Comp 110/401
Composite Annotations

Instructor: Prasun Dewan

Prerequisite

- Composite Objects Shapes

.

Interpreted How?

```
public interface RectangleWithPoint {
  public int getX();
  public int getY();
  public void getWidth();
  public void getHeight();
  public Point getPoint();
}
```

Object interpreted as either an atomic rectangle or an atomic point depending on whether Point or Rectangle in the name gets precedence

Bad programming as code for defining Rectangle cannot be used in situations where rectangle without point is needed

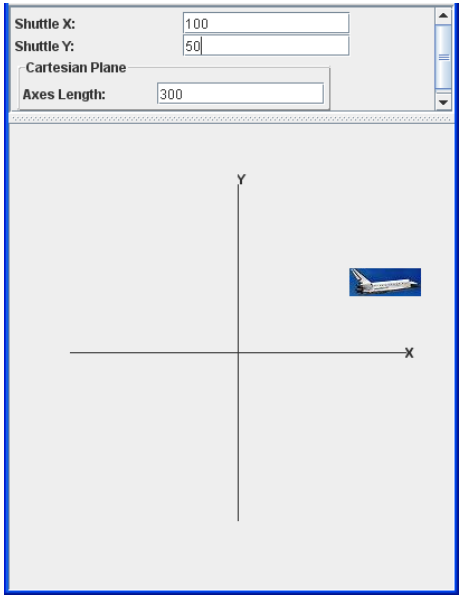
Reusable Design

```
public interface Rectangle {
  public int  getX();
  public int  getY();
  public void getWidth();
  public void getHeight();
}
```

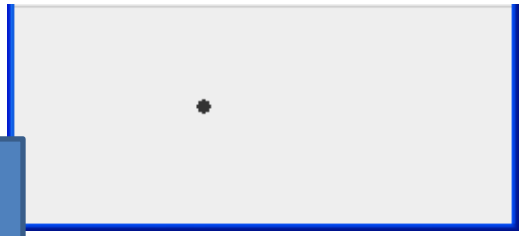
```
public interface RectangleWithPoint {
  public Rectangle getRectangle();
  public Point getPoint();
}
```

ObjectEditor Graphics Rules

```
public interface ShuttleLocation {
  public FancyCartesianPlane
        getCartesianPlane();
  public ImageLabel getShuttleLabel();
  public int  getShuttleX();
  public void setShuttleX(int newVal);
  public int  getShuttleY();
  public void setShuttleY(int newVal);
}
```




```
public interface NotAPoint {
  public FancyCartesianPlane
        getCartesianPlane();
  public ImageLabel getShuttleLabel();
  public int  getX();
  public void setX(int r
  public int  getY();
  public void setY(int r
  public Point getLocati
}
```



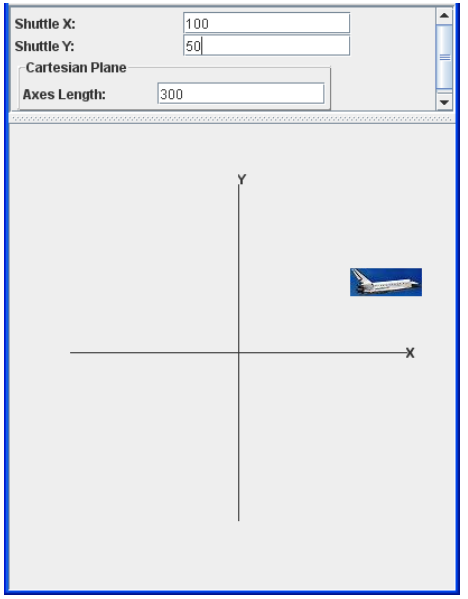
Type interpreted as a Point as its name contains "Point" and has X and Y Properties

IS_ATOMIC_SHAPE CLASS ANNOTATION

```
import util.annotations.IsAtomicShape;
@IsAtomicShape(false)
//same as AShuttleLocation except interface name
public class AShuttleLocationImplementingABadlyNamedInterface
                              implements NotAPoint {
```

