Concurrency

EECS 4315

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

Books

- Brian Goetz, Tim Peierls, Joshua Bloch, Joseph Bowbeer, David Holmes and Doug Lea. *Java Concurrency in Practice*. Addison-Wesley, 2006.
- Mary Campione, Kathy Walrath and Alison Huml. *The Java Tutorial. Lesson: Threads: Doing Two or More Tasks At Once*.
- James Gosling, Bill Joy, Guy L. Steele Jr., Gilad Bracha and Alex Buckley. *The Java Language Specification*. 2015.

Concurrency

Threads can exchange information by accessing and updating shared attributes.

Question

One thread executes

```
v = 1;
v = v + 1;
```

and another thread executes

```
v = 0;
```

What is the final value of v?

Concurrency

Threads can exchange information by accessing and updating shared attributes.

Question

One thread executes

```
v = 1;
v = v + 1;
```

and another thread executes

```
v = 0;
```

What is the final value of v?

Answer

0, 1 or 2. This example shows that concurrency gives rise to nondeterminism.

Question

One thread executes

```
v = v + 1;
```

and another thread executes

```
v = v + 1;
```

If the initial value of v is 0, then what is the final value of v?

Question

One thread executes

```
v = v + 1;
```

and another thread executes

```
v = v + 1;
```

If the initial value of v is 0, then what is the final value of v?

Answer

1 or 2.

Question

How can the final value of v be 1?

Question

How can the final value of v be 1?

Answer

The assignment `v = v + 1` is not atomic.

Question

How can the final value of v be 1?

Answer

The assignment `v = v + 1` is not atomic.

```
0: getstatic
3: iconst_1
4: iadd
5: putstatic
```

Question

One thread executes

```
v = 0;
```

and another thread executes

```
v = Long.MAX_VALUE;
```

How many different final values can v have?

Question

One thread executes

```
v = 0;
```

and another thread executes

```
v = Long.MAX_VALUE;
```

How many different final values can v have?

Answer

4.

Question

How can v have 4 different final values?

Question

How can v have 4 different final values?

Answer

The assignments `v = 0` and `v = Long.MAX_VALUE` may not be atomic (on 32 bit machines).

Thread creation

In Java, threads are created dynamically:

```
// create and initialize Thread object
Thread thread = new Thread();
// execute run method of Thread object concurrently
thread.start();
```

The class `Thread` is part of package `java.lang` (and, hence, does not need to be imported). Its API can be found at the URL

`https://docs.oracle.com/javase/8/docs/api/java/lang/Thread.html` .

Thread API

- `public Thread(String name)`
  Initializes a new `Thread` object with the specified name as its name.
- `public void start()`
  Causes this thread to begin execution; the Java virtual machine calls the run method of this thread.
- `public void run()`
  This method does nothing and returns.

Printer

Question

Develop a Java class called `Printer` that is a `Thread` and prints its name 1000 times.

Printer

```
public class Printer extends Thread {
  public Printer(String name) {
    super(name);
  }

  public void run() {
    final int NUMBER = 1000;
    for (int i = 0; i < NUMBER; i++) {
      System.out.print(this.getName());
    }
  }
}
```

Two concurrent printers

Question

Develop an app that creates two `Printer`s with names 1 and 2 and run them concurrently.

Two concurrent printers

```
public class TwoPrinters {
  public static void main(String[] args) {
    Printer one = new Printer("1");
    Printer two = new Printer("2");
    one.start();
    two.start();
  }
}
```

Two concurrent printers

Question

What is the output of the app?

Two concurrent printers

Question

What is the output of the app?

Answer

A sequence of 1000 1's and 2's (arbitrarily interleaved). This example shows that concurrency gives rise to nondeterminism.

Two concurrent printers

Question

What happens if we replace `start` with `run` in the app?

Two concurrent printers

Question

What happens if we replace `start` with `run` in the app?

Answer

Let's try it.

Two concurrent printers

Question

What happens if we replace `start` with `run` in the app?

