The State Space in XML Format

EECS 4315

https://wiki.eecs.yorku.ca/course_archive/2016-17/W/4315/

Today's Plan

- XML
- JPF Report System
- Implementing a ***PublisherExtension***
- Parameterizing a Listener
- Model Classes

Today's Plan

- XML
- JPF Report System
- Implementing a *PublisherExtension*
- Parameterizing a Listener
- Model Classes

XML

- XML stands for eXtensible Markup Language
- XML was designed to store and transport data
- XML was designed to be both human- and machine-readable

XML vs HTML

XML and HTML were designed with different goals:

	XML	HTML
Goal	Carry data	Display data
Focus	what data	How data looks

- XML tags are not predefined like HTML tags are

Our XML report file

```
<state_space>
  <state id = 1>
        <transition target = 2> </transition>
        <transition target = 3> </transition>
  </state>
  <state id = 2 >
        <transition target = 3> </transition>
  </state>
</state_space >
```

Today's Plan

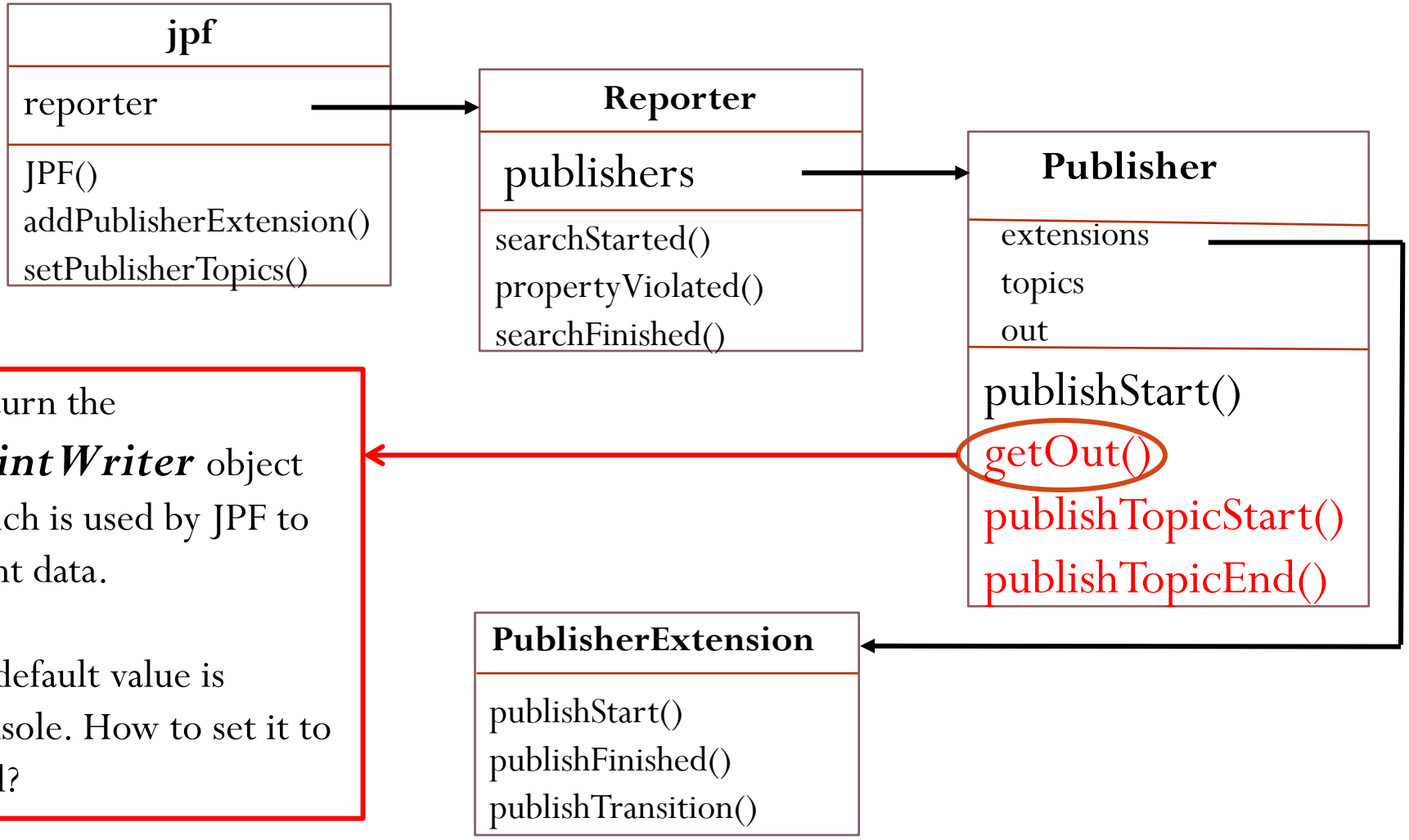
- XML
- JPF Report System
- Implementing a *PublisherExtension*
- Parameterizing a Listener
- Model Classes

