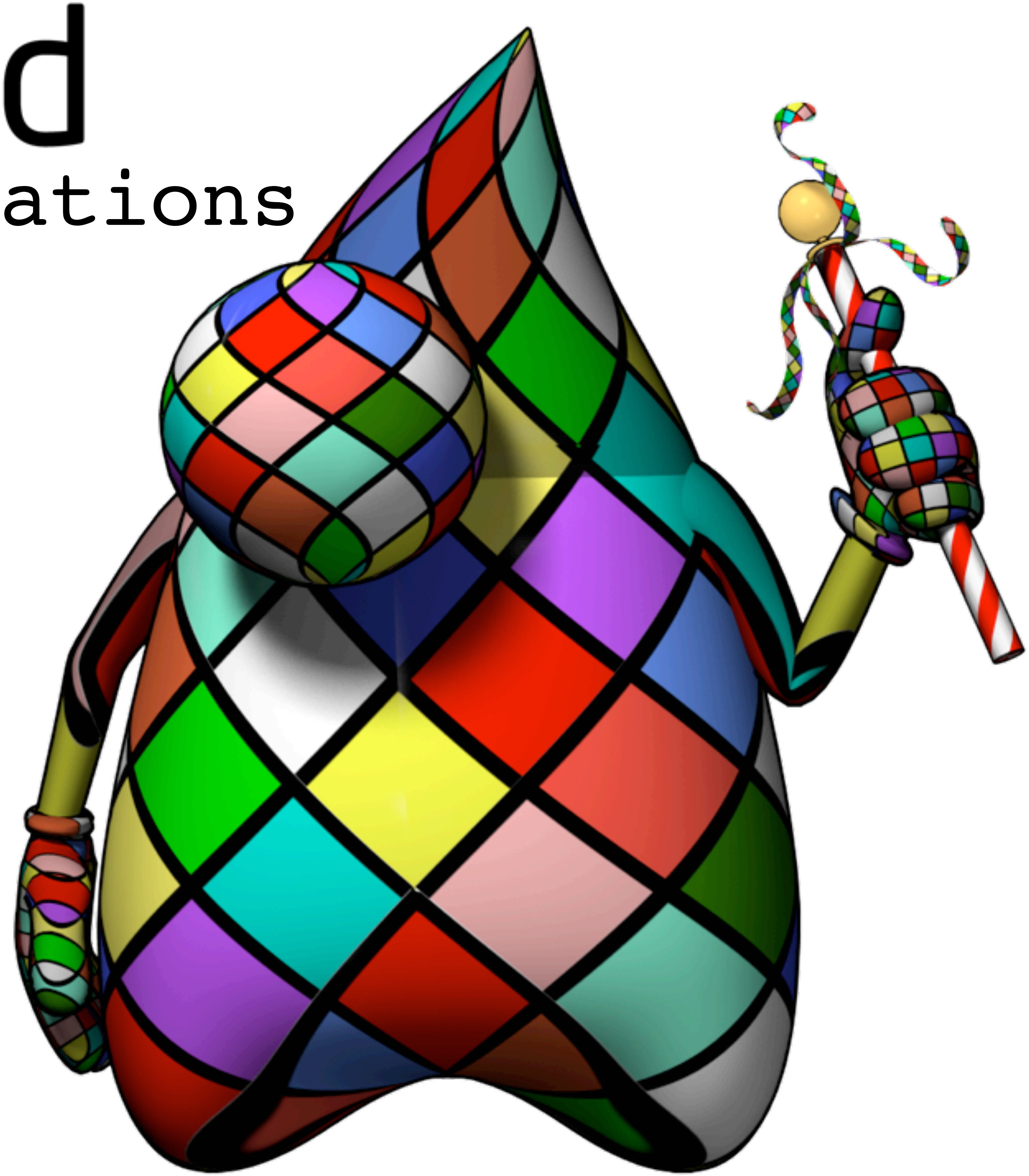
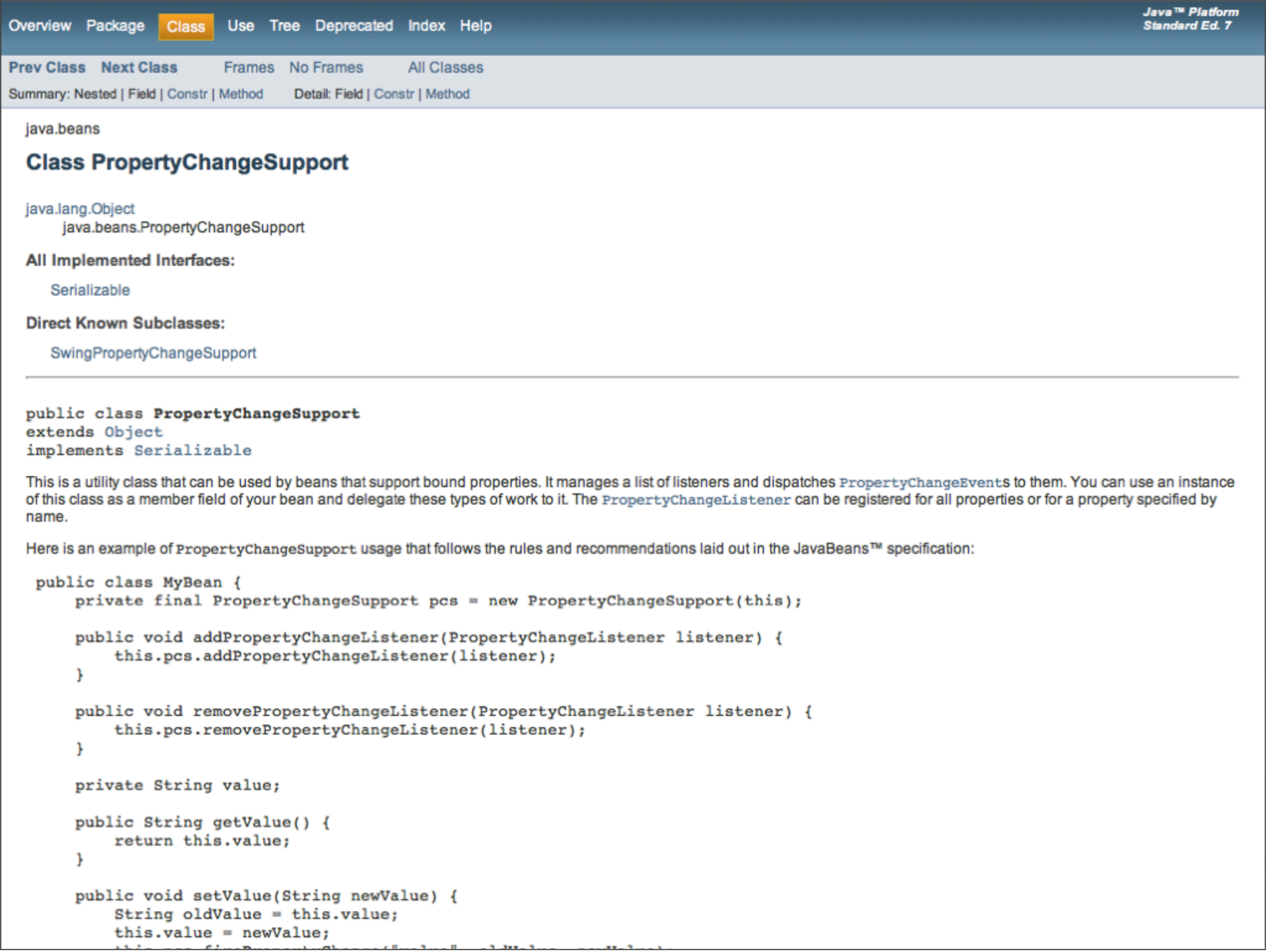
Xtend

