : p[j] \models f)$

iff $s \models g \vee (s \models f \wedge \forall s \to t : t \models \forall(f \,\mathsf{U}\, g))$

iff $s \in Sat(g) \vee (s \in Sat(f) \wedge \forall t \in succ(s) : t \in Sat(\forall(f \,\mathsf{U}\, g)))$

iff $s \in Sat(g) \cup \{\, s \in Sat(f) \mid succ(s) \subseteq Sat(\forall(f \,\mathsf{U}\, g)) \,\}$

Model checking CTL

As we have seen

$$s \in Sat(\forall(f \ \mathsf{U} \ g))$$
$$\text{iff } s \in Sat(g) \cup \{\, s \in Sat(f) \mid succ(s) \subseteq Sat(\forall(f \ \mathsf{U} \ g)) \,\}$$

Model checking CTL

As we have seen

$$s \in Sat(\forall(f \ \mathsf{U} \ g))$$
$$\text{iff } s \in Sat(g) \cup \{\, s \in Sat(f) \mid succ(s) \subseteq Sat(\forall(f \ \mathsf{U} \ g)) \,\}$$

Hence, the set $Sat(\forall(f \ \mathsf{U} \ g))$ is a subset T of S such that

$$T = Sat(g) \cup \{\, s \in Sat(f) \mid succ(s) \subseteq T \,\}$$

Model checking CTL

As we have seen

$$s \in Sat(\forall(f \,\mathsf{U}\, g))$$
$$\text{iff } s \in Sat(g) \cup \{\, s \in Sat(f) \mid succ(s) \subseteq Sat(\forall(f \,\mathsf{U}\, g)) \,\}$$

Hence, the set $Sat(\forall(f \,\mathsf{U}\, g))$ is a subset T of S such that

$$T = Sat(g) \cup \{\, s \in Sat(f) \mid succ(s) \subseteq T \,\}$$

Proposition

The set $Sat(\forall(f \,\mathsf{U}\, g))$ is the smallest subset T of S such that

$$T = Sat(g) \cup \{\, s \in Sat(f) \mid succ(s) \subseteq T \,\}$$

Model checking CTL

```
Sat(f):
switch (f) {
  case a :          return { s ∈ S | a ∈ ℓ(s) }
  case f ∧ g :      return Sat(f) ∩ Sat(g)
  case ¬f :         return S \ Sat(f)
  case ∃◯ f :       return { s ∈ S | succ(s) ∩ Sat(f) ≠ ∅ }
  case ∀◯ f :       return { s ∈ S | succ(s) ⊆ Sat(f) }
  case ∃(f U g) :   T = ∅
                    while T ≠ F(T)
                      T = F(T)
                    return T
  case ∀(f U g) :   T = ∅
                    while T ≠ G(T)
                      T = G(T)
                    return T
}
```

Size of a CTL formula

$$
\begin{aligned}
|a| &= 1 \\
|f \wedge g| &= 1 + |f| + |g| \\
|\neg f| &= 1 + |f| \\
|\exists \bigcirc f| &= 1 + |f| \\
|\exists (f \,\mathsf{U}\, g)| &= 1 + |f| + |g| \\
|\forall \bigcirc f| &= 1 + |f| \\
|\forall (f \,\mathsf{U}\, g)| &= 1 + |f| + |g|
\end{aligned}
$$

The complexity of CTL model checking

By improving the model checking algorithm (see, for example, the textbook of Baier and Katoen for details), we obtain

Theorem

For a transition system TS, with N states and K transitions, and a CTL formula f, the model checking problem $TS \models f$ can be decided in time $O((N + K)|f|)$.

The complexity of CTL model checking

By improving the model checking algorithm (see, for example, the textbook of Baier and Katoen for details), we obtain

Theorem

For a transition system TS, with N states and K transitions, and a CTL formula f, the model checking problem $TS \models f$ can be decided in time $O((N + K)|f|)$.

Theorem

For a transition system TS, with N states and K transitions, and a LTL formula g, the model checking problem $TS \models f$ can be decided in time $O((N + K)2^{|g|})$.

The complexity of CTL model checking

By improving the model checking algorithm (see, for example, the textbook of Baier and Katoen for details), we obtain

Theorem

For a transition system TS, with N states and K transitions, and a CTL formula f, the model checking problem $TS \models f$ can be decided in time $O((N + K)|f|)$.

Theorem

For a transition system TS, with N states and K transitions, and a LTL formula g, the model checking problem $TS \models f$ can be decided in time $O((N + K)2^{|g|})$.

