Project 1 Report [Team-2]

Ashwini Tokekar (201405446), Deepak Kathayat (201101213), M.S.Soumya (201450872)

2016-02-15 Mon

Contents

1 Introduction

This document describes in detail an analysis of the codebase given for the Virtual Labs VLSI lab experiment-6 *Schematic design of D-latch and D-flipflop*. It also enlists a comparison of the code metrics of the Java code and the re-factored code in JavaScript.

2 Java Code analysis

The Schematic Design of D-Latch and D-flip-flops was initially written in Java. This section describes the analysis of the code written in Java. The experiment consists of two parts namely the D-flip-flop and the D-Latch. The experiments were to be tested on two types of triggers namely the positive edge trigger and the negative edge trigger. Hence, given code base consists of 4 java files namely $exp6_{\text{flipflop positive}}.java$, $exp6_{\text{flipflop negative}}.java$, $exp6_{\text{latch positive}}.java$ and $exp6_{\text{latch negative}}.java$. Each file consists of around 3500 lines of java code. In the following sections a detailed analysis of the design and the code is narrated.

2.1 Design smells

There are several visible design smells in the existing code. For determining the structure and design of the given code-base we used the *ObjectAid Eclipse Plugin*. The following are the class diagrams of the four java files.

From the diagrams above the following design smells can be inferred:

1. Inappropriate use of patterns: It is evident from the above class diagram that suitable deign patterns were not used to design the structure of the codebase.

2. Missing Abstraction: All the class files follow the same structure causing lots of redundancy in the methods. Using abstraction would have helped in reducing redundancy and also easier in understanding functionality.

3. Insufficient Modularization: Some classes are to large and could have been modularized furhter.

4. Leaky Encapsulation: Some of the encapsulation is unnecessary and bloated. There are several variables which can be combined together as one and used across all the classes.

2.1.1 Proposed design diagram

2.2 Code Smells

In this section we describe the code smells that were detected in the given code.

2.2.1 Code smells within classes

No comments No comments are used throughout the code explaining some finer details that need to be noted. This made some parts of the code incomprehensible.

Long Methods Most of the methods are long which makes the scope of their variables incomprehensible.

Long Parameters List The class MyPanel has several parameters which makes it very complex to understand.

Combinitorial Explosion The code in all the positive and negative files for both latch and the flipflop are mostly duplicated except for the values of the parameters. This need can be avoided by the use of proper inheritance.

Conditional complexity Some of the classes contain complex conditionals which can be avoided altogether by proper structuring.

Uncommunicative names Most of the variable names and method names do not suggest their purpose and are incomprehensible.

Inconsistent names The names of the methods in one class are intermingled with the names of other classes and do not lead to a proper understanding of the flow or their functionality.

Dead code and speculative Generality Some of the code is never used and most of it was modified to solve the problem, rather a hack to get its functionality working.

2.2.2 Code smells between classes

Refused Bequest There are a few classes which inherit other classes but rarely use the inherited functionality.

Divergent Change If any change is made in one class the trace of the change needs to be identified to make the change consistent across all the classes.

Indecent Exposure Most of the parameters and methods are public and are exposed.

2.3 Code metrics

The code metrics for the given code were measured using two tools *CCCC* and *Code Pro*. The results of the code metrics are listed in the following sections.

2.3.1 Code Pro

The results of the code metrics generated by code pro are as shown in the image below:

Results for test [top]

Metric Name	Value
Abstractness	0%
Average Block Depth	0.65
Average Cyclomatic Complexity	6.27
Average Lines Of Code Per Method	35.36
Average Number of Constructors Per Type	0.62
Average Number of Fields Per Type	13.62
Average Number of Methods Per Type	5.01
Average Number of Parameters	1.67
Comments Ratio	21.1%
Efferent Couplings	46
Lines of Code	12,747
Number of Characters	616,618
Number of Comments	2,699
Number of Constructors	36
Number of Fields	790
Number of Lines	19,357
Number of Methods	291
Number of Packages	6
Number of Semicolons	7,933
Number of Types	58
Weighted Methods	2,051

2.3.2 CCCC

The results are generated in HTML format which are easy to view in HTML itself rather than in screen shots of the page hence we are providing a link to the HTML page here in the document. Click here to view the results for D-flip-flop and here for D-latch. The results are present in the "/code-metrics-results/java-code/CCCC/" directory.