Active Annotations




Overview
Package
Class
Use
Tree
Deprecated
Index
Help
Java™ Platform
Standard Ed. 7
Prev Class
Next Class
Frames
No Frames
All Classes
Summary: Nested | Field | Constr | Method
Detail: Field | Constr | Method

java.beans

Class PropertyChangeSupport

java.lang.Object
    java.beans.PropertyChangeSupport

All Implemented Interfaces:

Serializable

Direct Known Subclasses:

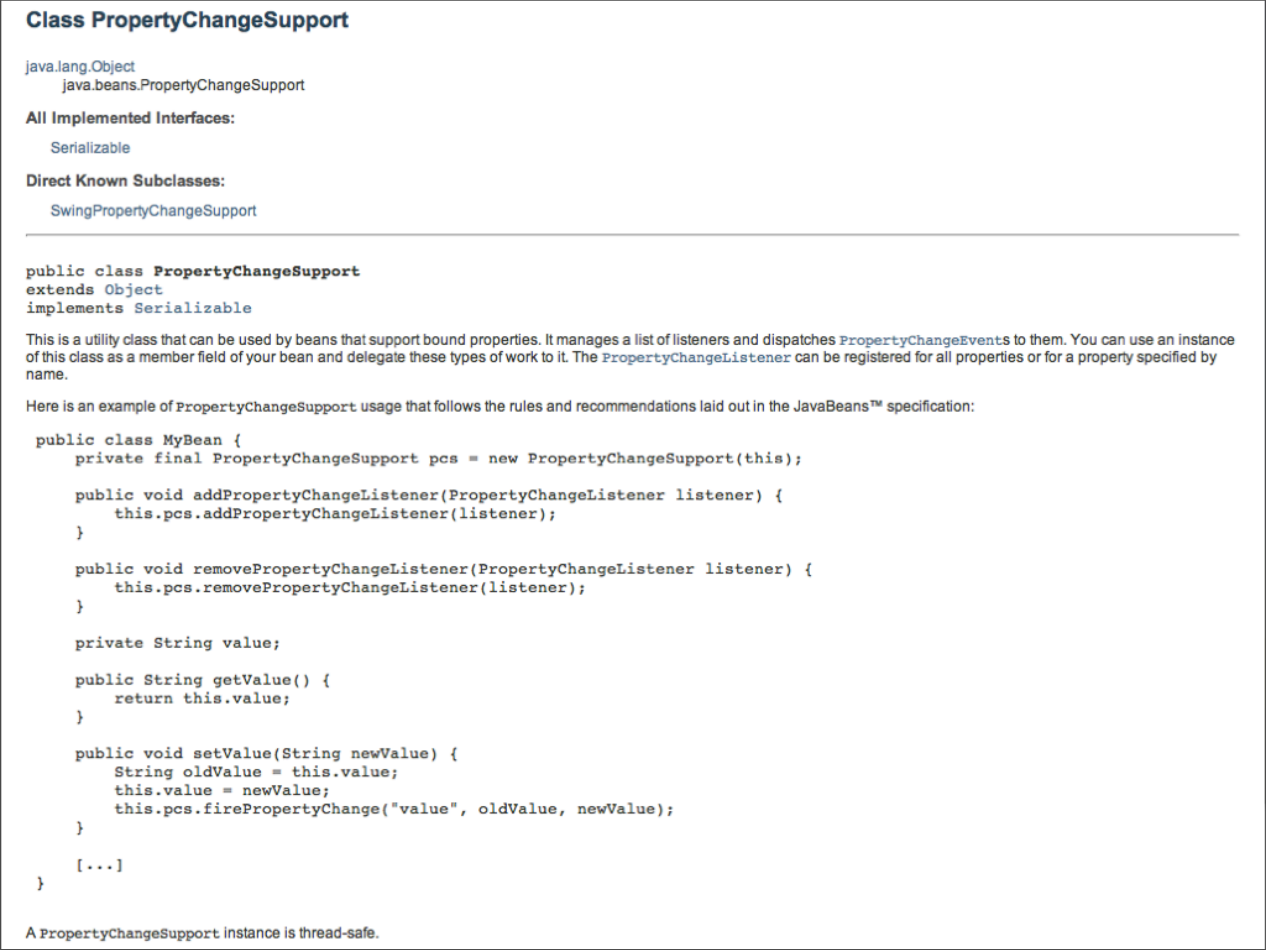
SwingPropertyChangeSupport

```
public class PropertyChangeSupport
extends Object
implements Serializable
```

This is a utility class that can be used by beans that support bound properties. It manages a list of listeners and dispatches `PropertyChangeEvent`s to them. You can use an instance of this class as a member field of your bean and delegate these types of work to it. The `PropertyChangeListener` can be registered for all properties or for a property specified by name.

Here is an example of `PropertyChangeSupport` usage that follows the rules and recommendations laid out in the JavaBeans™ specification:

```
 public class MyBean {
     private final PropertyChangeSupport pcs = new PropertyChangeSupport(this);

     public void addPropertyChangeListener(PropertyChangeListener listener) {
         this.pcs.addPropertyChangeListener(listener);
     }

     public void removePropertyChangeListener(PropertyChangeListener listener) {
         this.pcs.removePropertyChangeListener(listener);
     }

     private String value;

     public String getValue() {
         return this.value;
     }

     public void setValue(String newValue) {
         String oldValue = this.value;
         this.value = newValue;
```



Class PropertyChangeSupport

java.lang.Object
    java.beans.PropertyChangeSupport

All Implemented Interfaces:

Serializable

Direct Known Subclasses:

SwingPropertyChangeSupport

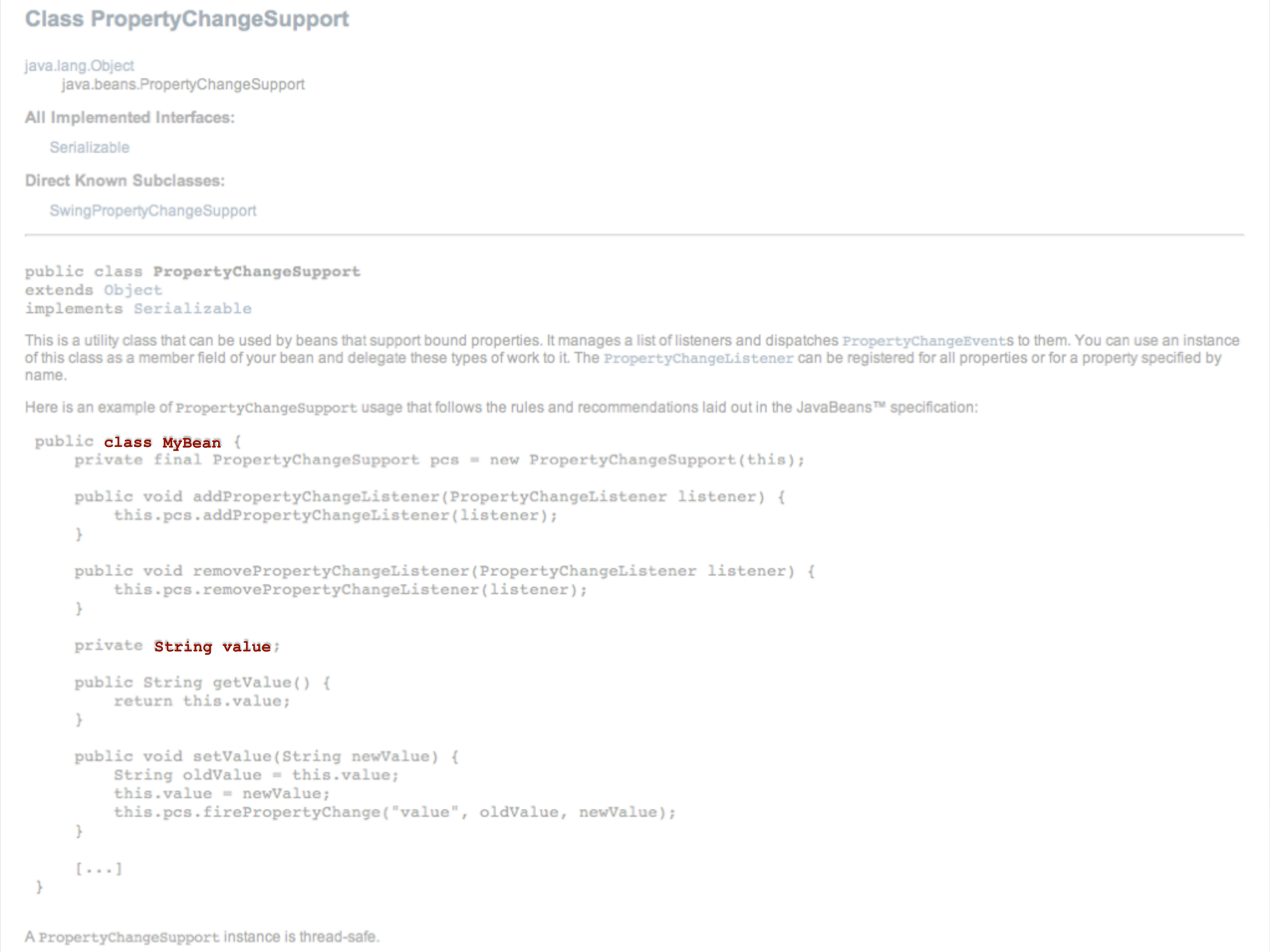
```
public class PropertyChangeSupport
extends Object
implements Serializable
```

This is a utility class that can be used by beans that support bound properties. It manages a list of listeners and dispatches `PropertyChangeEvent`s to them. You can use an instance of this class as a member field of your bean and delegate these types of work to it. The `PropertyChangeListener` can be registered for all properties or for a property specified by name.

Here is an example of `PropertyChangeSupport` usage that follows the rules and recommendations laid out in the JavaBeans™ specification:

```
 public class MyBean {
     private final PropertyChangeSupport pcs = new PropertyChangeSupport(this);

     public void addPropertyChangeListener(PropertyChangeListener listener) {
         this.pcs.addPropertyChangeListener(listener);
     }

     public void removePropertyChangeListener(PropertyChangeListener listener) {
         this.pcs.removePropertyChangeListener(listener);
     }

     private String value;

     public String getValue() {
         return this.value;
     }

     public void setValue(String newValue) {
         String oldValue = this.value;
         this.value = newValue;
         this.pcs.firePropertyChange("value", oldValue, newValue);
     }

     [...]
 }
```

A `PropertyChangeSupport` instance is thread-safe.



Class PropertyChangeSupport

java.lang.Object
    java.beans.PropertyChangeSupport

All Implemented Interfaces:

Serializable

Direct Known Subclasses:

SwingPropertyChangeSupport

```
public class PropertyChangeSupport
extends Object
implements Serializable
```

This is a utility class that can be used by beans that support bound properties. It manages a list of listeners and dispatches `PropertyChangeEvent`s to them. You can use an instance of this class as a member field of your bean and delegate these types of work to it. The `PropertyChangeListener` can be registered for all properties or for a property specified by name.

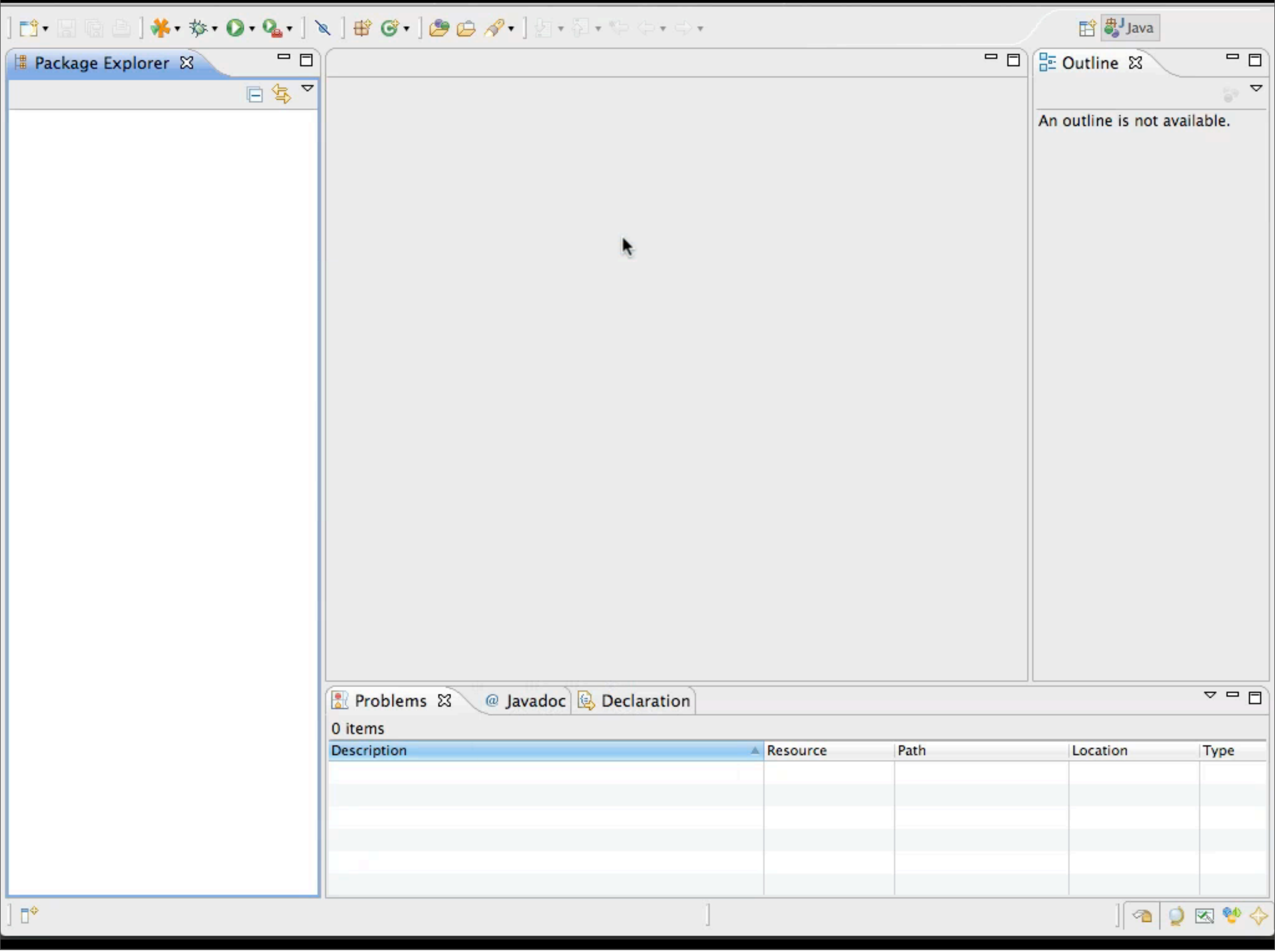
Here is an example of `PropertyChangeSupport` usage that follows the rules and recommendations laid out in the JavaBeans™ specification:

```
public class MyBean {
    private final PropertyChangeSupport pcs = new PropertyChangeSupport(this);

    public void addPropertyChangeListener(PropertyChangeListener listener) {
        this.pcs.addPropertyChangeListener(listener);
    }

    public void removePropertyChangeListener(PropertyChangeListener listener) {
        this.pcs.removePropertyChangeListener(listener);
    }

    private String value;

    public String getValue() {
        return this.value;
    }

    public void setValue(String newValue) {
        String oldValue = this.value;
        this.value = newValue;
        this.pcs.firePropertyChange("value", oldValue, newValue);
    }

    [...]
}
```