Theorem

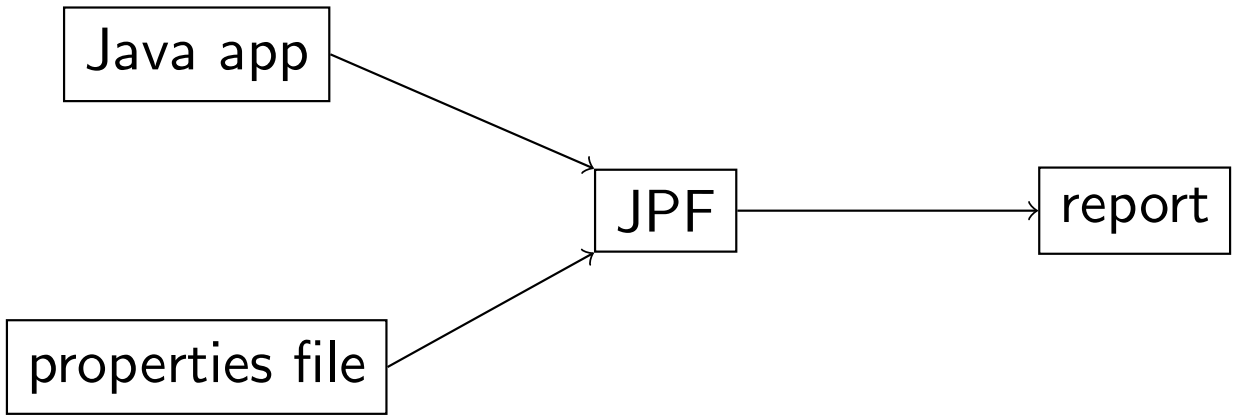
If P $\neq$ NP then there exist LTL formulas g_n whose size is a polynomial in n, for which equivalent CTL formulas exist, but not of size polynomial in n.

Testing JPF

EECS 4315

www.eecs.yorku.ca/course/4315/

Java PathFinder (JPF)



Testing JPF

Question

In order to test a part of JPF, say a listener, how many pieces of input do we need to provide and what type of input?

Testing JPF

Question

In order to test a part of JPF, say a listener, how many pieces of input do we need to provide and what type of input?

Answer

Two: a Java app and an application properties file.

Failed assertions

We want to test whether JPF can detect failed assertions.

Question

Write a Java app that can be used to test whether JPF can detect failed assertions.

Failed assertions

We want to test whether JPF can detect failed assertions.

Question

Write a Java app that can be used to test whether JPF can detect failed assertions.

Answer

```
public class FailedAssertion {
  public static void main(String[] args) {
    assert false;
  }
}
```

Failed assertions

We want to test whether JPF can detect failed assertions.

Question

Write the corresponding application properties file.

Failed assertions

We want to test whether JPF can detect failed assertions.

Question

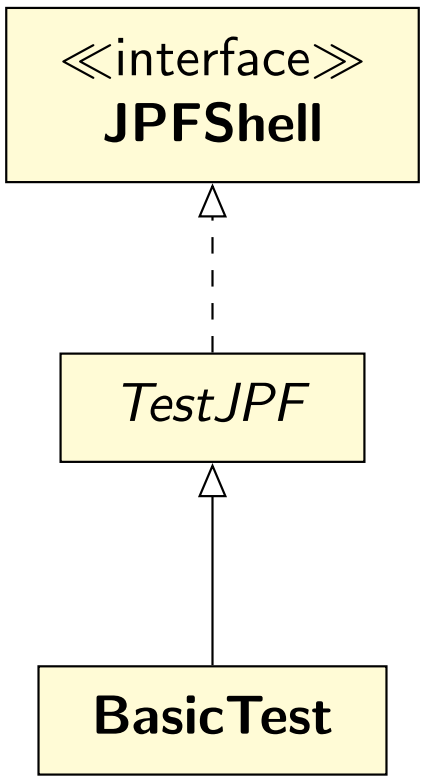
Write the corresponding application properties file.

Answer

```
target=FailedAssertion
classpath=.
```

JPF testing framework

JPF provides a framework very similar to JUnit so that the body of the `main` method of the Java app and the content of the application properties file can be combined into a single unit test.



Skeleton of a test case

```
import gov.nasa.jpf.util.test.TestJPF;
import org.junit.Test;

public class ...Test extends TestJPF {

  /**
   * Runs the test methods with the given names.
   * If no names are given, all test methods are run.
   *
   * @param testMethods the names of the test methods.
   */
  public static void main(String[] testMethods) {
    TestJPF.runTestsOfThisClass(testMethods);
  }
}
```

Skeleton of a unit test

```
@Test
public void test...() {
  if (this.verify...(application properties)) {
        // body of main method of app
  }
}
```

Example of a unit test

```
@Test
public void testAssert() {
  if (this.verifyAssertionError("+classpath=.")) {
    assert false;
  }
}
```

Example of a unit test

```
@Test
public void testAssert() {
  if (this.verifyAssertionError(    "+classpath=."    )) {
    assert false;
  }
}
```

application properties

Example of a unit test

```
@Test
public void testAssert() {
  if (this.verifyAssertionError("+classpath=.")) {
    assert false;
  }
}
```

body

Running a test case

To run the `BasicTest` from the command line, use

```
java -cp <path to>/jpf.jar:. BasicTest
```

It produces output similar to the following.

