Moldable analysis with Moose

@chis_andrei
feenk.com
package org.argounl.ui;

import java.awt GridBagConstraints;
import java.awt.Insets;

import javax.swing.JTextArea;
import javax.swing.JTextField;

/**
 * Utility class for easy alignment of components in a GridBagLayout.
 * @author drahmann
 */
public final class GridBagUtils {
    /**
     * Constant representing text alignment.
     */
    public static final int CENTER = 2;

    /**
     * Constant representing text alignment.
     */
    public static final int LEFT = 0;

    /**
     * Constant representing text alignment.
     */
    public static final int RIGHT = 1;

    private static Insets buttonInsets;
    private static Insets captionInsets;
    private static Insets inputFieldInsets;
    private static int rowDistance;

    static {
        rowDistance = 2;
        buttonInsets = new Insets(rowDistance, 5, rowDistance, 5);
        captionInsets = new Insets(rowDistance + 2, 5, rowDistance, 5);
        inputFieldInsets = new Insets(rowDistance, 0, rowDistance, 0);
    }

    /**
     * Returns GridBagConstraints suitable for a button.
     */
    public static GridBagConstraints createButtonConstraints() {
        return new GridBagConstraints(0, 0, 1, 1, 1.0, 1.0, GridBagConstraints.CENTER,
                                     GridBagConstraints.VERTICAL, GridBagConstraints.BOTH, 0, 0);  // column, row, width, height, weight, fill, anchor, grid type, grid width, grid height
    }

    /**
     * Returns GridBagConstraints suitable for a caption.
     */
    public static GridBagConstraints createCaptionConstraints() {
        return new GridBagConstraints(0, 0, 1, 1, 1.0, 1.0, GridBagConstraints.CENTER,
                                     GridBagConstraints.VERTICAL, GridBagConstraints.BOTH, 0, 0);  // column, row, width, height, weight, fill, anchor, grid type, grid width, grid height
    }

    /**
     * Returns GridBagConstraints suitable for an input field.
     */
    public static GridBagConstraints createInputConstraints() {
        return new GridBagConstraints(0, 0, 1, 1, 1.0, 1.0, GridBagConstraints.CENTER,
                                     GridBagConstraints.VERTICAL, GridBagConstraints.BOTH, 0, 0);  // column, row, width, height, weight, fill, anchor, grid type, grid width, grid height
    }
}
Decide knowingly
Humane assessment is the method for making software engineering decisions. It helps your team make decisions that are aligned with your business goals.
hypothesize -> apply analysis
confident? -> interpret

Diagram: hypothesize -> apply analysis
confident? -> interpret
hypothesize -> existing analysis? -> apply analysis

confident? -> interpret
hypothesize

existing analysis?

apply analysis

craft analysis

confident?

interpret
Question: Existing analysis? 

Hypothesize 

Confident? 

Interpret 

Apply analysis 

Craft analysis
Moose is a platform for software and data analysis. It helps programmers craft custom analyses cheaply. It's based on Pharo and it's open source under BSD/MIT.

Install  Follow @moosetechnology

moosetechnology.org
case study:
investigating deprecations
@Deprecated
public class Token
    extends PackedTokenAttributeImpl
    implements FlagsAttribute, PayloadAttribute
{

    private int flags;
    private BytesRef payload;

    ...
}

A selection boolean statement using 'each' to refer to each element.
self select: [ :each |
    each isAnnotatedWith: #Deprecated ]
self select: [:each | each isAnnotatedWith: #Deprecated]
self select: [ :each | 
  each isAnnotatedWith: #Deprecated and: [ 
    each clientTypes isEmpty ] ]
self select: [ :each |
  (each isAnnotatedWith: #Deprecated) and: [
    each clientTypes isEmpty ] ]
self select: [:each |
  (each isAnnotatedWith: #Deprecated) and: [
    each clientTypes isNotEmpty ] ]
classes := self select: [ :each |
    each isAnnotatedWith: #Deprecated and: [ each clientTypes isEmpty ] ]
classes := self select: [ :each | 
    each isAnnotatedWith: #Deprecated and: [ 
    each clientTypes isEmpty ] ]

view := RTMondrian new.
view shape circle
    if: [ :c | c isAnnotatedWith: #Deprecated ]
        color: Color red.
view nodes:
    (classes,
        (classes flatCollect: #clientTypes)) asSet.
view edges connectFromAll: #clientTypes.
view layout force.
classes := self select: [:each | (each isAnnotatedWith: #Deprecated) and: [ each clientTypes isNotEmpty ] ].