A `PropertyChangeSupport` instance is thread-safe.

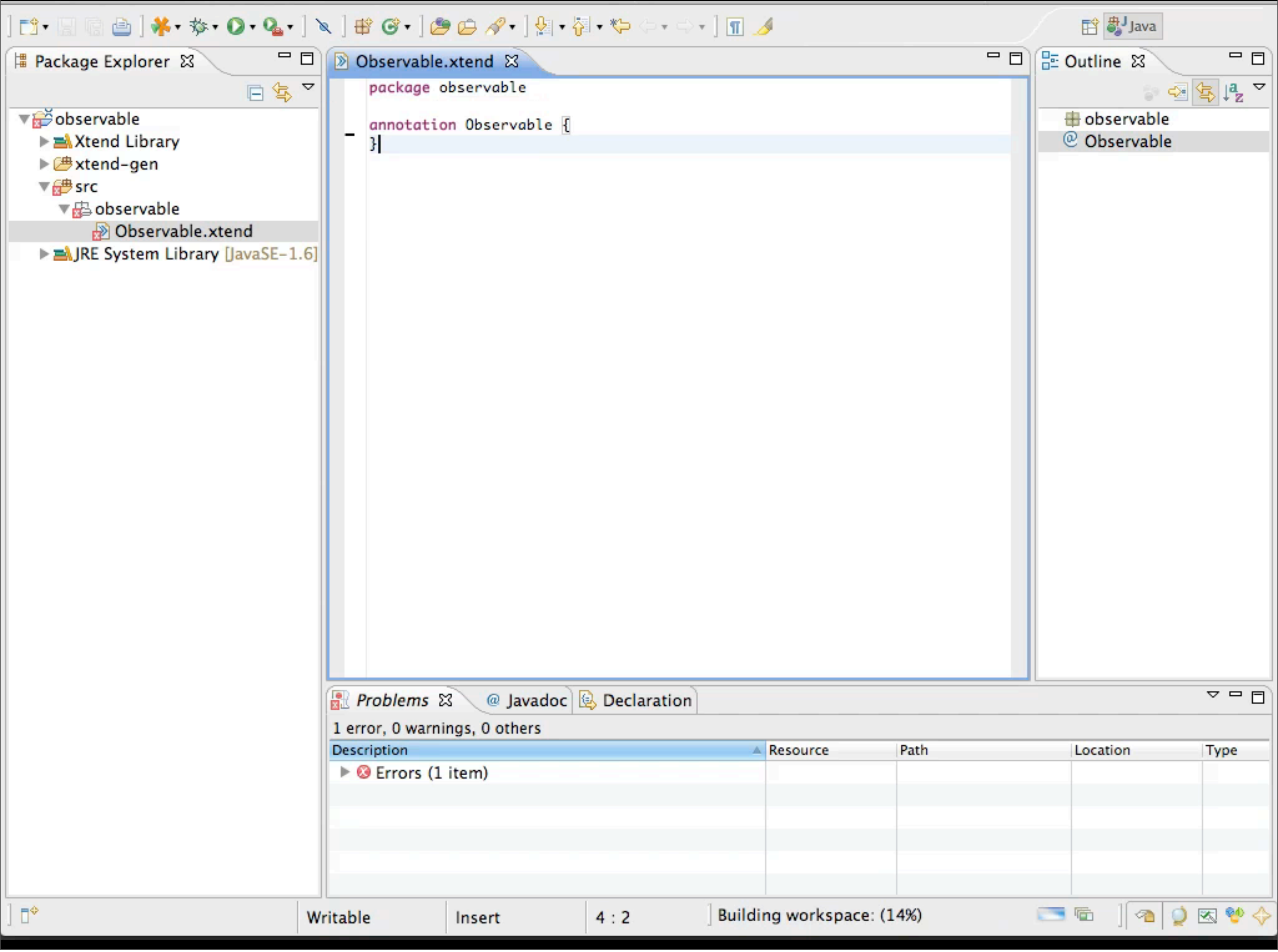
```
class MyBean {
  String value
}
```

```
@Observable
class MyBean {
  String value
}
```


Java
Package Explorer
Outline
Problems
Javadoc
Declaration
Description
Resource
Path
Location
Type

An outline is not available.