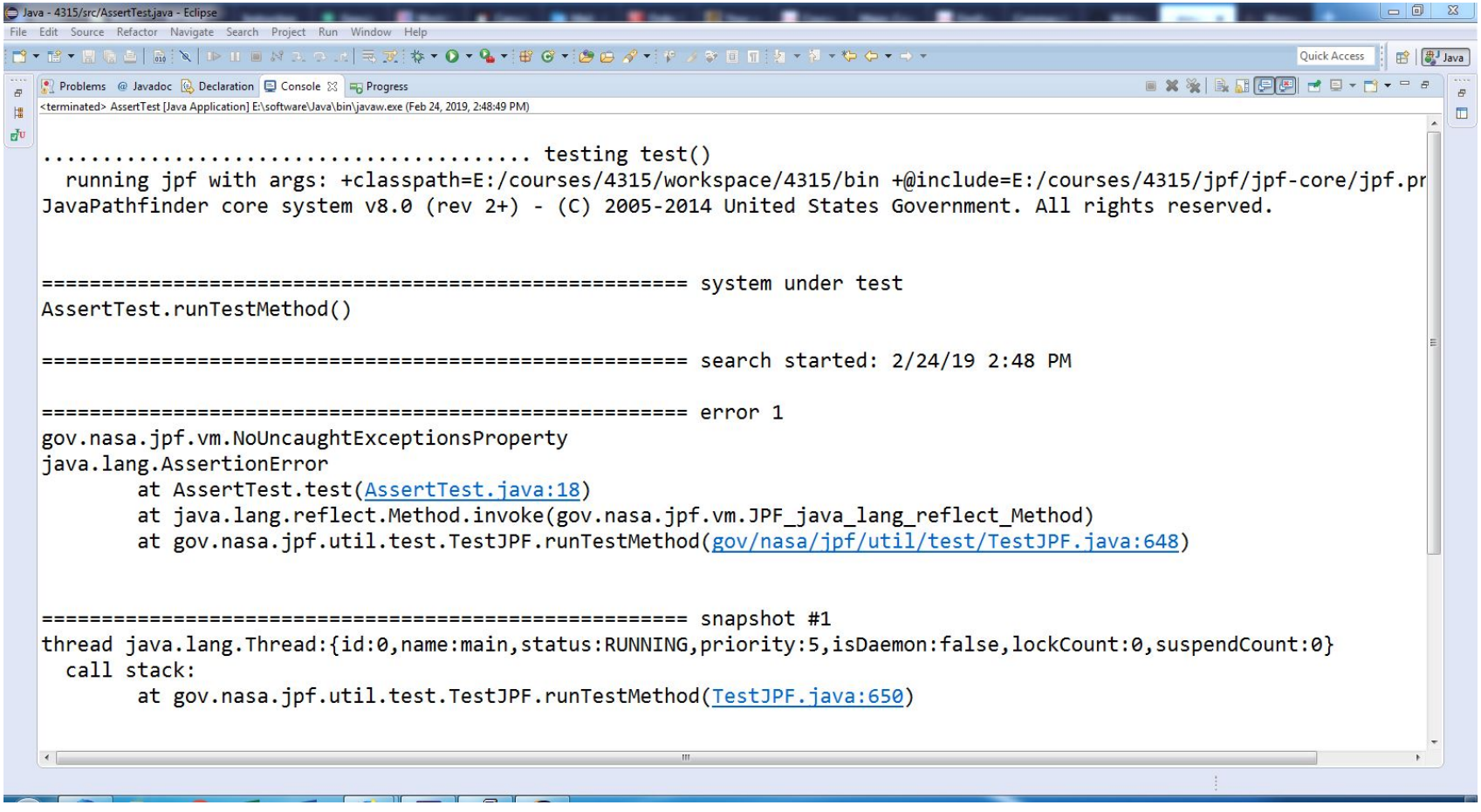
```
..................................... testing testAsser
running jpf with args: +classpath=.
JavaPathfinder core system v8.0 (rev ..) - (C) 2005-2014 Ur
...

============================================================ sear
....................... test: Ok

....................... execution of testsuite: AssertTest
.... [1] test: Ok
....................... tests: 1, failures: 0, errors: 0
```

Running a test case

The class can also be run as an app in eclipse. In that case, you may have to add