view := RTMondrian new.
view shape circle if: [:c | c isAnnotatedWith: 'Deprecated' ]
   color: Color red.
view nodes: (classes , (classes flatCollect: #clientTypes)) asSet.
view edges connectFromAll: #clientTypes.
view layout force.
view view pushBackEdges.
view
view deprecated clients
deprecated := (self flatCollect: #methods) select: [:aMethod | aMethod clientTypes notEmpty ].
clients := (deprecated collect: [:aMethod|aMethod clientMethods reject: [:each | each parentType = aMethod parentType ]] ) flatten.
view := RTMondrian new.
view shape rectangle
  color: Color veryLightGray muchLighter;
if: [:each | each isAnnotatedWith: 'Deprecated'] color: Color lightRed;
ifElement: [:each | each model container mooseNameWithDots beginsWith: 'org.apache.solr' ] bordercolor: [ Color green ];
ifElement: [:each | each model container mooseNameWithDots beginsWith: 'org.apache.lucene' ] bordercolor: [ Color blue ].
view nodes: (self , (self flatCollect: #clientTypes)) asSet
forEach: [:aClass |nodes| view shape circle size: [:each | (each parentType isAnnotatedWith: 'Deprecated') ifTrue: [:true | each clientMethods intersection: clients] size max: 5 ] ifFalse: [ 5 ].

nodes := view nodes: (aclass methods intersection: deprecated, clients).
nodes @ (RTHighlightable new highlightElements: [:element | element view edges select: [:e | e model key = element model ] ]).
view layout grid ].
(view edges connectFromAll: #clientTypes) do: [:anEdge |
anEdge trachelShape pushBack ].
view shape line color: Color transparent.
view edges notUseInLayout;
source: deprecated connectFrom: #yourself toAll: #clientMethods.
view layout force strength: 0.15; charge: -350; length: 70.
| view deprecated clients |
| deprecated := (self flatCollect: #methods) select: [ :aMethod | aMethod clientTypes notEmpty ].
| clients := (deprecated collect: [:aMethod|aMethod clientMethods reject: [:each | each parentType = aMethod parentType ]] ) flatten.
| view := RTMondrian new.
| view shape rectangle
| color: Color veryLightGray muchLighter;
| if: [ :each | each isAnnotatedWith: 'Deprecated' ] color: Color lightRed;
| ifElement: [ :each | each model container mooseNameWithDots beginsWith: 'org.apache.solr' ] borderColor: [ Color green ];
| ifElement: [ :each | each model container mooseNameWithDots beginsWith: 'org.apache.lucene' ] borderColor: [ Color blue ].
| view nodes: (self , (self flatCollect: #clientTypes)) asSet
| forEach: [:aClass | nodes view shape circle size: [ :each | (each parentType isAnnotatedWith: 'Deprecated') ifTrue: [ (each clientMethods intersection: clients) size max: 5 ] ifFalse: [ 5 ] ].
| nodes := view nodes: (aClass methods intersection: deprecated, clients).
| nodes @ (RTHighlightable new highlightElements: [ :element | element view edges select: [:e | e model key = element model ]] ).
| view layout grid ].
| (view edges connectFromAll: #clientTypes) do: [ :anEdge |
| anEdge trachelShape pushBack ].
| view shape line color: Color transparent.
| view edges notUseInLayout;
| source: deprecated connectFrom: #yourself toAll: #clientMethods.
| view layout force strength: 0.15; charge: -350; length: 70.
/**
 * @param input input tokenstream
 * @param synonyms synonym map
 * @param ignoreCase case-folds input for matching with \{@link Character#toLowerCase(int)}
 * Note, if you set this to true, it's your responsibility to lowercase
 * the input entries when you create the \{@link SynonymMap}
 */