Annotation is like a comment except it is typed and available at runtime

IsAtomicShape(false) before class name says do not interpret the class as an atomic shape (it can be a composition)



ShuttleLocation Alternatives

```
public interface ShuttleLocation {
        public FancyCartesianPlane getCartesianPlane();
        public ImageLabel getShuttleLabel();
        public int getShuttleX();
        public void setShuttleX(int newVal);
        public int getShuttleY();
        public void setShuttleY(int newVal);
}
```

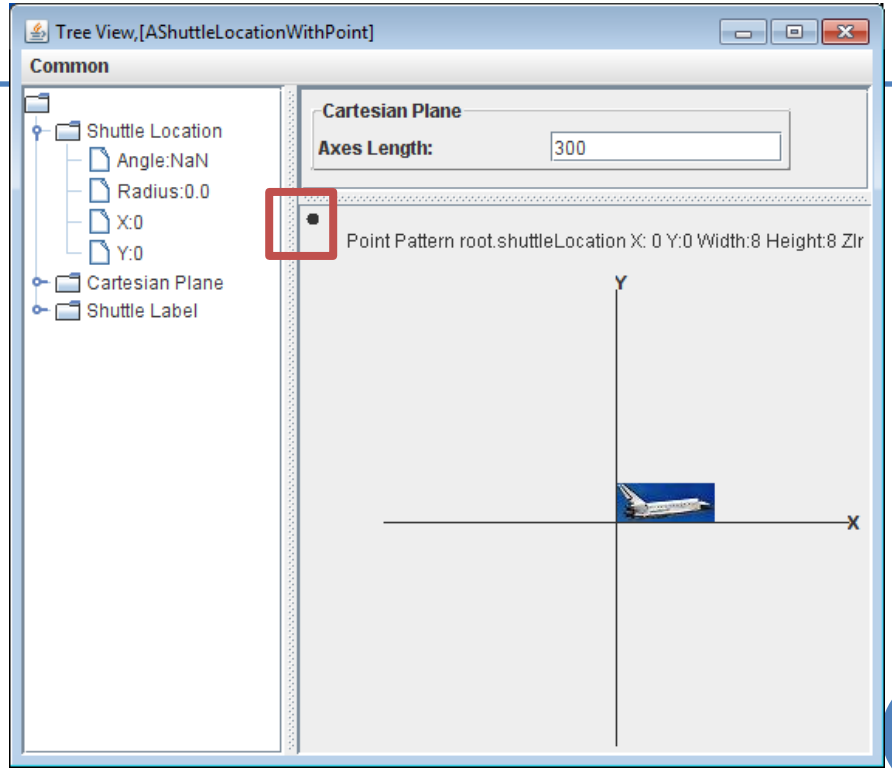
```
public interface ShuttleLocationWithPoint {
        public FancyCartesianPlane getCartesianPlane();
        public ImageLabel getShuttleLabel();
        public Point getShuttleLocation();
        public void setShuttleLocation(Point newVal);
}
```

Point Property

```
public class AShuttleLocationWithPoint implements
        ShuttleLocationWithPoint {
    public Point getShuttleLocation() {
        return shuttleLocation;
    }
    ...
}
```