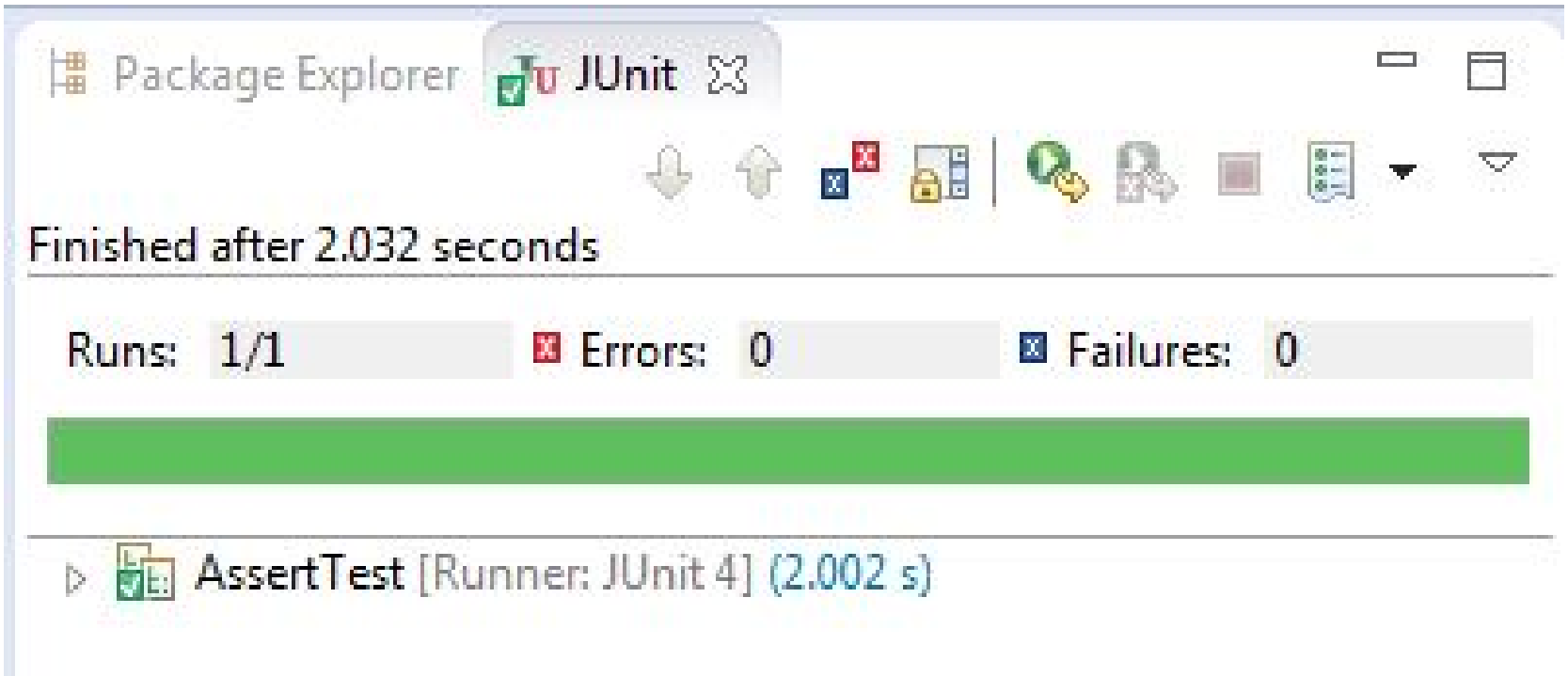
`"+@include=<path to>jpf/jpf-core/jpf.properties"`

as a second argument of the invocation of `verifyAssertionError`.