0 items

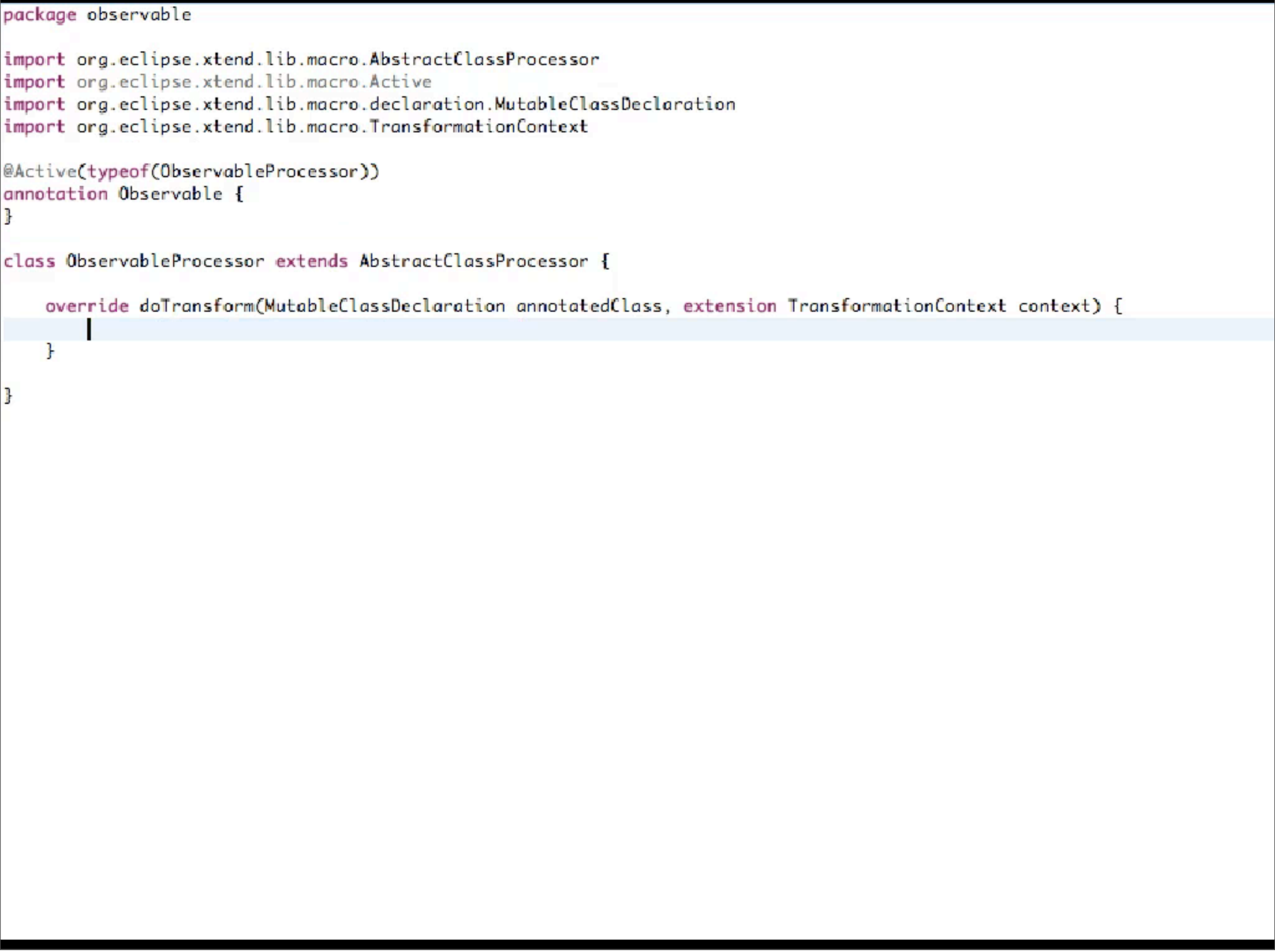

Java
Package Explorer
observable
Xtend Library
xtend-gen
src
observable
Observable.xtend
JRE System Library [JavaSE-1.6]
Observable.xtend
Outline
observable
Observable
Problems
Javadoc
Declaration

```
package observable

annotation Observable {
}
```

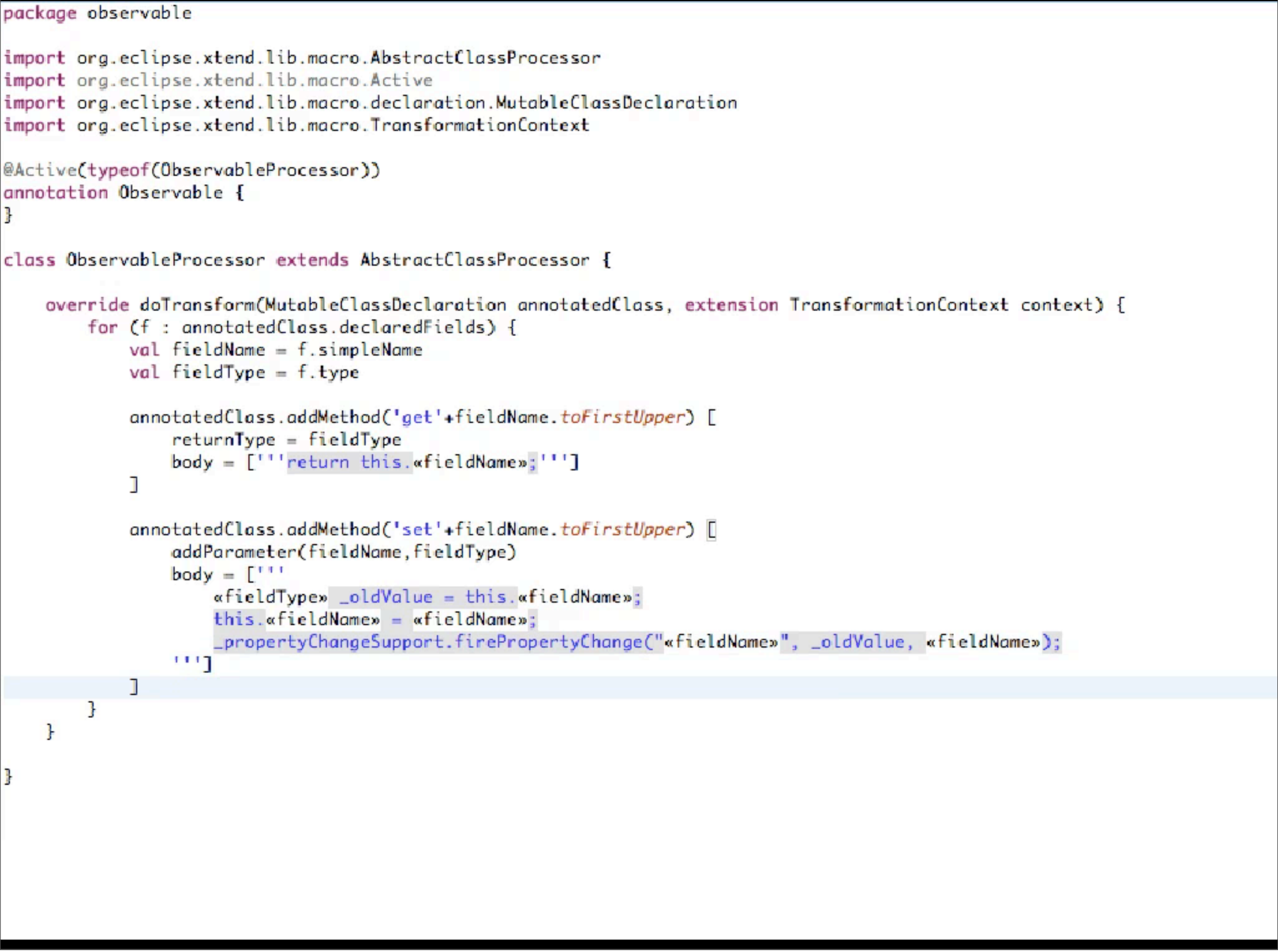
1 error, 0 warnings, 0 others

Description	Resource	Path	Location	Type
Errors (1 item)				