3 Refactored code analysis

Because of the large number of code smells and design smells the code was refactored using JavaScript. The code metrics for the JavaScript were obtained in HMTL format. The results are present in the "/code-metrics-results/js-code/code-pro/" directory. The file for the full result is present in the "exp6$_{\text{flipflop positive}}$.html" and "exp$_{\text{latch positive}}$.html" for the flip-flop and

FLIP FLOP Metrics

- **Sloc :**55
- **Maintainability**:45.53
- **Jshint**:0.00
- **Complexity :**
  - **Sloc** : logical:40, physical :55
  - **Cyclomatic** :1,
  - **Halstead** : **operators** : {"distinct":8,"total":85 }, **operands**: {"distinct":28,"total":139 },
  - **Cyclomatic density :** 2.5
- **Length**:224,
- **Vocabulary**:36,
- **Difficulty**:19.857142857142858,
- **Volume**:1158.063200323078,
- **Effort**:22995.826406415406,
- **Bugs**:0.3860210667743593,
- **Time**:1277.5459114675225

latch files respectively.

LATCH METRICS

- **Sloc** :67
- **Maintainability** :43.15704726965145}
- **Complexity** :
  - **Sloc** :{logical:48,physical :67},
  - **Cyclomatic**:1,
  - **Halstead**:{**operators**:{distinct:8,total:102,}, **operands**:{distinct:31,total:168}
- **Length**:270
- **Vocabulary**:39
- **Difficulty**:21.677419354838708
- **Volume**:1427.958599092807
- **Effort:**30934.947696463427
- **Bugs**:0.47568619969760234
- **Time**:1718.6082053590792
- **Cyclomatic density**:2.083333333333333

