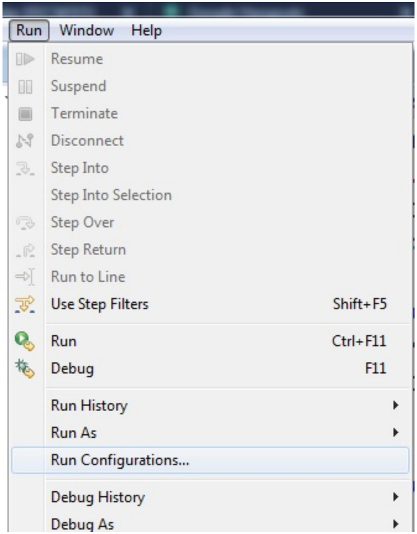
Running JPF in Eclipse

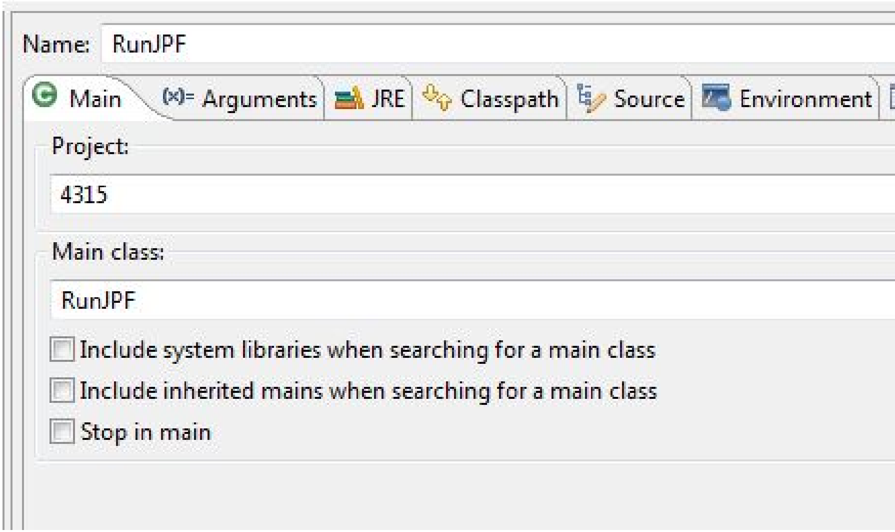
Create the following app.

```
public class RunJPF {
  public static void main(String[] args) {
    // location of .jpf directory
    final String HOME = "/courses/4315/";
    System.setProperty("user.home", HOME);
    gov.nasa.jpf.tool.RunJPF.main(args);
  }
}
```

Eclipse: Run → Run Configurations ...



Eclipse: Run → Run Configurations ... → Main



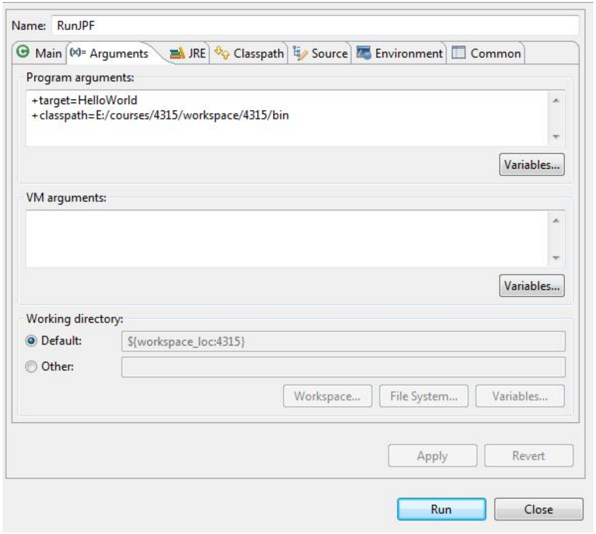
Name: RunJPF

Project:
4315

Main class:
RunJPF

☐ Include system libraries when searching for a main class
☐ Include inherited mains when searching for a main class
☐ Stop in main

Eclipse: Run → Run Configurations ... → Arguments



Name: RunJPF

Program arguments:

```
+target=HelloWorld
+classpath=E:/courses/4315/workspace/4315/bin
```

VM arguments:

Working directory:

[✓ Default:] ${workspace_loc:4315}

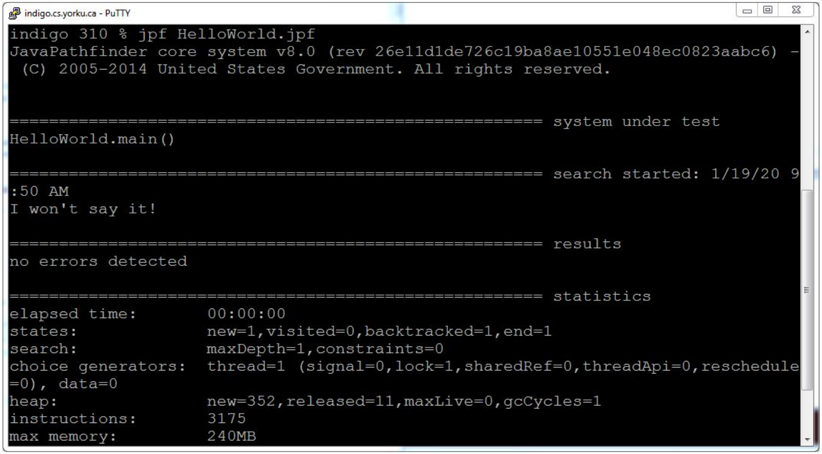
Other:

Eclipse plugin

Section 1.5 of the notes describes how to install the JPF plugin and Section 2.3 describes how to run JPF with the plugin.

Linux command line

Go to the directory that contains the application properties file (for example, HelloWord.jpf) and type
`jpf <name of the application properties file>`.



indigo.cs.yorku.ca - PuTTY

```
indigo 310 % jpf HelloWorld.jpf
JavaPathfinder core system v8.0 (rev 26e11d1de726c19ba8ae10551e048ec0823aabc6) -
 (C) 2005-2014 United States Government. All rights reserved.


====================================================== system under test
HelloWorld.main()

====================================================== search started: 1/19/20 9
:50 AM
I won't say it!

====================================================== results
no errors detected

====================================================== statistics
elapsed time:        00:00:00
states:              new=1,visited=0,backtracked=1,end=1
search:              maxDepth=1,constraints=0
choice generators:   thread=1 (signal=0,lock=1,sharedRef=0,threadApi=0,reschedule
=0), data=0
heap:                new=352,released=11,maxLive=0,gcCycles=1
instructions:        3175
max memory:          240MB
```

Viewing .dot files

`https://www.graphviz.org/`

The ColorTest revisited

In a previous lecture, we wrote a JUnit test case to test the `Color` class.

- JPF can only be run on apps, that is, classes that contain a main method.
- By default JPF checks for uncaught exceptions.

The ColorTest revisited

In a previous lecture, we wrote a JUnit test case to test the `Color` class.

- JPF can only be run on apps, that is, classes that contain a main method.
- By default JPF checks for uncaught exceptions.

Question

Write an app that invokes the JUnit test case methods.

The ColorTest revisited

```
package lab;

public class RunTest {
  public static void main(String[] args) {
    ColorTest tester = new ColorTest();
    tester.testConstructorAndAccessors();
    tester.testBLACK();
    tester.testEquals();
    tester.testEqualsRandomly();
  }
}
```

The ColorTest revisited

```
target=lab.RunTest
classpath=<path to directory that contains lab>; \
<paths to several jar files>
cg.enumerate_random=true
```

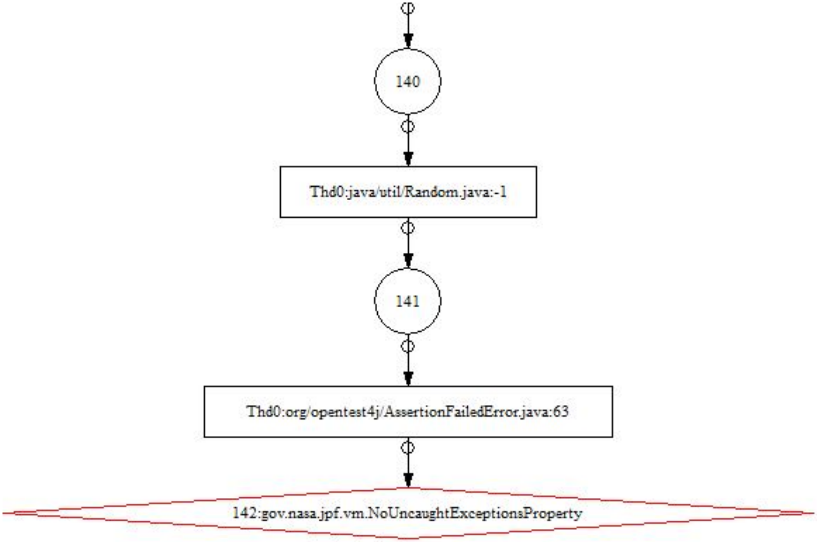
- `target` contains both the class name and the package name.
- Some of the JUnit jars needs to be added to the `classpath`.