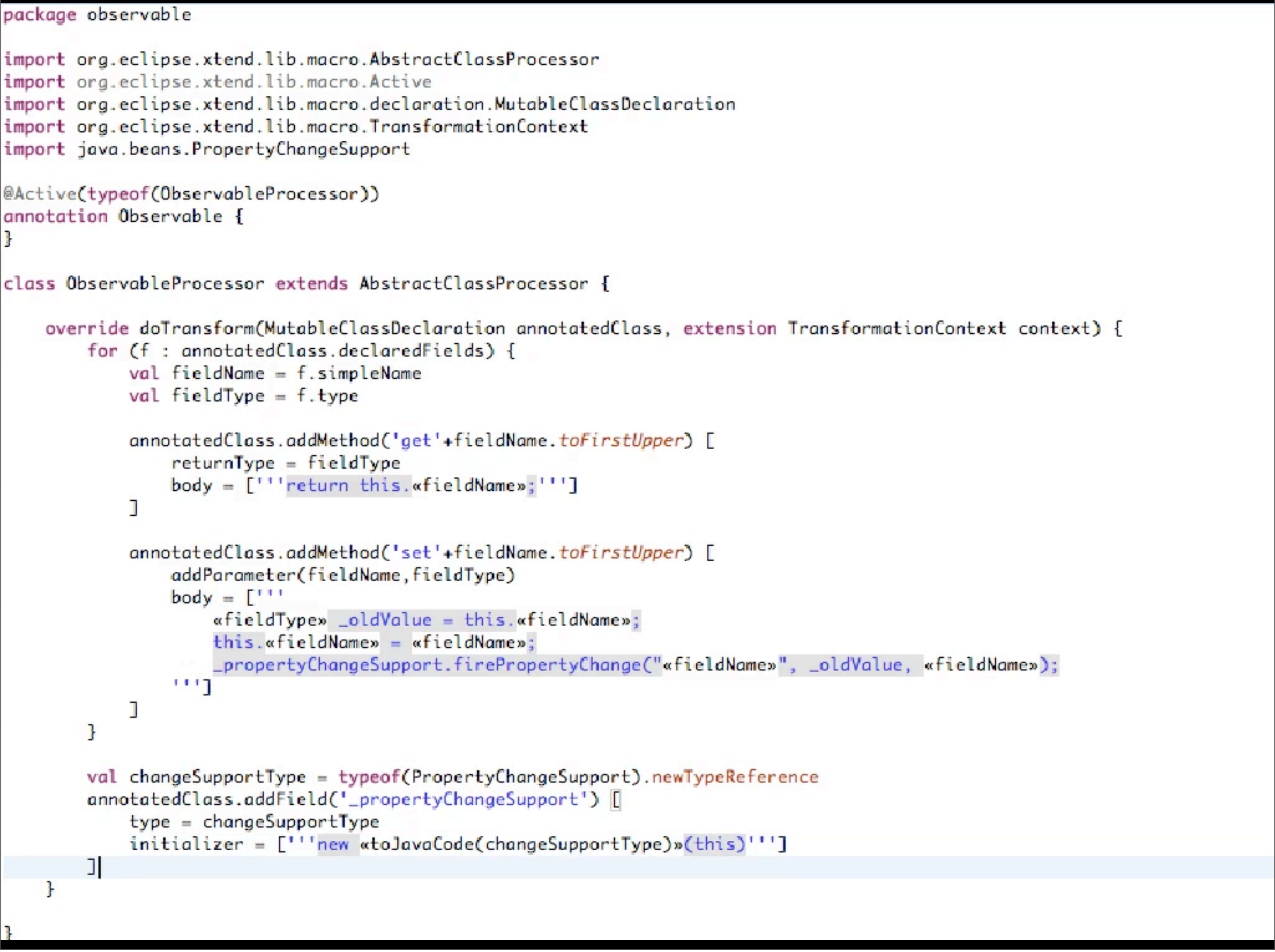
Writable
Insert
4 : 2
Building workspace: (14%)



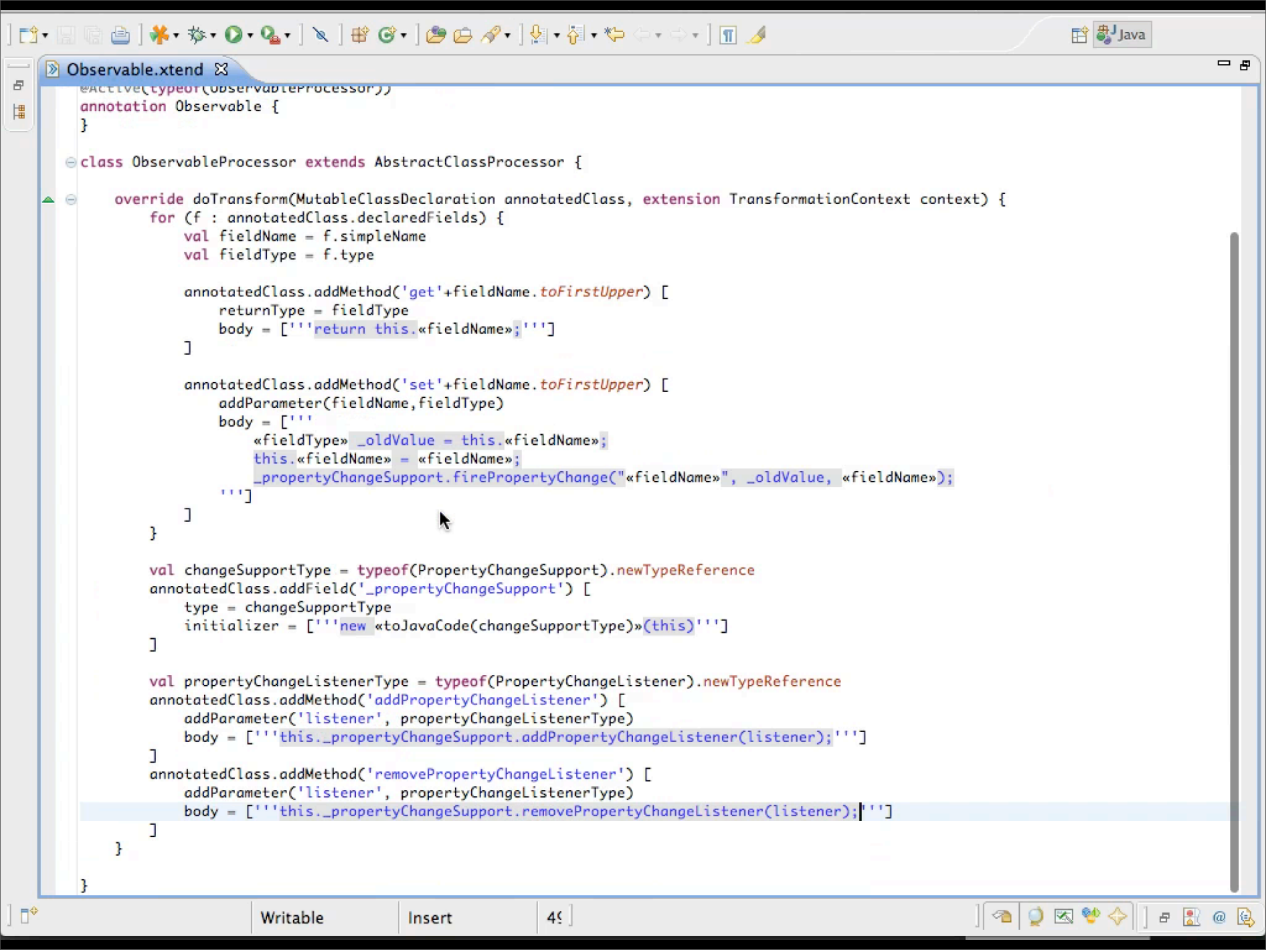
```
package observable

import org.eclipse.xtend.lib.macro.AbstractClassProcessor
import org.eclipse.xtend.lib.macro.Active
import org.eclipse.xtend.lib.macro.declaration.MutableClassDeclaration
import org.eclipse.xtend.lib.macro.TransformationContext

@Active(typeof(ObservableProcessor))
annotation Observable {
}

class ObservableProcessor extends AbstractClassProcessor {

    override doTransform(MutableClassDeclaration annotatedClass, extension TransformationContext context) {
        
    }

}
```



