**Team 03**

**Project-5 Documentation**

**Members:**

Nachiappan Lakshmanan

Ashutosh Dey

Ashwin Srinivasan

Srinivasan Sundar

Chandan Yadav

**Class Diagram auto generated from Astah:**

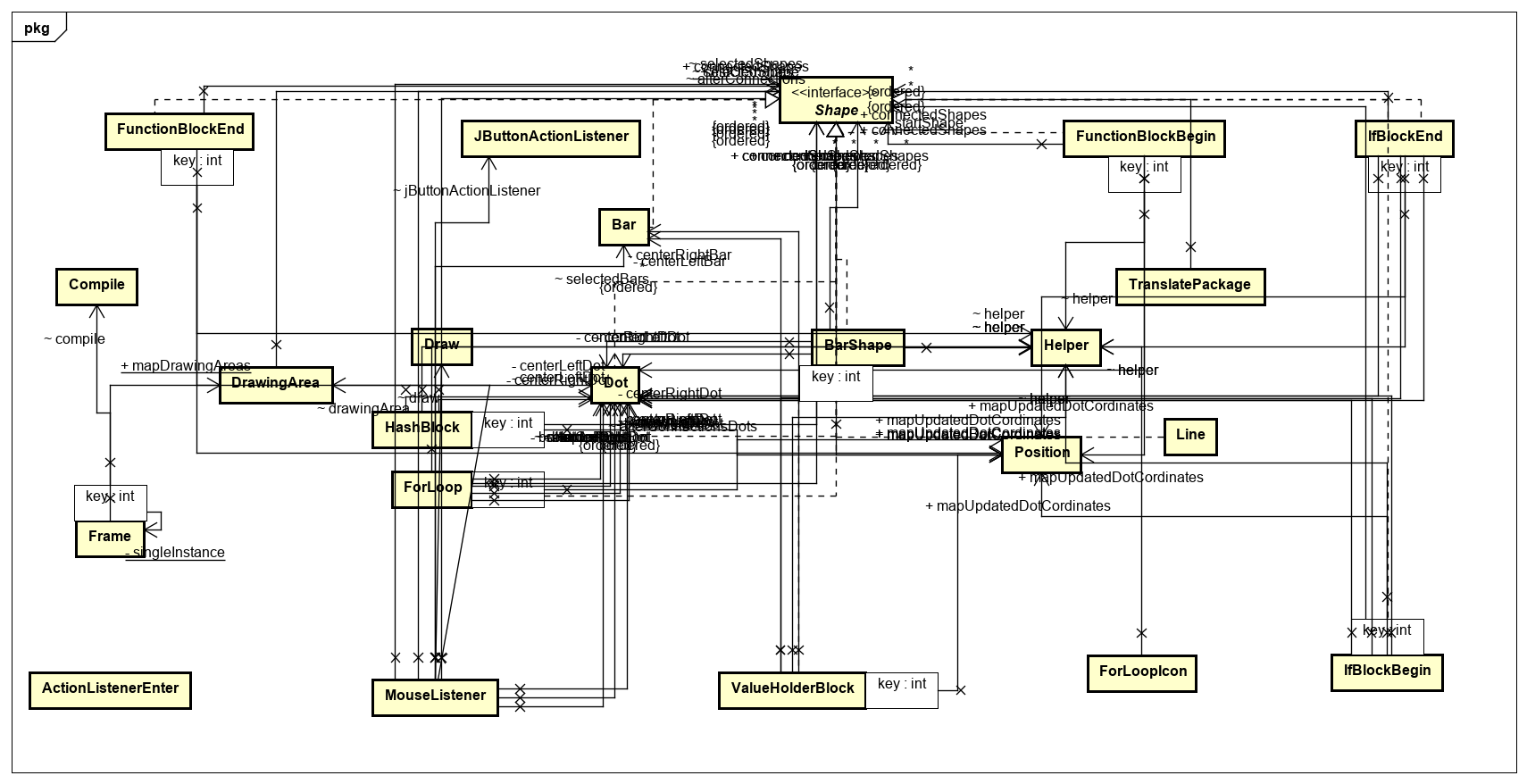


Fig1 : Class diagram for Project five – Team 03

**Robert C Martin metrics calculated for each class:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Class** | **Incoming(Ca)** | **Outgoing(Ce)** | **Abstraction(A)** | **Instability(I)** | **Distance(D)** |
| ActionListenerEnter | 0 | 0 | 0 | 0 | 1 |
| Bar | 2 | 0 | 0 | 0 | 1 |
| BarShape | 0 | 4 | 0 | 1 | 0 |
| BarShapeIcon | 0 | 1 | 0 | 1 | 0 |
| Compile | 1 | 0 | 0 | 0 | 1 |
| Constants | 0 | 0 | 0 | 0 | 1 |
| Dot | 7 | 0 | 0 | 0 | 1 |
|
| Draw | 1 | 1 | 0 | 0.5 | 0.5 |
| DrawingArea | 2 | 1 | 0 | 0.33 | 0.67 |
| ForLoop | 1 | 4 | 0 | 0.8 | 0.2 |
| ForLoopIcon | 0 | 1 | 0 | 1 | 0 |
| Frame | 1 | 3 | 0 | 0.75 | 0.25 |
| FunctionBlockBegin | 0 | 4 | 0 | 1 | 0 |
| FunctionBlockBeginIcon | 0 | 1 | 0 | 1 | 0 |
| FunctionBlockEnd | 0 | 4 | 0 | 1 | 0 |
| FunctionBlockEndIcon | 0 | 1 | 0 | 1 | 0 |
| Helper | 10 | 0 | 0 | 0 | 1 |
|
|
|
|
| IfBlockBegin | 0 | 4 | 0 | 1 | 0 |
|
|
|
| IfBlockBeginIcon | 0 | 1 | 0 | 1 | 0 |
| IfBlockEnd.java | 0 | 4 | 0 | 1 | 0 |
| IfBlockEndIcon.java | 0 | 1 | 0 | 1 | 0 |
|
| InputDialog.java | 0 | 0 | 0 | 0 | 1 |
|
| JButtonActionListener.java | 1 | 2 | 0 | 0.66 | 0.34 |
|
|
|
|
|
|
| Line.java | 0 | 1 | 0 | 1 | 0 |
|  |  |  |  |  |  |
| MouseListener.java | 0 | 7 | 0 | 1 | 0 |
|
|
|
|
|
|
| Position.java | 7 | 0 | 0 | 0 | 1 |
|
|
|
|
|
|
| Shape.java | 9 | 0 | 1 | 0 | 0 |
|
|
|
|
|
|
|
|
| ShapePanel.java | 0 | 9 | 0 | 1 | 0 |
|
|
|
|
|
|
|
|
|
| ValueHolderBlock.java | 0 | 3 | 0 | 1 | 0 |
|
|
| ValueHolderBlockIcon.java | 0 | 1 | 0 | 1 | 0 |
|
| Translate Package | 0 | 1 | 0 | 1 | 0 |
|
| HashBlock | 0 | 5 | 0 | 1 | 0 |
|
| ValueHolderBlock | 0 | 1 | 0 | 1 | 0 |
|

**Graph:**

Instability vs Abstraction:

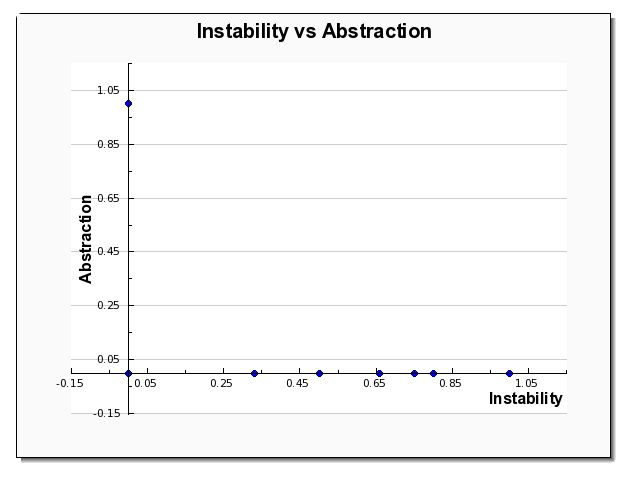


Fig2. Instability vs Abstraction Graph

**There are no useless classes in the Project**

Painful Classes:

1. ActionListenerEnter
2. Bar
3. Compile
4. Constants
5. Dot
6. Input dialog
7. Helper
8. Position

The Drawing Area class has been moved out of the Pain zone by passing the object as parameter in other classes as promised from the previous projects.

**SDP: The below connections do not follow Instability metric.**

* Draw Class to Line Class