Running a test case

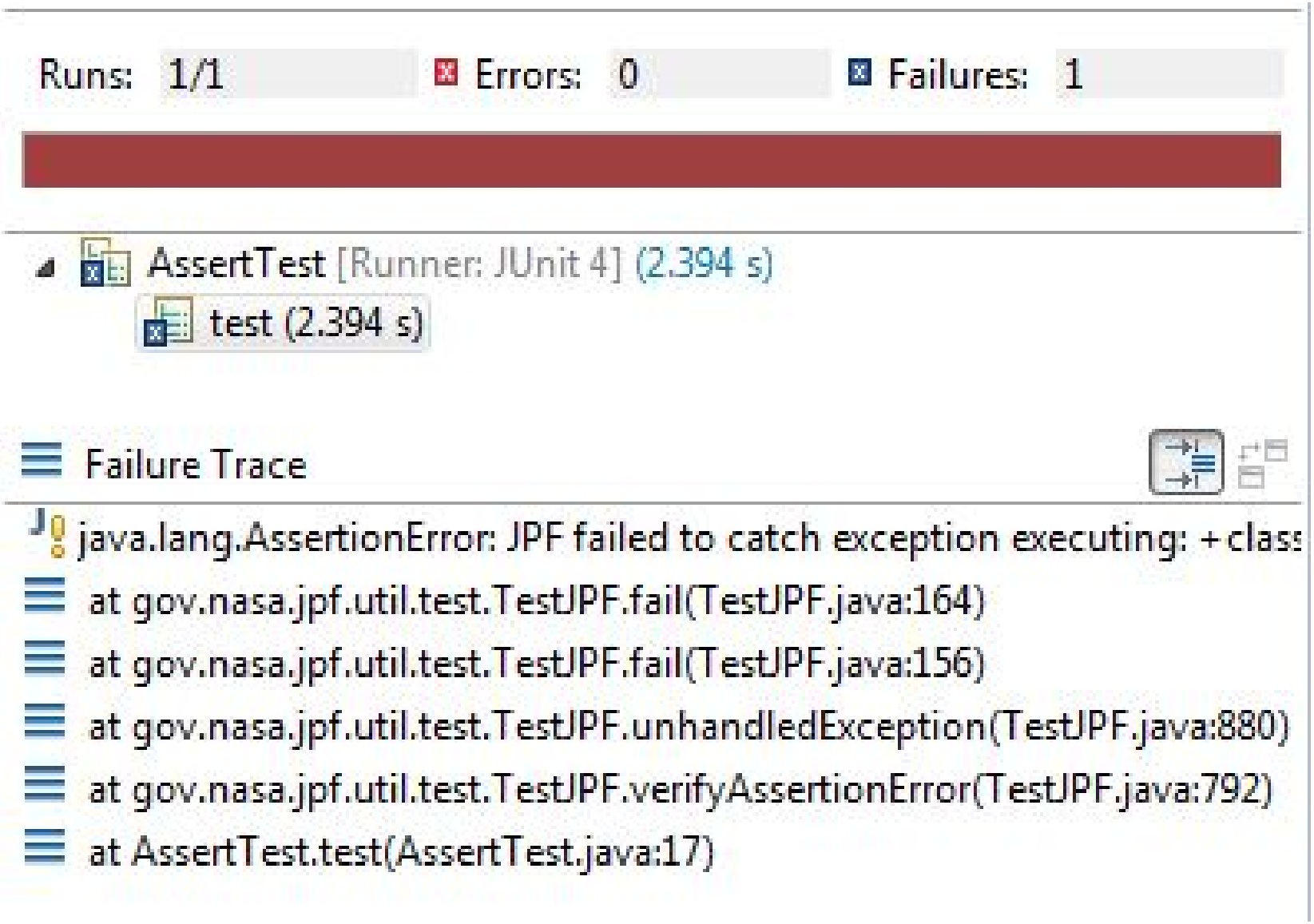
The class can also be run as a JUnit test in eclipse.



Example of a failing unit test

```
@Test
public void testAssert() {
  if (this.verifyAssertionError("+classpath=.")) {
    assert true;
  }
}
```

Example of a failing unit test



Unhandled exception

```
@Test
public void TestException() {
  if (this.verifyUnhandledException(
        "java.lang.NullPointerException",
        "+classpath=.")) {
    throw new NullPointerException();
  }
}
```

Unhandled exception

```
@Test
public void testException() {
  if (this.verifyUnhandledException(
        "java.lang.NullPointerException",
          type of unhandled exception
        "+classpath=.")) {
    throw new NullPointerException();
  }
}
```

Unhandled exception

```
@Test
public void testException() {
  if (this.verifyUnhandledException(
     "java.lang.NullPointerException",
       "+classpath=."    )) {
   application properties
    throw new NullPointerException();
  }
}
```

Unhandled exception

```
@Test
public void testException() {
  if (this.verifyUnhandledException(
    "java.lang.NullPointerException",
    "+classpath=.")) {
    throw new NullPointerException();
  }
}
```

body

Property violation

```
@Test
public void testPropertyViolation() {
  if (this.verifyPropertyViolation(
      new TypeRef("gov.nasa.jpf.vm.NoUncaughtExceptionsProperty")
      "+classpath=.")) {
    throw new RuntimeException();
  }
}
```

The class `TypeRef` is part of the package `gov.nasa.jpf.util`.

Property violation

```
@Test
public void testPropertyViolation() {
  if (this.verifyPropertyViolation(
        new TypeRef("gov.nasa.jpf.vm.NoUncaughtExceptionsProperty
                          type of violated property
        "+classpath=.")) {
    throw new RuntimeException();
  }
}
```

Property violation

```
@Test
public void testPropertyViolation() {
  if (this.verifyPropertyViolation(
      new typeRef("gov.nasa.jpf.vm.NoUncaughtExceptionsProperty")
       "+classpath=."    )) {
   application properties
    throw new RuntimeException();
  }
}
```

Property violation

```
@Test
public void testPropertyViolation() {
  if (this.verifyPropertyViolation(
      new typeRef("gov.nasa.jpf.vm.NoUncaughtExceptionsProperty")
      "+classpath=.")) {
    throw new RuntimeException();
  }
}
```

body

No property violation

```
@Test
public void testNoPropertyViolation() {
  if (this.verifyNoPropertyViolation("+classpath=.")) {
    // nothing
  }
}
```