JPF Report System

Three major components:

- the ***Reporter***
- any number of format specific ***Publisher*** objects
- any number of tool-, property- and Publisher-specific ***PublisherExtension*** objects

JPF Report System



Configure the Properties

- Set the publisher to be console or xml where console is the default value

report.publisher = xml

- Set the output file name

report.xml.file = HelloWorld

Today's Plan

- XML
- JPF Report System
- Implementing a ***PublisherExtension***
- Parameterizing a Listener
- Model Classes

The *PublisherExtension* Interface

```
public interface PublisherExtension {
        void publishStart(Publisher publisher);
        void publishTransition(Publisher publisher);
        void publishPropertyViolation(Publisher publisher);
        void publishConstraintHit(Publisher publisher);
        void publishFinished(Publisher publisher);
        void publishProbe(Publisher publisher);
}
```

Question:

How the Publisher object is used?

Answer:

PrintWriter out = publisher.getOut();

or

publisher.publishTopicStart("…");

publisher.publishTopicEnd("…");

Question:

How to print the first tag? Which method should we implement?

Answer: publishStart(Publisher){ }

Question:

How to print the last tag? Which method should we implement?

Answer: publishFinished(Publisher){ }

Question:

How to print the transitios? Which method should we implement?

Answer: publishTransition(Publisher){ }

```
<state_space>
  <state id = 1>
        <transition target = 2> </transition>
        <transition target = 3> </transition>
  </state>
</state_space >
```

StateSpaceXML

```
public class StateSpaceXML extends ListenerAdapter implements SearchListener, PublisherExtension{
        private int source;
        private int target;

        public StateSpaceXML(Config config, JPF jpf) {
            source = -1;
            target = -1;
            jpf.addPublisherExtension(Publisher.class, this);
        }

        @Override
        public void publishTransition(Publisher publisher) {
            PrintWriter out = publisher.getOut();
            if (source != -1) {
                out.println("<state id = " + this.source +">");
                //publisher. publishTopicStart("state id =" + this.source);
            }
        }
}
```

// The StateSpaceXML is improved to print the exact XML format in this lecture

//Notice that the application only works with BFS search strategy

Today's Plan

- XML
- JPF Report System
- Implementing a *PublisherExtension*
- Parameterizing a Listener
- Model Classes

Parameterizing a Listener

```
private String separator;
public StateSpacePrinter(Config config) {
    source = -1;
    target = -1;
    separator = config.getString("stateSpacePrinter.separator", "->");
}
```

We can set the separator in the application properties file:

```
stateSpacePrinter.separator =  -->
```

The default value is given in the constructor.

Today's Plan

- XML
- JPF Report System
- Implementing a *PublisherExtension*
- Parameterizing a Listener
- Model Classes

```
public class Sine {
        public static void main(String[] args) {
                System.out.println(StrictMath.sin(0.3));
        }
}
```

Question:

error #1: gov.nasa.jpf.vm.NoUncaughtExceptionsProperty
"java.lang.UnsatisfiedLinkError: cannot find native…"

Answer:

Because the sin method is native.

public static native double sin(double a);

Question:

What is native method?

Answer:

A method that is implemented in a language other than Java but that is invoked from a Java app.

- Allows programmers to use code that has been already implemented in other languages.
- May increase the performance.
- May support certain platform-dependent features.

Java Native Interface (JNI)

JNI provides the infrastructure for Java code to use libraries written in other languages such as C, C++ and assembly.

How JPF handle native methods?

- Using **model classes**.
- Using native peers.
- Using a combination of model classes and native peers.
- Using the extension jpf-nhandler.

Model Class

A model class captures the behaviour of a native method in pure Java.

Question:

How can we capture the behaviour of the sin method?

Answer:

For example, we approximate the sine function with the Bhaskara I's sine approximation formula:

$$\sin(a) = \frac{16a(\pi - a)}{5\pi^2 - 4a(\pi - a)}$$

```
package java.lang;


public class StrictMath {
    public static double sin(double a) {
        return 16 * a * (Math.PI - a) / (5 *
Math.PI * Math.PI - 4 * a * (Math.PI - a));

        //this also works
        //return Math.sin(a);
    }
}
```

Model Class

- The model class StrictMath is part of the package java.lang.
- The model class only contains one method, whereas the original StrictMath class contains many more.
- Add the path of StrictMath.class to native_classpath

Thank you!