Does not have X and Y properties, so not a point

But location displayed as a point shape



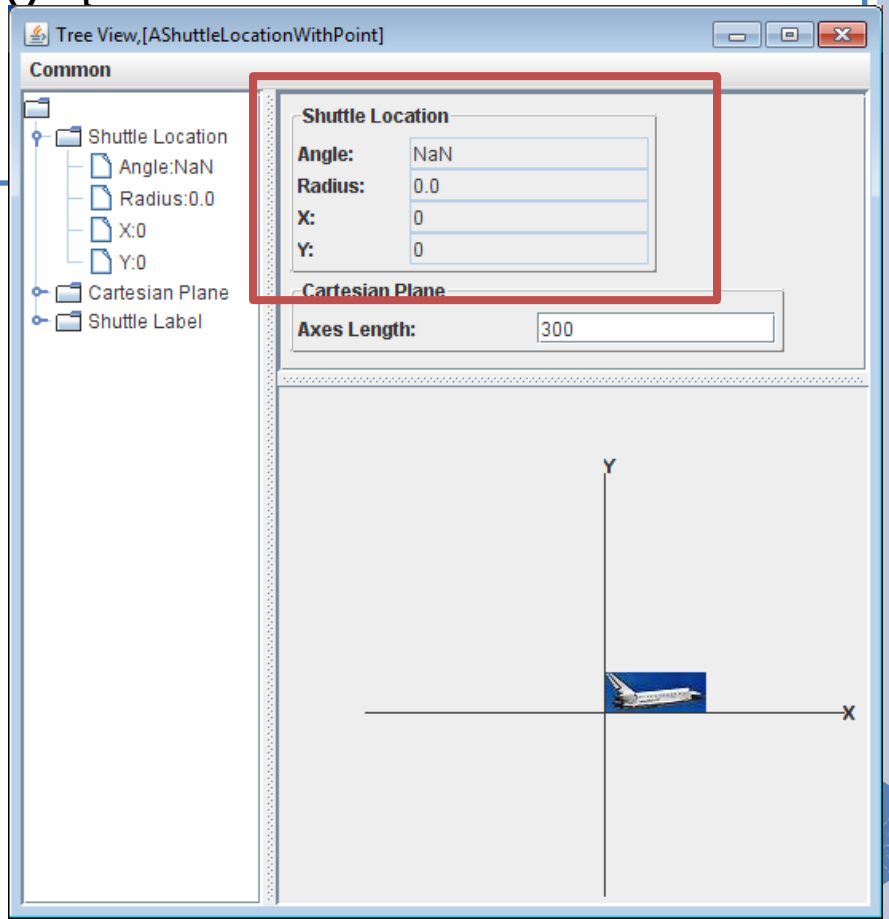
IS_ATOMIC_SHAPE ANNOTATION OF CLASS GETTER

```
import util.annotations.IsAtomicShape;
public class AShuttleLocationWithPoint implements
        ShuttleLocationWithPoint {
    @IsAtomicShape(false)
    public Point getShuttleLocation() {
        return shuttleLocation;
    }
    ...
}
```

IsAtomicShape(false) before getter of a property means property not displayed as an atomic shape

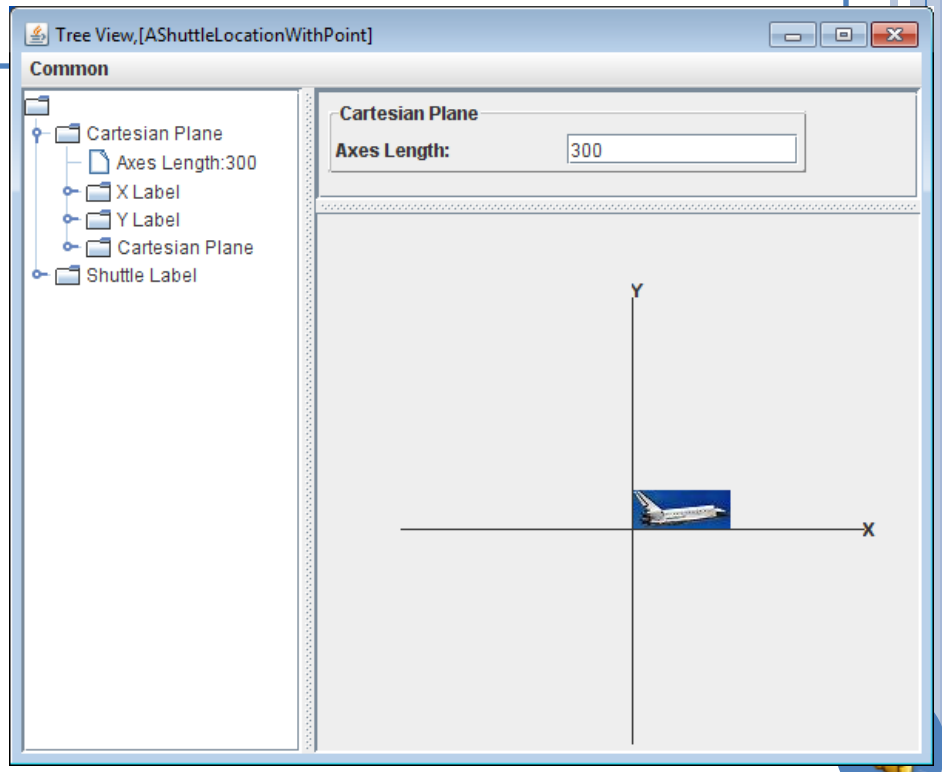
ShuttleLocation is not a textual property displayed in main panel

What if we do not want it in main panel either?