No property violation

```
@Test
public void testNoPropertyViolation() {
  if (this.verifyNoPropertyViolation(
        "+classpath=."    )) {
     application properties
    // nothing
  }
}
```

No property violation

```
@Test
public void testNoPropertyViolation() {
  if (this.verifyNoPropertyViolation("+classpath=.")) {
    // nothing
```

body

```
  }
}
```

Testing the listener CallMonitor

The listener `gov.nasa.jpf.listener.CallMonitor` monitors method invocations. When JPF finishes, it publishes for each method invocation, the ID of the thread that executed the method invocation, the depth of the stack, the name of the class, the name of the method, and its arguments.

Testing the listener CallMonitor

The listener `gov.nasa.jpf.listener.CallMonitor` monitors method invocations. When JPF finishes, it publishes for each method invocation, the ID of the thread that executed the method invocation, the depth of the stack, the name of the class, the name of the method, and its arguments.

Problem

How to test the listener CallMonitor.

Sample app

```
public class Example {
  public static void main(String [] args) {
    first(1, true);
  }

  private static void first(int i, boolean b) {
    second(i + 1);
  }

  private static void second(int i) {
    // do nothing
  }
}
```

Sample application properties file

```
target=Example
classpath=.
listener=gov.nasa.jpf.listener.CallMonitor
```

Sample output

```
========================================== method invocations
0:                 java.lang.Boolean.<clinit>()
...
0:   Example.main([Ljava.lang.String;@bb)
0:    Example.first(1,true)
0:     Example.second(2)
```

Testing the listener CallMonitor

Question

What is the simplest app on which we can run the listener CallMonitor?

Testing the listener CallMonitor

Question

What is the simplest app on which we can run the listener CallMonitor?

Answer

An app with an empty main method.

Testing the listener CallMonitor

Question

To test the CallMonitor listener, which of the following `verify` methods can we use for the empty app?

- `verifyAssertionError`
- `verifyNoPropertyViolation`
- `verifyPropertyViolation`
- `verifyUnhandledException`

Testing the listener CallMonitor

Question

To test the CallMonitor listener, which of the following `verify` methods can we use for the empty app?

- `verifyAssertionError`
- `verifyNoPropertyViolation`
- `verifyPropertyViolation`
- `verifyUnhandledException`

Answer

`verifyNoPropertyViolation`

Testing the listener CallMonitor

Question

What is the corresponding application properties file?

Testing the listener CallMonitor

Question

What is the corresponding application properties file?

Answer

```
target=CallMonitorTest
classpath=.
listener=gov.nasa.jpf.listener.CallMonitor
```

Testing the listener CallMonitor

Question

Write the corresponding unit test.

Testing the listener CallMonitor

Question

Write the corresponding unit test.

Answer

```
@Test
public void testEmpty() {
  if (this.verifyNoPropertyViolation(
        "+listener=gov.nasa.jpf.listener.CallMonitor",
        "+classpath=.")) {
    // do nothing
  }
}
```

Testing the listener CallMonitor

Question

Consider the following method.

```
private static void staticMethod() {}
```

Write a unit test that invokes this method.

Testing the listener CallMonitor

Question

Consider the following method.

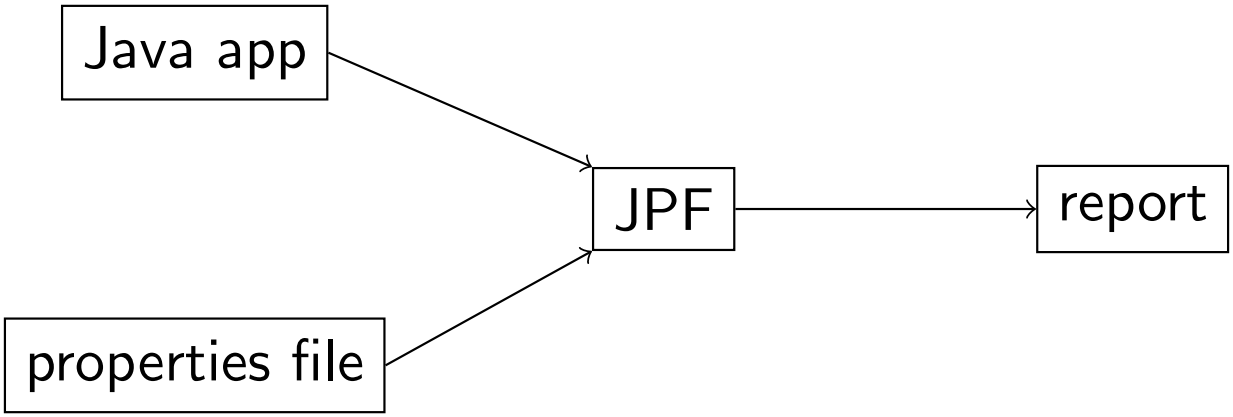
```
private static void staticMethod() {}
```

Write a unit test that invokes this method.