Answer

Let's try it.

Answer

The output is a sequence of 1000 1's followed by 1000 2's

Java only supports single inheritance

The following is not allowed in Java.

```
public class Printer extends Applet, Thread
```

Thread creation

```
// create and initialize Runnable object
Runnable runnable = new ...();
// create and initialize Thread object
Thread thread = new Thread(runnable);
// execute run method of Runnable object concurrently
thread.start();
```

The interface `Runnable` is part of package `java.lang` (and, hence, does not need to be imported). Its API can be found at the URL

`https://docs.oracle.com/javase/8/docs/api/java/lang/Runnable.html`

Runnable is an interface

In Java, you cannot create instances of an interface.

```
public class Printer implements Runnable {
  ...
}
```

The assignment

```
Runnable printer = new Printer();
```

is valid since the class `Printer` implements the interface `Runnable`.

Printer

Question

Develop a Java class called `Printer` that implements `Runnable` and prints the thread's name 1000 times.

Printer

```
public class Printer implements Runnable {
  public void run() {
    final int NUMBER = 1000;
    for (int i = 0; i < NUMBER; i++) {
      System.out.print(Thread.currentThread().getName());
    }
  }
}
```

Printer

```
public class TwoPrinters {
  public static void main(String[] args)    {
    Printer printer = new Printer();
    Thread one = new Thread(printer, "1");
    Thread two = new Thread(printer, "2");
    one.start();
    two.start();
  }
}
```

Incrementer

Question

Develop a Java class called `Incrementer` that is a `Thread` and increments a shared static attribute named `value`.

Incrementer

```
public class Incrementer extends Thread {
  public static int value = 0;

  public void run () {
    Incrementer.value++;
  }
}
```

Incrementer

Question

Develop an app that creates two `Incrementers` and run them concurrently. Assert that the final value of `value` is two.

Two incrementers

```
public class TwoIncrementers {
  public static void main(String[] args) {
    try {
      Incrementer one = new Incrementer();
      Incrementer two = new Incrementer();
      one.start();
      two.start();
      one.join();
      two.join();
      assert Incrementer.value == 2;
    } catch (InterruptedException e) {}
  }
}
```

Using JPF

We can use JPF to check whether the assertion hold for each execution.

```
target=TwoIncrementers
classpath=.
```

Using JPF

```
JavaPathfinder core system v8.0 (rev d772dfa80ea692f916aa67

====================================================== syst
TwoIncrementers.main()

====================================================== sear

====================================================== erro
gov.nasa.jpf.vm.NoUncaughtExceptionsProperty
java.lang.AssertionError
        at TwoIncrementers.main(TwoIncrementers.java:7)
...
```