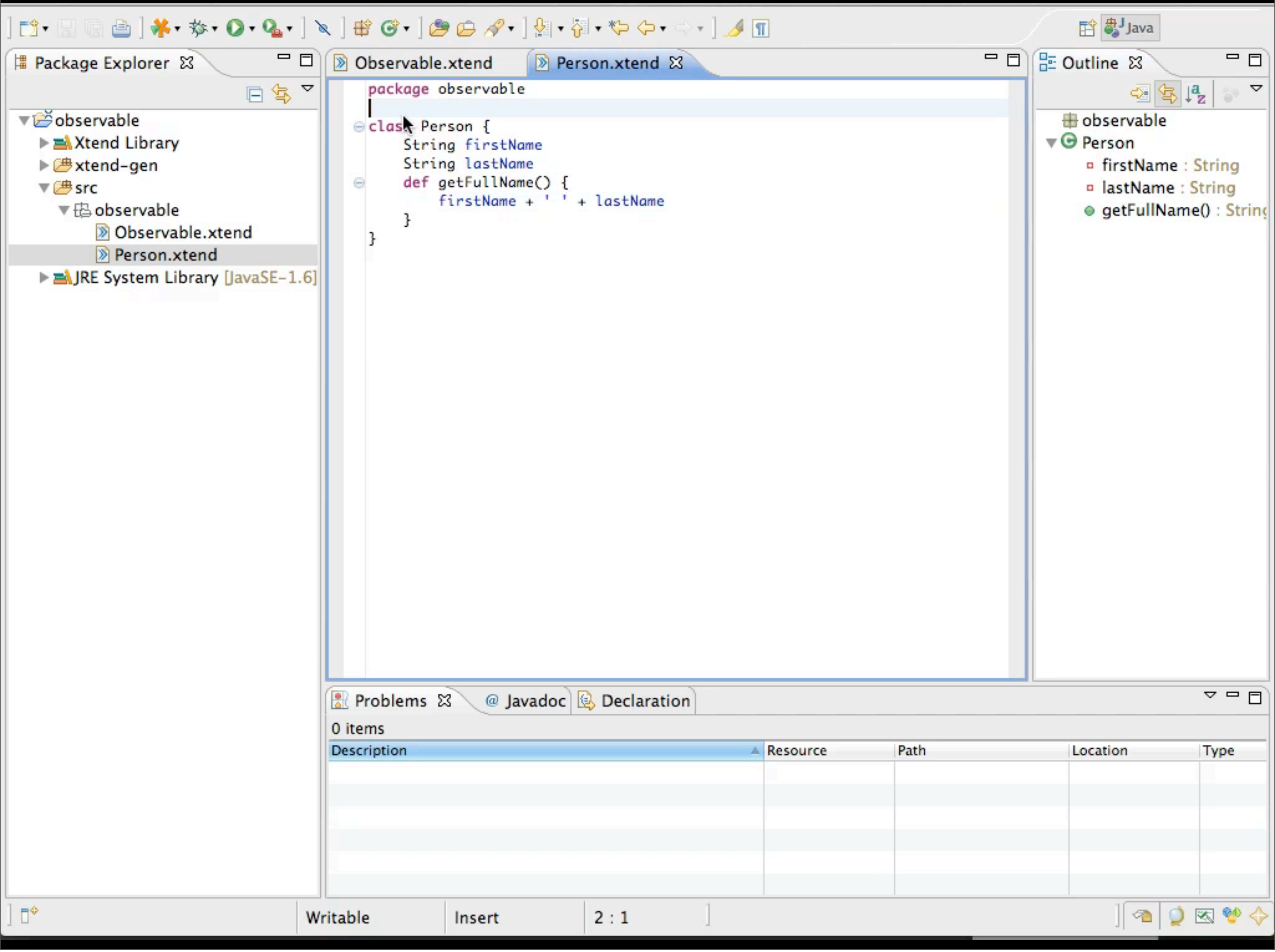
```
package observable

import org.eclipse.xtend.lib.macro.AbstractClassProcessor
import org.eclipse.xtend.lib.macro.Active
import org.eclipse.xtend.lib.macro.declaration.MutableClassDeclaration
import org.eclipse.xtend.lib.macro.TransformationContext

@Active(typeof(ObservableProcessor))
annotation Observable {
}

class ObservableProcessor extends AbstractClassProcessor {

    override doTransform(MutableClassDeclaration annotatedClass, extension TransformationContext context) {
        for (f : annotatedClass.declaredFields) {
            val fieldName = f.simpleName
            val fieldType = f.type

            annotatedClass.addMethod('get'+fieldName.toFirstUpper) [
                returnType = fieldType
                body = ['''return this.«fieldName»;''']
            ]

            annotatedClass.addMethod('set'+fieldName.toFirstUpper) [
                addParameter(fieldName,fieldType)
                body = ['''
                    «fieldType» _oldValue = this.«fieldName»;
                    this.«fieldName» = «fieldName»;
                    _propertyChangeSupport.firePropertyChange("«fieldName»", _oldValue, «fieldName»);
                ''']
            ]
        }
    }

}
```



```
package observable

import org.eclipse.xtend.lib.macro.AbstractClassProcessor
import org.eclipse.xtend.lib.macro.Active
import org.eclipse.xtend.lib.macro.declaration.MutableClassDeclaration
import org.eclipse.xtend.lib.macro.TransformationContext
import java.beans.PropertyChangeSupport

@Active(typeof(ObservableProcessor))
annotation Observable {
}

class ObservableProcessor extends AbstractClassProcessor {

    override doTransform(MutableClassDeclaration annotatedClass, extension TransformationContext context) {
        for (f : annotatedClass.declaredFields) {
            val fieldName = f.simpleName
            val fieldType = f.type

            annotatedClass.addMethod('get'+fieldName.toFirstUpper) [
                returnType = fieldType
                body = ['''return this.«fieldName»;''']
            ]

            annotatedClass.addMethod('set'+fieldName.toFirstUpper) [
                addParameter(fieldName,fieldType)
                body = ['''
                    «fieldType» _oldValue = this.«fieldName»;
                    this.«fieldName» = «fieldName»;
                    _propertyChangeSupport.firePropertyChange("«fieldName»", _oldValue, «fieldName»);
                ''']
            ]
        }

        val changeSupportType = typeof(PropertyChangeSupport).newTypeReference
        annotatedClass.addField('_propertyChangeSupport') [
            type = changeSupportType
            initializer = ['''new «toJavaCode(changeSupportType)»(this)''']
        ]
    }

}
```