Visible Annotation of Class Getter

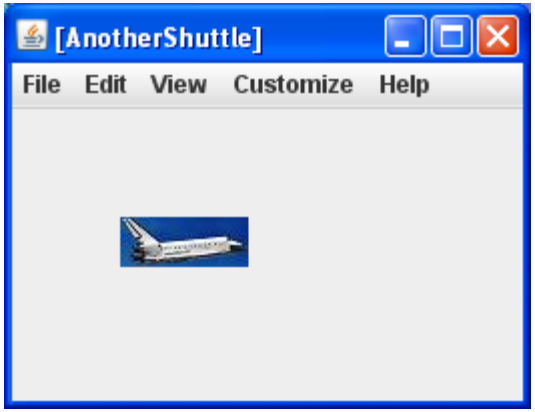
```
import util.annotations.Visible;
public class AShuttleLocationWithPoint implements
        ShuttleLocationWithPoint {
    @Visible(false)
    public Point getShuttleLocation() {
        return shuttleLocation;
    }
    …
}
```



Visible(false) before getter of a property means property not displayed at all

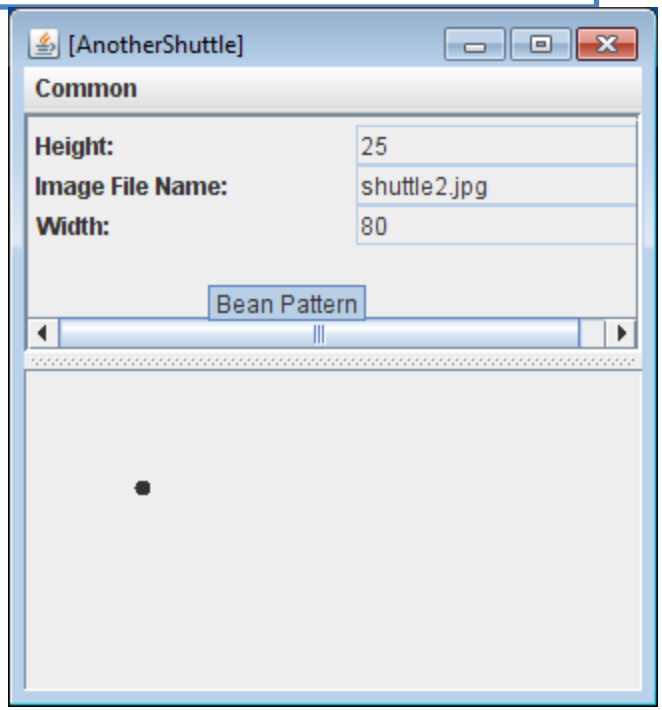
Automatic Label Pattern

```
public class AShuttle implements SpecificImageLabel {
      …
}
```



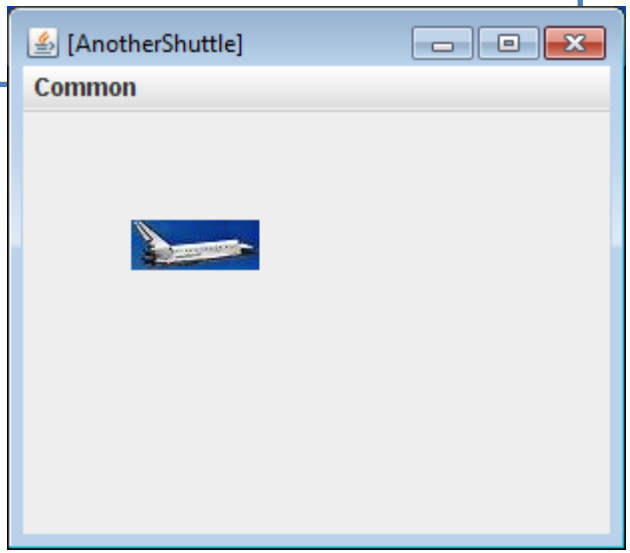
Not Following Automating Label Pattern

```
public class AShuttle implements Shuttle {
      …
}
```