Answer

```
@Test
public void testStaticMethod() {
  if (this.verifyNoPropertyViolation(
        "+listener=gov.nasa.jpf.listener.CallMonitor",
        "+classpath=.")) {
    CallMonitorTest.staticMethod();
  }
}
```

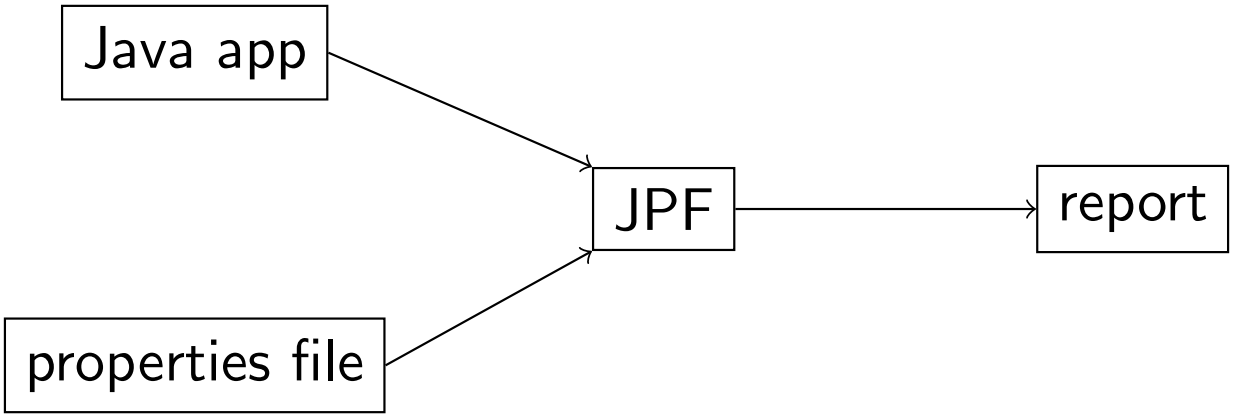
Testing the listener CallMonitor



Question

To test the CallMonitor listener in this way, for what in the report should we look?

Testing the listener CallMonitor



Question

To test the CallMonitor listener in this way, for what in the report should we look?

Answer

The report should contain the string `CallMonitorTest.staticMethod()`.

Testing the listener CallMonitor

Question

To where is the report written?

Testing the listener CallMonitor

Question

To where is the report written?

Answer

The console.

Testing the listener CallMonitor

Question

How can we write to a “string” instead of the console?

Testing the listener CallMonitor

Question

How can we write to a "string" instead of the console?

Answer

```
ByteArrayOutputStream stream = new ByteArrayOutputStream();
System.setOut(new PrintStream(stream));
```

Testing the listener CallMonitor

The class `TestJPF` contains static methods `assert...` similar to those found in JUnit.

```
TestJPF.assertTrue("Invocation of staticMethod incorrect",
  stream.toString().contains("CallMonitorTest.staticMethod(
```

Testing the listener CallMonitor

Question

How do we ensure that `TestJPF.assertTrue` writes to the console (and not to the `stream`)?

Testing the listener CallMonitor

Question

How do we ensure that `TestJPF.assertTrue` writes to the console (and not to the `stream`)?

Answer

```
PrintStream out = System.out; // remember for later
ByteArrayOutputStream stream = new ByteArrayOutputStream();
System.setOut(new PrintStream(stream));
...
System.setOut(out); // reset
TestJPF.assertTrue("Invocation of staticMethod incorrect",
  stream.toString().contains("CallMonitorTest.staticMethod(
```

Host JVM versus JPF's VM

- The test method is executed by the host JVM once. The JVM stops its execution if the test fails.
- The test method is model checked by JPF whenever the host JVM invokes a `verify...` method.

Host JVM versus JPF's VM

- The test method is executed by the host JVM once. The JVM stops its execution if the test fails.
- The test method is model checked by JPF whenever the host JVM invokes a `verify...` method.
- When a method `verify...` is executed by the host JVM, it invokes JPF to model check the enclosing test method, after which it returns false.
- When a method `verify...` is model checked by JPF it returns true.