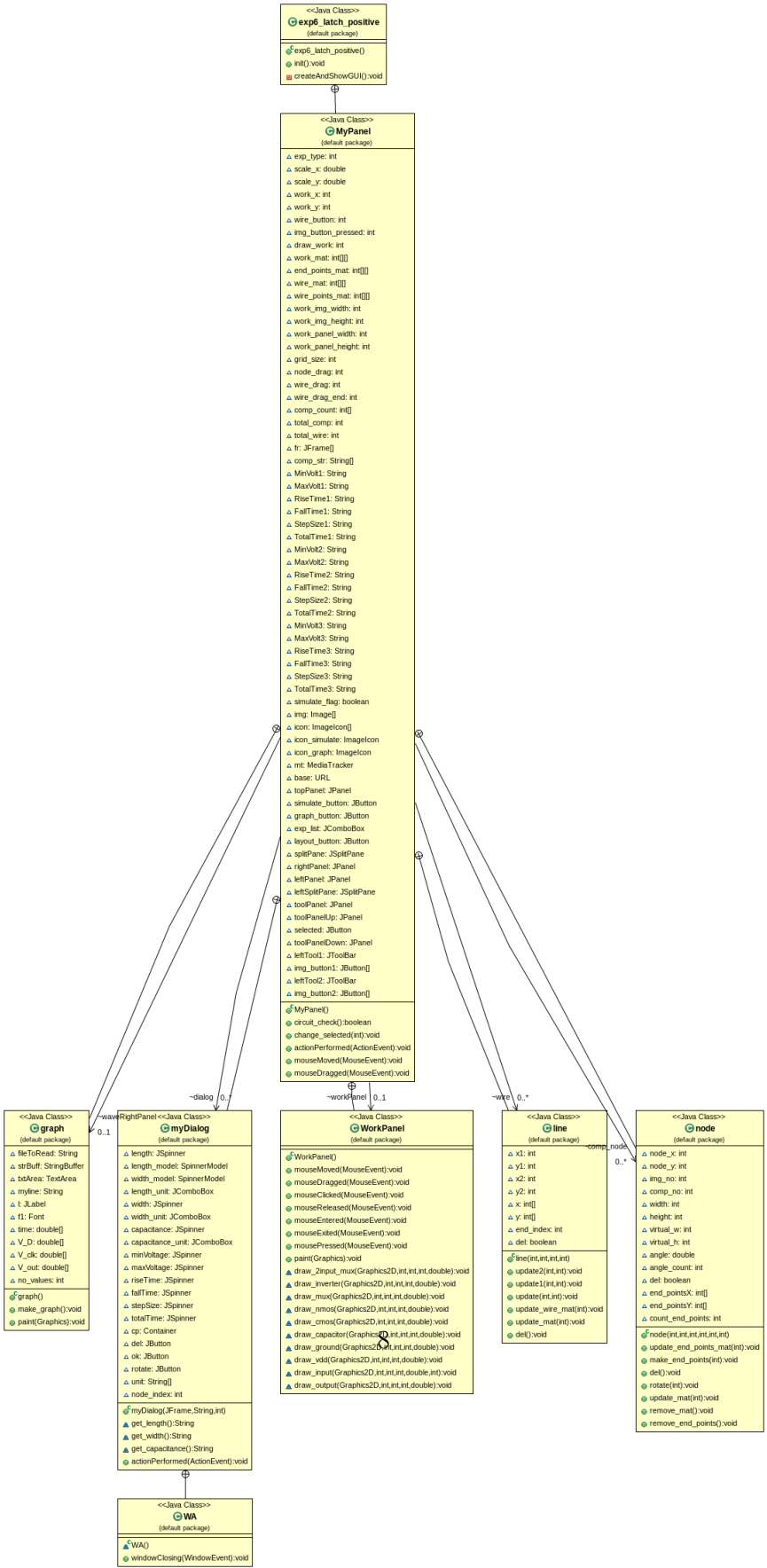
4 Conclusion

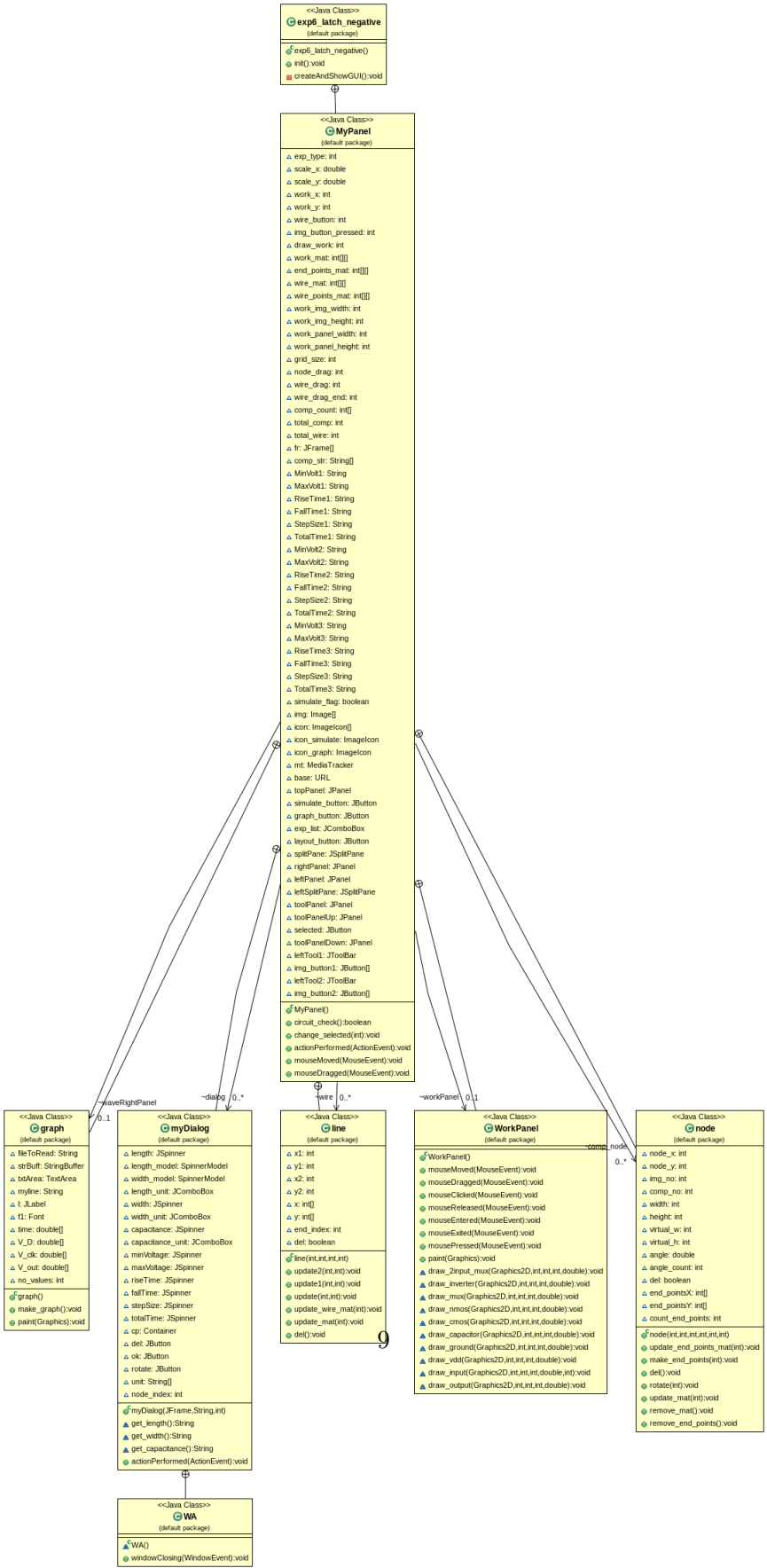
From the metrics above it is evident that the use of design patterns for while designing and keeping in mind a few bets practices while writing the code helps improve the maintainability of the code to a great extent.