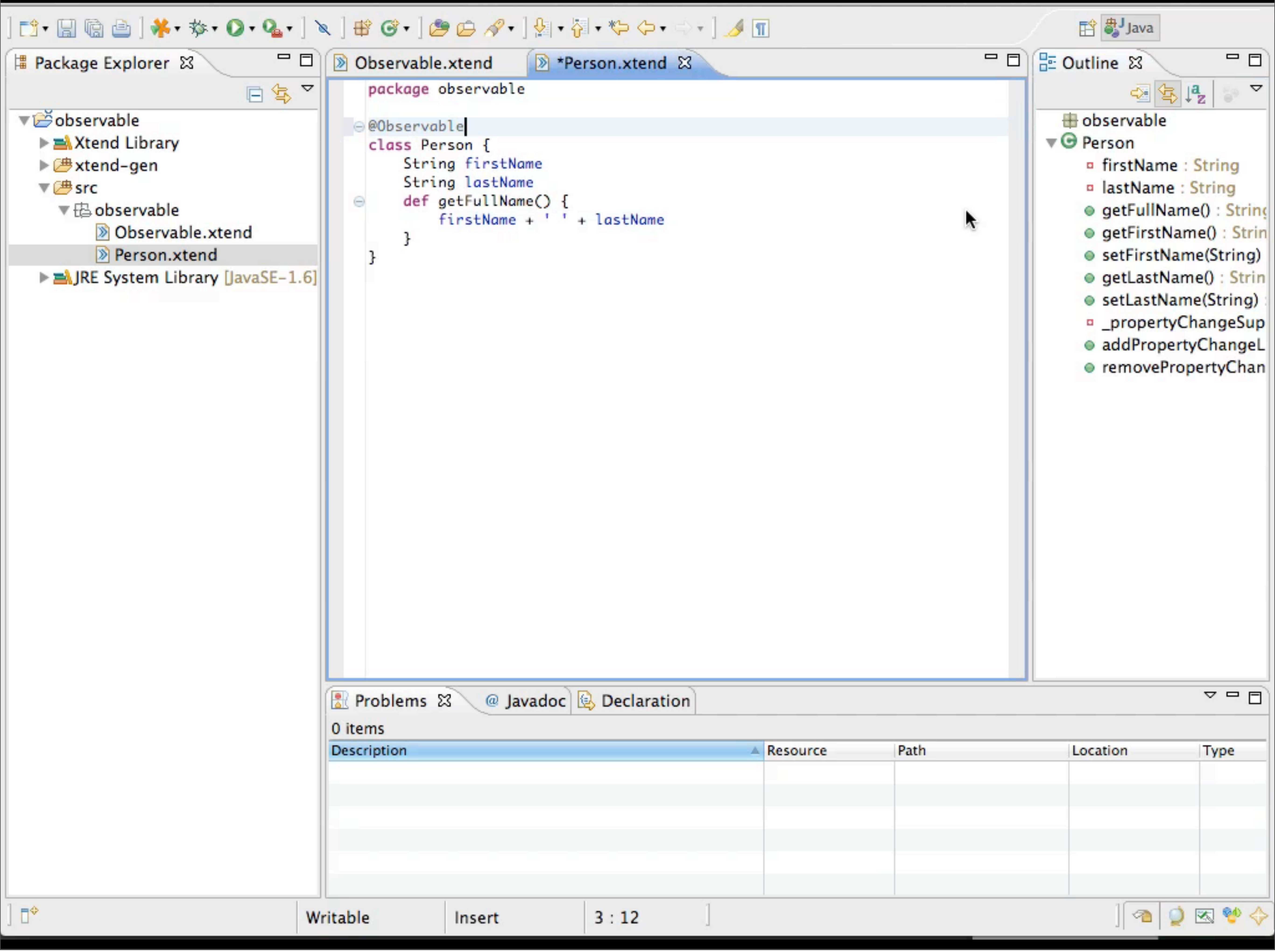
```
@Active(typeof(ObservableProcessor))
annotation Observable {
}

class ObservableProcessor extends AbstractClassProcessor {

    override doTransform(MutableClassDeclaration annotatedClass, extension TransformationContext context) {
        for (f : annotatedClass.declaredFields) {
            val fieldName = f.simpleName
            val fieldType = f.type

            annotatedClass.addMethod('get'+fieldName.toFirstUpper) [
                returnType = fieldType
                body = ['''return this.«fieldName»;''']
            ]

            annotatedClass.addMethod('set'+fieldName.toFirstUpper) [
                addParameter(fieldName,fieldType)
                body = ['''
                    «fieldType» _oldValue = this.«fieldName»;
                    this.«fieldName» = «fieldName»;
                    _propertyChangeSupport.firePropertyChange("«fieldName»", _oldValue, «fieldName»);
                ''']
            ]
        }

        val changeSupportType = typeof(PropertyChangeSupport).newTypeReference
        annotatedClass.addField('_propertyChangeSupport') [
            type = changeSupportType
            initializer = ['''new «toJavaCode(changeSupportType)»(this)''']
        ]

        val propertyChangeListenerType = typeof(PropertyChangeListener).newTypeReference
        annotatedClass.addMethod('addPropertyChangeListener') [
            addParameter('listener', propertyChangeListenerType)
            body = ['''this._propertyChangeSupport.addPropertyChangeListener(listener);''']
        ]
        annotatedClass.addMethod('removePropertyChangeListener') [
            addParameter('listener', propertyChangeListenerType)
            body = ['''this._propertyChangeSupport.removePropertyChangeListener(listener);''']
        ]
    }

}
```


Java
Package Explorer
observable
Xtend Library
xtend-gen
src
observable
Observable.xtend
Person.xtend
JRE System Library [JavaSE-1.6]
Observable.xtend
Person.xtend
Outline
observable
Person
firstName : String
lastName : String
getFullName() : String
Problems
Javadoc
Declaration
Writable
Insert
2 : 1

```
package observable

class Person {
    String firstName
    String lastName
    def getFullName() {
        firstName + ' ' + lastName
    }
}
```

0 items

Description	Resource	Path	Location	Type



Java
Package Explorer
observable
Xtend Library
xtend-gen
src
observable
Observable.xtend
Person.xtend
JRE System Library [JavaSE-1.6]
Observable.xtend
*Person.xtend
Outline
observable
Person
firstName : String
lastName : String
getFullName() : String
getFirstName() : Strin
setFirstName(String)
getLastName() : Strin
setLastName(String)
_propertyChangeSup
addPropertyChangeL
removePropertyChan
Problems
Javadoc
Declaration
0 items
Description
Resource
Path
Location
Type
Writable
Insert
3 : 12

```
package observable

@Observable
class Person {
    String firstName
    String lastName
    def getFullName() {
        firstName + ' ' + lastName
    }
}
```

DEMO