Explicit Pattern Specification

```
import util.annotations.StructurePattern;
import util.annotations.StructurePatternNames;
@StructurePattern(StructurePatternNames.LABEL_PATTERN)
public class AShuttle implements Shuttle{
    …
}
```



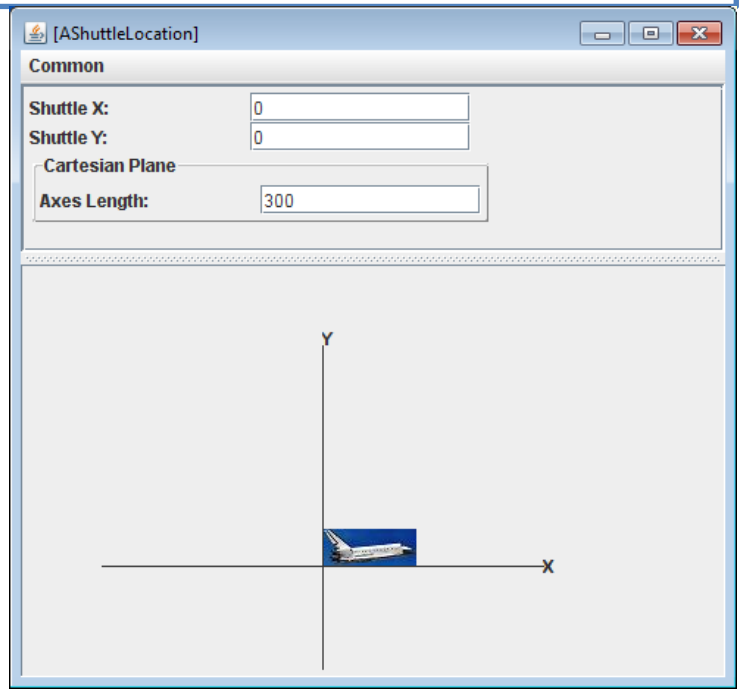
Structure(<PatternName>) before class asserts that the class is following the pattern. ObjectEditor ignores class name and gives warnings if methods do not follow the pattern

Explicit Pattern Specification

```
import util.annotations.StructurePattern;
import util.annotations.StructurePatternNames;
@StructurePattern(StructurePatternNames.LINE_PATTERN)
public class AShuttle implements Shuttle{
    …
}
```

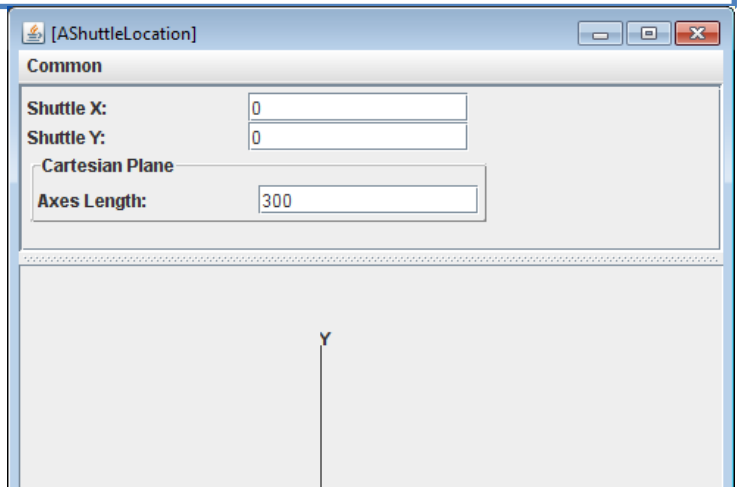
Explicit Pattern Specification

```
public class AShuttleLocation implements ShuttleLocation {
    …
}
```