Order

```
@Test
public void order() {
  System.out.println("1");
  if (this.verifyNoPropertyViolation(...)) {
    System.out.println("2");
  } else {
    System.out.println("3");
  }
  System.out.println("4");
}
```

Order

```
1
running jpf with args: ...
JavaPathfinder core system v8.0 (rev ...) - (C) 2005-2014 U
================================== system under test
BasicTest.runTestMethod()
================================== search started: 03/03/19
1
2
4
================================== results
no errors detected
================================== search finished: 03/03/1
3
4
```

Order

Step 1: black part of executed by the host JVM.

```
@Test
public void order() {
  System.out.println("1");
  if (this.verifyNoPropertyViolation(...))  {
    System.out.println("2");
  } else {
    System.out.println("3");
  }
  System.out.println("4");
}
```

Order

Step 2: black part is model checked by JPF.

```
@Test
public void order() {
  System.out.println("1");
  if (this.verifyNoPropertyViolation(...))  {
    System.out.println("2");
  } else {
    System.out.println("3");
  }
  System.out.println("4");
}
```

Order

Step 3: black part is executed by host JVM.

```
@Test
public void order() {
  System.out.println("1");
  if (this.verifyNoPropertyViolation(...)) {
    System.out.println("2");
  } else {
    System.out.println("3");
  }
  System.out.println("4");
}
```

Order

```
@Test
public void order() {
  System.out.println("1");
  if (this.verifyAssertionError(...)) {
    System.out.println("2");
  } else {
    System.out.println("3");
  }
  System.out.println("4");
}
```

Order

```
1
running jpf with args: ...
JavaPathfinder core system v8.0 (rev 32+) - (C) 2005-2014 U
================================================== syst
BasicTest.runTestMethod()
================================================== sear
1
2
4
================================================== resu
no errors detected
================================================== sear
```

Order

Step 1: black part of executed by the host JVM.

```
@Test
public void order() {
  System.out.println("1");
  if (this.verifyAssertionError(...))  {
    System.out.println("2");
  } else {
    System.out.println("3");
  }
  System.out.println("4");
}
```

Order

Step 2: black part is model checked by JPF.

```
@Test
public void order() {
  System.out.println("1");
  if (this.verifyAssertionError(...))  {
    System.out.println("2");
  } else {
    System.out.println("3");
  }
  System.out.println("4");
}
```

Order

Step 3: execution by the host JVM stop as the test failed.

```
@Test
public void order() {
  System.out.println("1");
  if (this.verifyAssertionError(...)) {
    System.out.println("2");
  } else {
    System.out.println("3");
  }
  System.out.println("4");
}
```

Testing the listener CallMonitor

```
@Test
public void staticMethodTest() {
  PrintStream out = System.out;
  ByteArrayOutputStream stream = new ByteArrayOutputStream(
  System.setOut(new PrintStream(stream));
  if (this.verifyNoPropertyViolation(
        "+listener=gov.nasa.jpf.listener.CallMonitor",
        "+classpath=.")) {
    CallMonitorTest.staticMethod();
  } else {
    System.setOut(out);
    TestJPF.assertTrue("Invocation of staticMethod incorrec
      stream.toString().contains("CallMonitorTest.staticMet
  }
}
```

Testing the listener CallMonitor

To avoid that

```
PrintStream out = System.out;
ByteArrayOutputStream stream = new ByteArrayOutputStream();
System.setOut(new PrintStream(stream));
```

is model checked, we can use

```
PrintStream out = null;
ByteArrayOutputStream stream = null;
if (!TestJPF.isJPFRun()) {
  out = System.out;
  stream = new ByteArrayOutputStream();
  System.setOut(new PrintStream(stream));
}
```

isJPFRun and isJUnitRun

The static methods `isJPFRun` and `isJUnitRun` of the class `TestJPF` check whether the code is model checked or tested, respectively.

isJPFRun and isJUnitRun

The static methods `isJPFRun` and `isJUnitRun` of the class `TestJPF` check whether the code is model checked or tested, respectively.

- When the method `isJPFRun` is executed by the host JVM, it returns false.
- When the method `isJPFRun` is model checked by JPF, it returns true.

isJPFRun and isJUnitRun

The static methods `isJPFRun` and `isJUnitRun` of the class `TestJPF` check whether the code is model checked or tested, respectively.

- When the method `isJPFRun` is executed by the host JVM, it returns false.
- When the method `isJPFRun` is model checked by JPF, it returns true.
- When the method `isJUnitRun` is executed by the host JVM within a test method, it returns true.
- When the method `isJUnitRun` is model checked by JPF, it returns false.

Testing the listener CallMonitor

Use techniques and tools, as discussed in the course EECS 4313 Software Engineering Testing, to develop the test case for the `CallMonitor` listener.

Note that tools such as EclEmma may not provide accurate feedback, as the code is executed in different modes.

```
65      public void methodTest() {
66          PrintStream out = null;
67          ByteArrayOutputStream stream = null;
68          if (!TestJPF.isJPFRun()) {
69              out = System.out;
70              stream = new ByteArrayOutputStream();
71              System.setOut(new PrintStream(stream));
72          }
73          if (this.verifyNoPropertyViolation(CallMonito
74              this.method();
75          } else {
76              System.setOut(out);
77              TestJPF.assertTrue("Invocation of method
78          }
79      }
```