Using jpf-visual

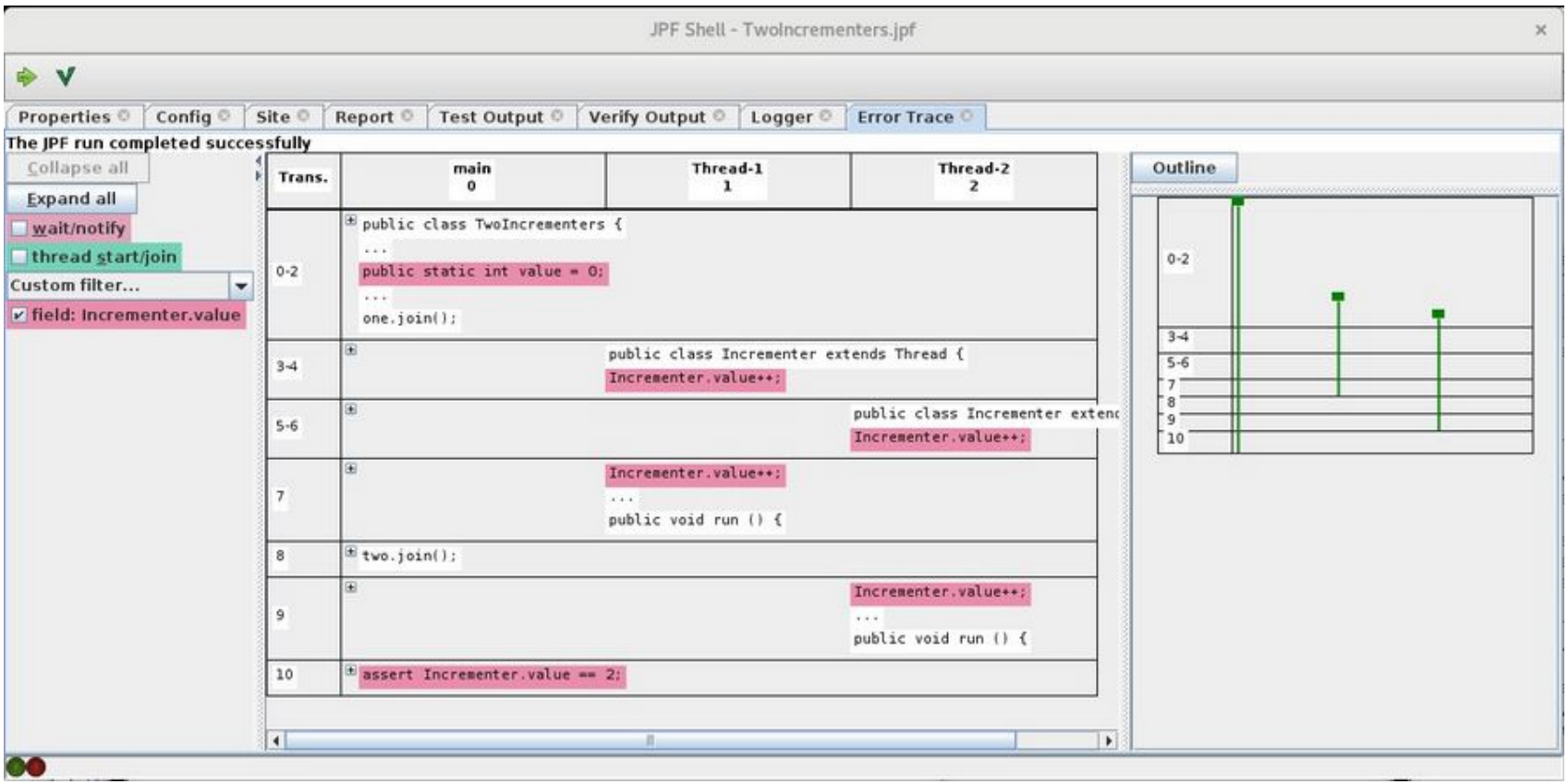
Install jpf-shell and jpf-visual.

```
target=TwoIncrementers
classpath=.
sourcepath=.

@using=jpf-visual

report.errorTracePrinter.property_violation=trace
report.publisher+=,errorTracePrinter
report.errorTracePrinter.class=ErrorTracePrinter
shell=gov.nasa.jpf.shell.basicshell.BasicShell
shell.panels+=,errorTrace
shell.panels.errorTrace=ErrorTracePanel
```

jpf-visual


JPF Shell - TwoIncrementers.jpf
Properties
Config
Site
Report
Test Output
Verify Output
Logger
Error Trace
The JPF run completed successfully
Collapse all
Expand all
wait/notify
thread start/join
Custom filter...
field: Incrementer.value
Trans.
main
0
Thread-1
1
Thread-2
2
0-2
public class TwoIncrementers {
...
public static int value = 0;
...
one.join();
3-4
public class Incrementer extends Thread {
Incrementer.value++;
5-6
public class Incrementer extenc
Incrementer.value++;
7
Incrementer.value++;
...
public void run () {
8
two.join();
9
Incrementer.value++;
...
public void run () {
10
assert Incrementer.value == 2;
Outline
0-2
3-4
5-6
7
8
9
10