Explicit Pattern Specification

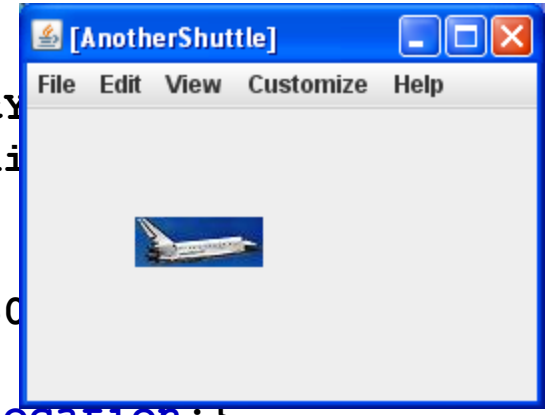
```
import util.annotations.StructurePattern;
import util.annotations.StructurePatternNames;
@StructurePattern(StructurePatternNames.LINE_PATTERN)
public class AShuttleLocation implements ShuttleLocation {
    …
}
```




```
E***lectures.composition.AShuttleLocation not recognized as a shape  as it does not have a location (X,Y or Point)
E***class lectures.composition.AShuttleLocation does not follow declared pattern: Line Pattern
W***Use annotation @util.annotations.StructurePattern(Bean Pattern) for class lectures.composition.AShuttleLocation
W***Use annotation @util.annotations.StructurePattern(Bean Pattern) for class lectures.composition.AFancyCartesianPlane
W***Use annotation @util.annotations.StructurePattern(Label Pattern) for class lectures.composition.ALabel
W***Use annotation @util.annotations.StructurePattern(Bean Pattern) for class lectures.composition.ACartesianPlane
W***Use annotation @util.annotations.StructurePattern(Line Pattern) for class lectures.graphics.ALine
```

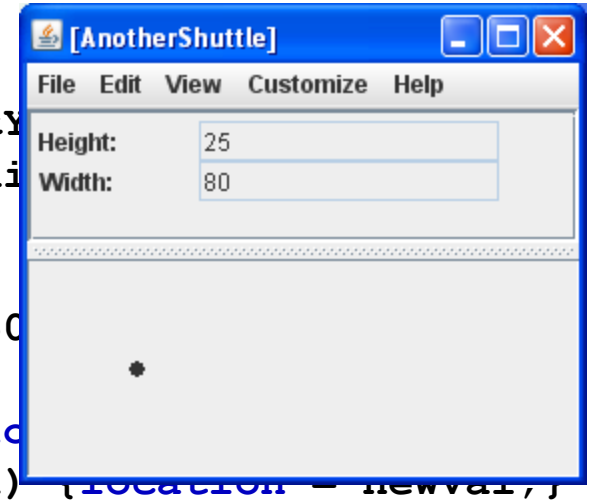
Explicit and Implicit Label Pattern

```
import util.annotations.StructurePattern;
import util.annotations.StructurePatternNames;
@StructurePattern(StructurePatternNames.LABEL_PATTERN)
public class AnotherShuttle implements SpecificImageLabel {
 static final String IMAGE = "shuttle2.jpg";
 static final int WIDTH = 80;
 static final int HEIGHT = 25;
     Point location;
     public AShuttle (int initX, int initY
        location = new ACartesianPoint(ini
     }
     public AShuttle () {
         location = new ACartesianPoint(50
     }
     public Point getLocation() {return location;}
     public void setLocation(Point newVal) {location = newVal;}
     public int getWidth() { return WIDTH;}
     public int getHeight() {return HEIGHT;}
     public String getImageFileName() {return IMAGE;}
}
```