public SynonymFilter(TokenStream input, SynonymMap synonyms, boolean ignoreCase) {
    super(input);
    this.synonyms = synonyms;
    this.ignoreCase = ignoreCase;
    this.fst = synonyms.fst;
    if (fst == null) {
        throw new IllegalArgumentException("fst must be non-null");
    }
    this.fstReader = fst.getBytesReader();

    // Must be 1+ so that when roll buffer is at full
    // lookahead we can distinguish this full buffer from
    // the empty buffer:
    rollBufferSize = 1+synonyms.maxHorizontalContext;

    futureInputs = new PendingInput(rollBufferSize);
    futureOutputs = new PendingOutputs(rollBufferSize);
    for (int pos=0; pos<rollBufferSize; pos++) {
        futureInputs[pos] = new PendingInput();
        futureOutputs[pos] = new PendingOutputs();
    }
}
```java
self select: [:each |
  (each isAnnotatedWith: #Deprecated) and: [
    each clientTypes isNotEmpty ]
]
view deprecated clients
deprecated := (self flatCollect: #methods)
select: [:aMethod |
  aMethod clientTypes notEmpty ].
clients := (deprecated collect: [:aMethod aMethod clientMethods reject: [
  :each | each parentType = aMethod parentType ] ])
flattened.
view := RTMondrian new.
view shape rectangle
  color: Color veryLightGray muchLighter;
if: [:each | each isAnnotatedWith: 'Deprecated']
  color: Color lightRed;
  ifelement: [:each |
    each model container
      mooseNameWithDots
        beginsWith:
          'org.apache.solr'
      borderColor: [ Color green ];
```

```java
/**
 * @param input input tokenstream
 * @param synonyms synonym map
 * @param ignoreCase case-folds input for matching with @link Character#toLowerCase(int).
 * Note, if you set this to true, it’s your responsibility to lowercase the input entries when you create the @link SynonymMap
 */
public SynonymFilter(TokenStream input, SynonymMap synonyms,
  boolean ignoreCase) {
  super(input);
  this.synonyms = synonyms;
  this.ignoreCase = ignoreCase;
  this.fst = synonyms.fst;
  if (fst == null) {
    throw new
      IllegalArgumentException("fst must be non-null");
  }
  this.fstReader =
    fst.getReader();
  // Must be 1+ so that when roll buffer is at full
```
case study:
towards splitting an angular application
module

component

injectable

template
moduleX.js
angular.module('moduleX', [])
  .config(function config($routeProvider) {
    $routeProvider.when('urlX', {
      templateUrl: 'moduleX.tpl.html', ...
    })
  })
  .component('someComponentA', ...)
**moduleX.js**

```javascript
angular.module('moduleX', [])
  .config(function config($routeProvider) {
    $routeProvider.when('urlX', {
      templateUrl: 'moduleX.tpl.html', ...
    }) })
  .component('someComponentA' ...)
```

**moduleX.tpl.html**

```html
<some-component-b>
  <div> ... </div>
</some-component-b>
```
moduleX.js
angular.module('moduleX', []).
  .config(function config($routeProvider) {
    $routeProvider.when('urlX', {
      templateUrl: 'moduleX.tpl.html', ...
    })
  })
  .component('someComponentA', ...)
moduleX.js
angular.module('moduleX', [])
  .config(function config($routeProvider) {
    $routeProvider.when('urlX', {
      templateUrl: 'moduleX.tpl.html'
    })
  });
  .component('someComponentA', ...)