The ColorTest revisited

```
=================================================== erro
gov.nasa.jpf.vm.NoUncaughtExceptionsProperty
org.opentest4j.AssertionFailedError: expected: <13> but was
at org.junit.jupiter.api.AssertionUtils.fail(org/junit/jup
at org.junit.jupiter.api.AssertEquals.failNotEqual(org/jun
at org.junit.jupiter.api.AssertEquals.assertEquals(org/jun
at org.junit.jupiter.api.AssertEquals.assertEquals(org/jun
at org.junit.jupiter.api.Assertions.assertEquals(org/junit/
at lab.ColorTest.testConstructorAndAccessors(lab/ColorTest.
at lab.RunTest.main(lab/RunTest.java:11)


=================================================== snap
thread java.lang.Thread:{id:0,name:main,status:RUNNING,prio
call stack:
at org.junit.jupiter.api.AssertionUtils.fail(AssertionUtils
at org.junit.jupiter.api.AssertEquals.failNotEqual(Assert
```

The ColorTest revisited


140
Thd0:java/util/Random.java:-1
141
Thd0:org/opentest4j/AssertionFailedError.java:63
142:gov.nasa.jpf.vm.NoUncaughtExceptionsProperty

The ColorTest revisited

By default, JPF stops after detecting a bug.

The ColorTest revisited

By default, JPF stops after detecting a bug.

To find multiple bugs . . .

```
target=lab.RunTest
classpath=<path to directory that contains lab>; \
<paths to several jar files>
cg.enumerate_random=true
search.multiple_errors=true
```

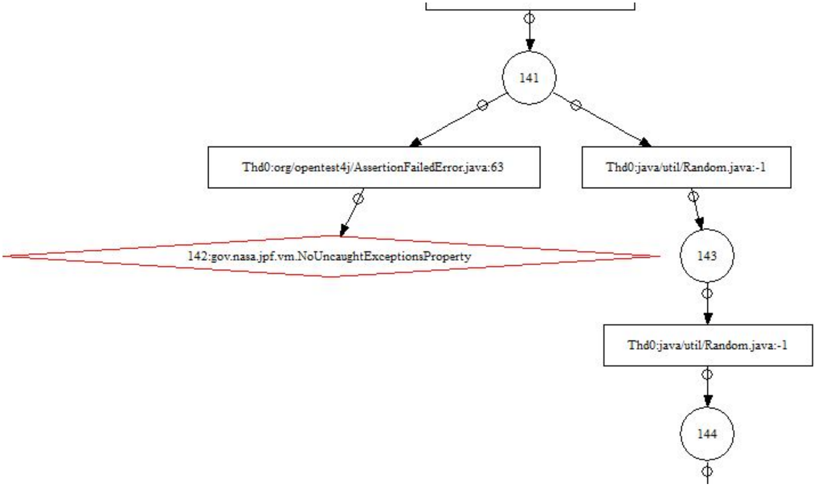
The ColorTest revisited

```
=========================================== erro
gov.nasa.jpf.vm.NoUncaughtExceptionsProperty
org.opentest4j.AssertionFailedError: expected: <13> but was
at org.junit.jupiter.api.AssertionUtils.fail(org/junit/jupi
...
at lab.RunTest.main(lab/RunTest.java:11)

=========================================== snap
thread java.lang.Thread:{id:0,name:main,status:RUNNING,prio
call stack:
...
at lab.RunTest.main(RunTest.java:11)

=========================================== erro
gov.nasa.jpf.vm.NoUncaughtExceptionsProperty
org.opentest4j.AssertionFailedError: expected: <13> but was
...
```

The ColorTest revisited

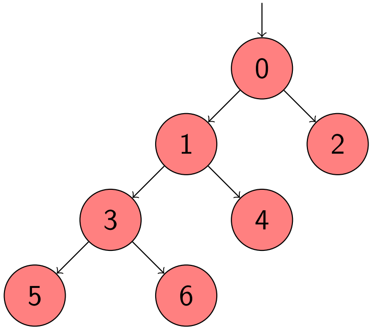

141
Thd0:org/opentest4j/AssertionFailedError.java:63
Thd0:java/util/Random.java:-1
142:gov.nasa.jpf.vm.NoUncaughtExceptionsProperty
143
Thd0:java/util/Random.java:-1
144

State Space Traversals

EECS 4315

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

Search



Question

Develop a Java app using the `nextBoolean` method of the `Random` class that gives rise to the above state space and that prints the number of a state when it is first visited.