Explicit and Attempted Implicit Label Pattern

```
import util.annotations.StructurePattern;
import util.annotations.StructurePatternNames;
@StructurePattern(StructurePatternNames.LABEL_PATTERN)
public class AnotherShuttle implements SpecificImageLabel {
 static final String IMAGE = "shuttle2.jpg";
 static final int WIDTH = 80;
 static final int HEIGHT = 25;
     Point location;
     public AShuttle (int initX, int initY
        location = new ACartesianPoint(ini
     }
     public AShuttle () {
         location = new ACartesianPoint(50
     }
     public Point getLocation() {return lo
     public void setLocation(Point newVal) {location = newVal;}
     public int getWidth() { return WIDTH;}
     public int getHeight() {return HEIGHT;}
     //public String getImageFileName() {return IMAGE;}
}
```




```
16: Copyright Prasun Dewan, 2010, All rights reserved.

xamples.representation.AnotherShuttle not recognized as a label shape  as it does not have a text or imagefile property
class examples.representation.AnotherShuttle does not follow declared pattern: Label Pattern
```

BMI Calculator Pattern?

```
public class ABMICalculator  {
   public double calculateBMI(double height, double weight) {
      return weight/(height*height);
   }
}
```

BMI Calculator Pattern?

```
@StructurePattern(StructurePatternNames.NO_PATTERN)
public class AnAnnotatedBMICalculator  {
  public double calculateBMI(double height, double weight) {
    return weight/(height*height);
  }
}
```

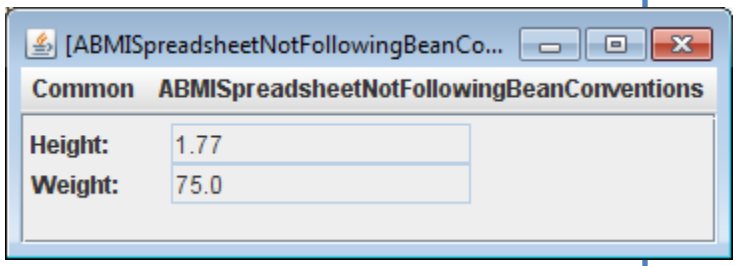
BMI Calculator Pattern?

```
@StructurePattern(StructurePatternNames.BEAN_PATTERN)
public class AnAnnotatedBMICalculator  {
  public double calculateBMI(double height, double weight) {
    return weight/(height*height);
  }
}
```

```
E***Expected one or more programmer-defined properties in class: lectures.functions.ABMICalcul
E***Class:lectures.functions.ABMICalculator does not follow declared pattern: Bean Pattern. Ig
W***Assuming implicit pattern: No Pattern instead of: Bean Pattern
```

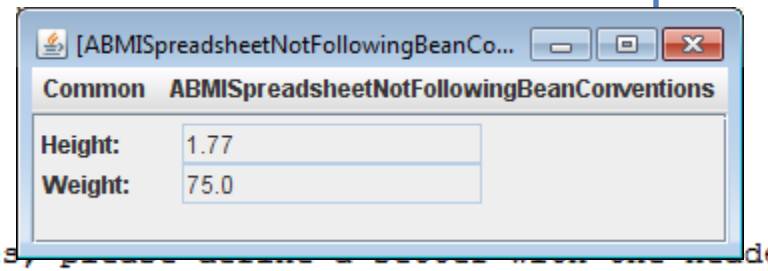
Bean Pattern?

```
@StructurePattern(StructurePatternNames.BEAN_PATTERN)
public class ABMISpreadsheetNotFollowingBeanConventions  {
  double height = 1.77;
  double weight = 75;
  public double getWeight() {
    return weight;
  }
  public void set(double newWeight, double newHeight) {
    weight = newWeight;
    height = newHeight;
  }
  public double getHeight() {
    return height;
  }
  public void setHeight(int newHeight) {
    height = newHeight;
  }
  public double BMI() {
    return weight/(height*height);
  }
}
```