* DrawingArea to Frame
* JbuttonActionListener to BarShape, FunctionBlockBegin, FunctionBlockEnd, IfBlockBegin, IfBlockEnd ,ValueHolderBlock

Future developments to fix the violations of SDP in our project:

These stable components on the left have an instability value which is very much less than the component that is dependent on (volatile component). This makes the volatile component difficult to change. The answer to this is to once again invert all the dependencies by applying the dependency inversion principle. We can create an interface or abstract parent, base class and put it in a new component. We can make the stable component depend on the new component and have the volatile component implement the concrete classes. By doing this, the volatile component is now made easier to change. We tried to remove most of the implementation by creating interface or by using it as a parameter in the extended class, but we could only implement it to Drawing area class and remove it from the painful zone.

**Design Patterns used in the Code:**

* Factory Pattern: **JbuttonActionListener Class** : Factory object of Factory shapes.
* Singleton Pattern: **Frame Class**

In our project we plan to fix the violations of SDP by:

We already have an interface shape existing which will be the new component here and the stable class will point to the new component i.e an object of the interface will be passed on as a parameter to the stable class thereby reducing the instability.

For the draw to line, DrawingArea to Frame class we will pass the line object as parameter.

Painful classes Solution:

The following classes are hard to change and inflexible which we were able to find from the analysis done. We were able to change the DrawingArea class from Painful zone to painless zone by implementing it as an interface and increase the abstraction of the class.

In our project:

ActionListenerEnter is a default class that we have to use for creating the input dialog for the user to enter their comment, compile class is for logic handling for validation, we will create abstract classes and make classes implement it so when we need to pass a new object we can add a new implementation and instantiate the interface object with the newly created class. So the classes become flexible and easier to change. They will not have dependencies, a class with less dependencies will become easier to change.