Search

```
Random random = new Random();
System.out.println("0");
if (random.nextBoolean()) {
  System.out.println("2");
} else {
  System.out.println("1");
  if (random.nextBoolean()) {
    System.out.println("4");
  } else {
    System.out.println("3");
    if (random.nextBoolean()) {
      System.out.println("6");
    } else {
      System.out.println("5");
    }
  }
}
```

Search

Question

What is the output when the state space is traversed using depth-first search?

Search

Question

What is the output when the state space is traversed using depth-first search?

Answer

0 1 3 5 6 4 2

Search

Question

What is the output when the state space is traversed using depth-first search?

Answer

0 1 3 5 6 4 2

Question

What is the output when the state space is traversed using breadth-first search?

Search

Question

What is the output when the state space is traversed using depth-first search?

Answer

0 1 3 5 6 4 2

Question

What is the output when the state space is traversed using breadth-first search?

Answer

0 1 2 3 4 5 6

Search

Task

Verify your program using JPF with BFS and DFS. To do that, you need to create an application properties file (.jpf file) for your Java app. Configure the search property to be `gov.nasa.jpf.search.heuristic.BFSHeuristic` or `gov.nasa.jpf.search.heuristic.DFSHeuristic`.

Search

```
target=Traversal
classpath=.
cg.enumerate_random=true
search=gov.nasa.jpf.search.heuristic.BFSHeuristic
```

Search

```
====================================== syst
Traversal.main()
====================================== sear
0
1
3
5
6
4
2
====================================
```

Search

That is not breadth first search!

Question

Have we set the search property correctly? How can we check that?

Search

That is not breadth first search!

Question

Have we set the search property correctly? How can we check that?

Answer

Use the following command line arguments

- `-log`: lists the order in which properties files got loaded
- `-show`: prints all configuration entries after the initialization is complete

Log option

```
loading property file: /eecs/home/franck/.jpf/site.properties
loading property file: /eecs/fac/pkg/jpf/jpf-core/jpf.properties
loading property file: Traversal.jpf
collected native_classpath=/eecs/fac/pkg/jpf/jpf-core/build/jpf.
collected native_libraries=null
```

Show option

```
...
search = gov.nasa.jpf.search.heuristic.BFSHeuristic
search.class = gov.nasa.jpf.search.DFSearch
...
```

Search

```
target=Traversal
classpath=.
cg.enumerate_random=true
search.class=gov.nasa.jpf.search.heuristic.BFSHeuristic
```

Search

```
======================================== syst
Traversal.main()
======================================== sear
0
1
2
3
4
5
6
======================================
```

Search

```
target=Traversal
classpath=.
cg.enumerate_random=true
search.class=gov.nasa.jpf.search.heuristic.DFSHeuristic
```

Search

```
======================================== syst
Traversal.main()
======================================== sear
0
1
2
3
4
5
6
======================================
```

Search

That is not depth first search!

Search

That is not depth first search!

Let's try instead `gov.nasa.jpf.search.DFSearch`.

Search

```
target=Traversal
classpath=.
cg.enumerate_random=true
search.class=gov.nasa.jpf.search.DFSearch
```