(Editable) Property Name Annotations

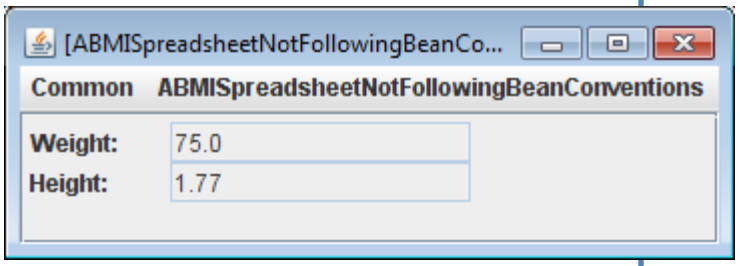
```
@StructurePattern(StructurePatternNames.BEAN_PATTERN)
@PropertyNames({ "Height", "Weight", "BMI"})
@EditablePropertyNames({"Height", "Weight"})
public class ABMISpreadsheetNotFollowingBeanConventions  {
  double height = 1.77;
  double weight = 75;
  public double getWeight() {
    return weight;
  }
  public void set(double newWeight,
    weight = newWeight;
    height = newHeight;
  }
  public double getHeight() {
```




```
E***For property: height in editable property names, please define a setter with the header:
        public void setHeight(double <parameter name>)
E***For property: weight in editable property names, please define a setter with the header:
        public void setWeight(double <parameter name>)
E***For property: BMI in  property names, please define a getter with the header:
        public <T> getBMI()
  public double BMI() {
    return weight/(height*height);
  }
}
```

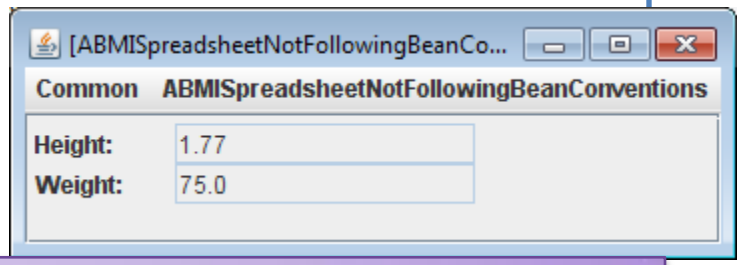
Order of Properties

```
@StructurePattern(StructurePatternNames.BEAN_PATTERN)
@PropertyNames({ "Weight", "Height", "BMI"})
@EditablePropertyNames({"Height", "Weight"})
public class ABMISpreadsheetNotFollowingBeanConventions  {
  double height = 1.77;
  double weight = 75;
  public double getWeight() {
    return weight;
  }
  public void set(double newWeight,
    weight = newWeight;
    height = newHeight;
  }
  public double getHeight() {
    return height;
  }
  public void setHeight(int newHeight) {
    height = newHeight;
  }
  public double BMI() {
    return weight/(height*height);
  }
}
```



Bean Pattern?

```
@StructurePattern(StructurePatternNames.BEAN_PATTERN)
public class ABMISpreadsheetNotFollowingBeanConventions  {
  double height = 1.77;
  double weight = 75;
  public double getWeight() {
    return weight;
  }
  public void set(double newWeight, double newHeight) {
    weight = newWeight;
    height = newHeight;
  }
  public double getHeight() {
    return height;
```



Warning if (editable) properties not declared?

Overhead, chances of mistake low, C# has built in support for properties

Why warning if no structure annotation?

```
}
```

Why Warnings

- Accidental patterns
- Efficiency

Large Search Space

```
public class StructurePatternNames {
  public static final String NO_PATTERN = "No Pattern";
  public static final String BEAN_PATTERN = "Bean Pattern";
  public static final String POINT_PATTERN = "Point Pattern";
  public static final String LINE_PATTERN = "Line Pattern";
  public static final String OVAL_PATTERN = "Oval Pattern";
  public static final String RECTANGLE_PATTERN = "Rectangle
Pattern";
  public static final String ARC_PATTERN = "Arc Pattern";
  public static final String LABEL_PATTERN = "Label Pattern";
  public static final String TEXT_PATTERN = "Text Pattern";
  public static final String STRING_PATTERN = "String Pattern";
  public static final String CURVE_PATTERN = "Curve Pattern";
  public static final String VECTOR_PATTERN = "Vector Pattern";
  public static final String LIST_PATTERN = "List Pattern";
  public static final String STACK_PATTERN = "Stack Pattern";
  public static final String HASHTABLE_PATTERN = "Hashtable
Pattern";
public static final String MAP_PATTERN = "Hashmap Pattern";
  public static final String ENUM_PATTERN = "Enum Pattern";
  …
}
```