moduleX.tpl.html
<some-component-b>
  <div> ... </div>
</some-component-b>

moduleY.js
angular.module('moduleY', [])
  .directive('someComponentB', function () {
    return {
      templateUrl: 'moduleY.componentB.tpl.html'
    }
  });

moduleY.componentB.tpl.html
<some-component-a ...
  ...
</some-component-a>
```javascript
moduleX.js
angular.module('moduleX', [])
  .config(function config($routeProvider) {
    $routeProvider.when('urlX', {
      templateUrl: 'moduleX.tpl.html', ...
    })
  });

.modulex.tpl.html
<some-component-b>
  <div> ... </div>
</some-component-b>

moduleY.js
angular.module('moduleY', [])
  .directive('someComponentB', function () {
    return {
      templateUrl: 'moduleY.componentB.tpl.html', ...
    };
  });

moduleY.componentB.tpl.html
<some-component-a ...
  ...
</some-component-a>
```
view := RTMondrian new.

view nodes:
    self allAngularTemplates, self allAngularComponents.
view := RTMondrian new.
view nodes:
    self allAngularTemplates, self allAngularComponents.

view shape line color: Color gray.
view edges
    source: self allAngularTemplates
    connectFrom: #yourself
toAll: #includedTemplates.
view shape line color: Color gray.
view edges
    source: self allAngularTemplates
    connectFrom: #yourself
toAll: #includedComponents.
view layout force.
self allAngularComponents select: [:each | each parentModule beginsWith: 'my.component'].
self allAngularComponents select: [:each | each parentModule name beginsWith: 'my.component' ].
self allAngularComponents select: [:each |
    each parentModule name beginsWith: 'partner'.]
case study:
evaluating a refactoring path
view := RTMondrian new.
class := UITheme.
view shape ellipse fillColor: Color gray.
view nodes: class methods.
view layout force.
view := RTMondrian new.
class := UITheme.
view shape ellipse fillColor: Color gray.
ignored := #(icons settings textFont initializeForms).
view nodes: (class methods reject: [:each | (ignored includes: each selector) or: [ each category = #’morph creation’ ] ]) .
view edges
  source: class methods
  connectFrom: #yourself toAll: [:each | each parseTree allChildren select: [:n | n isMessage and: [ n isSelfSend and: [ class methods anySatisfy: [:m | m selector = n selector ] ] ] ] thenCollect: [:n | class methods detect: [:m | m selector = n selector ] ]].
view layout force.
class methodsToClassesAssoc SelectorsToClasses View mElements cElements

<table>
<thead>
<tr>
<th>class methodsToClassesAssoc selectorsToClasses view mElements cElements</th>
</tr>
</thead>
<tbody>
<tr>
<td>class := UI::Theme.</td>
</tr>
<tr>
<td>methodsToClassesAssoc := class methods collect: [:each</td>
</tr>
<tr>
<td>(each senders collectAsSet: #methodClass) select: [:c</td>
</tr>
<tr>
<td>selectorsToClasses := methodsToClassesAssoc asDictionary.</td>
</tr>
<tr>
<td>view := RTM::Network new.</td>
</tr>
<tr>
<td>view shape ellipse fillColor: Color gray.</td>
</tr>
<tr>
<td>mElements := view nodes: (class methods reject: [:each</td>
</tr>
<tr>
<td>(icons settings textStyle backgroundColor initialize) includes: each</td>
</tr>
<tr>
<td>selector) or: [ each category = '#morph creation' ]] collect: #selector].</td>
</tr>
<tr>
<td>view shape ellipse fillColor: Color red.</td>
</tr>
<tr>
<td>view edges objects: class selectors; connectFrom: #yourself toAll: [:each</td>
</tr>
<tr>
<td>(class methodNamed: each) parseTree allChildren select: [:n</td>
</tr>
<tr>
<td>n isMessage and: [:n isSelfSend ]] thenCollect: [:n</td>
</tr>
<tr>
<td>view shape line color: (Color red alpha: 0.2); width: 1.</td>
</tr>
<tr>
<td>view edges fromElements: mElements; toElements: cElements;</td>
</tr>
<tr>
<td>connectFrom: [:each</td>
</tr>
<tr>
<td>view layout force.</td>
</tr>
</tbody>
</table>

view