Search

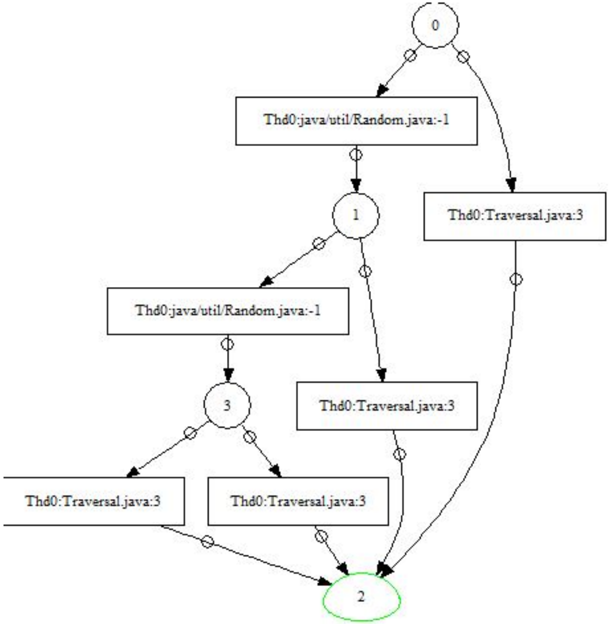
```
======================================================= syst
Traversal.main()
======================================================= sear
0
1
3
5
6
4
2
====================================================
```

Search

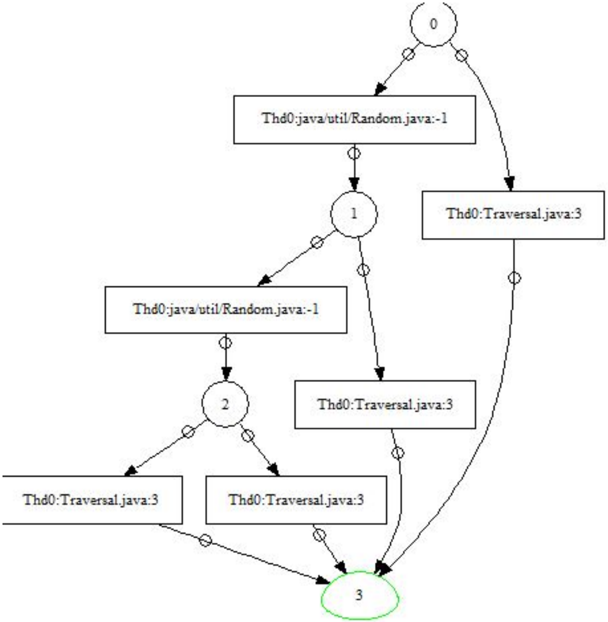
Task

Generate the state space diagram for BFS and DFS. To do this you need to set the listener to `StateSpaceDot`.

BFS


0
Thd0:java/util/Random.java:-1
1
Thd0:Traversal.java:3
Thd0:java/util/Random.java:-1
3
Thd0:Traversal.java:3
Thd0:Traversal.java:3
Thd0:Traversal.java:3
2

DFS


0
Thd0:java/util/Random.java:-1
1
Thd0:Traversal.java:3
Thd0:java/util/Random.java:-1
2
Thd0:Traversal.java:3
Thd0:Traversal.java:3
Thd0:Traversal.java:3
3

Search

Task

Verify your program using RS. RS can explore several random executions and in JPF you have the freedom to set the maximum number of executions you would like RS to explore. Firstly, set your search strategy to `gov.nasa.jpf.search.RandomSearch`. Secondly, set the `search.RandomSearch.path_limit` property to be any integer larger than 0. Compare the resulting state space diagrams.

Search

```
JavaPathfinder core system v8.0 (rev 2+) - (C) 2005-2014 Un


====================================== syst
Traversal.main()

====================================== sear
0
1
3
5
```

Search

No space diagram has been produced.

Search

No space diagram has been produced.

Bugs are everywhere, even in JPF!

Project

- January 24: install JPF (5%)
- February 14: draft proposal (2%)
- February 24: proposal (3%)
- March 9: first progress report (5%)
- March 23: second progress report (5%)
- March 30 and April 1: presentation (10%)
- Exam period: report (10%)
- Exam period: code (10%)

Project

Very brief descriptions of the last four years' projects can be found here.

Students can work alone or in groups of two on their project.

Students are expected to work on average two hours per week on their projects from now on.

Last year's project

Several group have made contributions to JPF:

- document some of JPF's search listeners (submitted to JPF)
- update the JPF plugin for Eclipse so that it works with version 4.10 (to be submitted soon)
- improve and test the listener gov.nasa.jpf.listener.MethodTracker (to be submitted)
- improve and test the listener gov.nasa.jpf.listener.NumericValueChecker (to be submitted)

Suggestions for this year's project

For a component of JPF (for example, a listener)

- improve the code,
- document the code,
- develop tests (JPF has its own testing framework that extends JUnit 4), and
- write a user manual.

Of course, students can propose other projects as well.