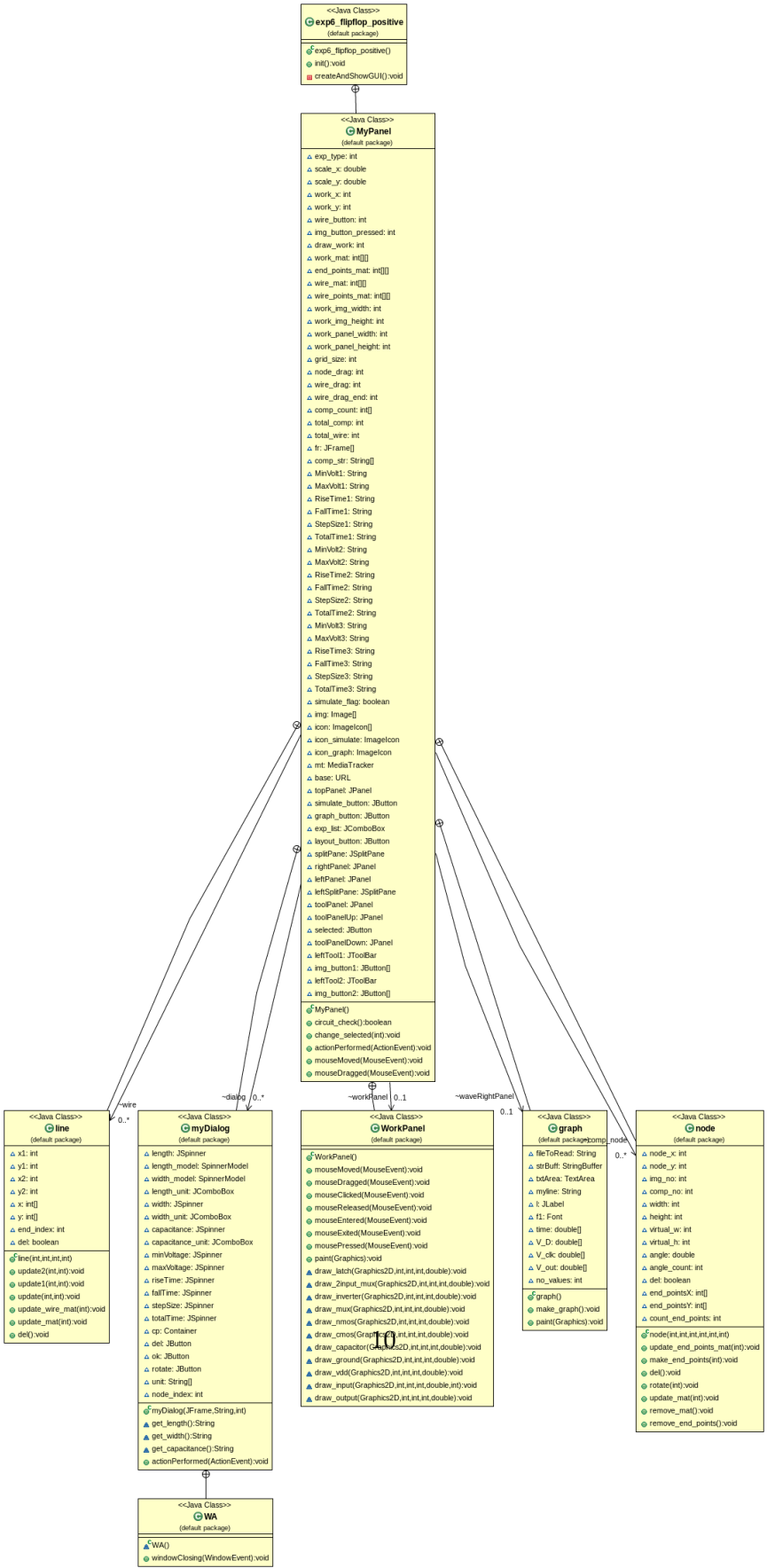
Comparison table:

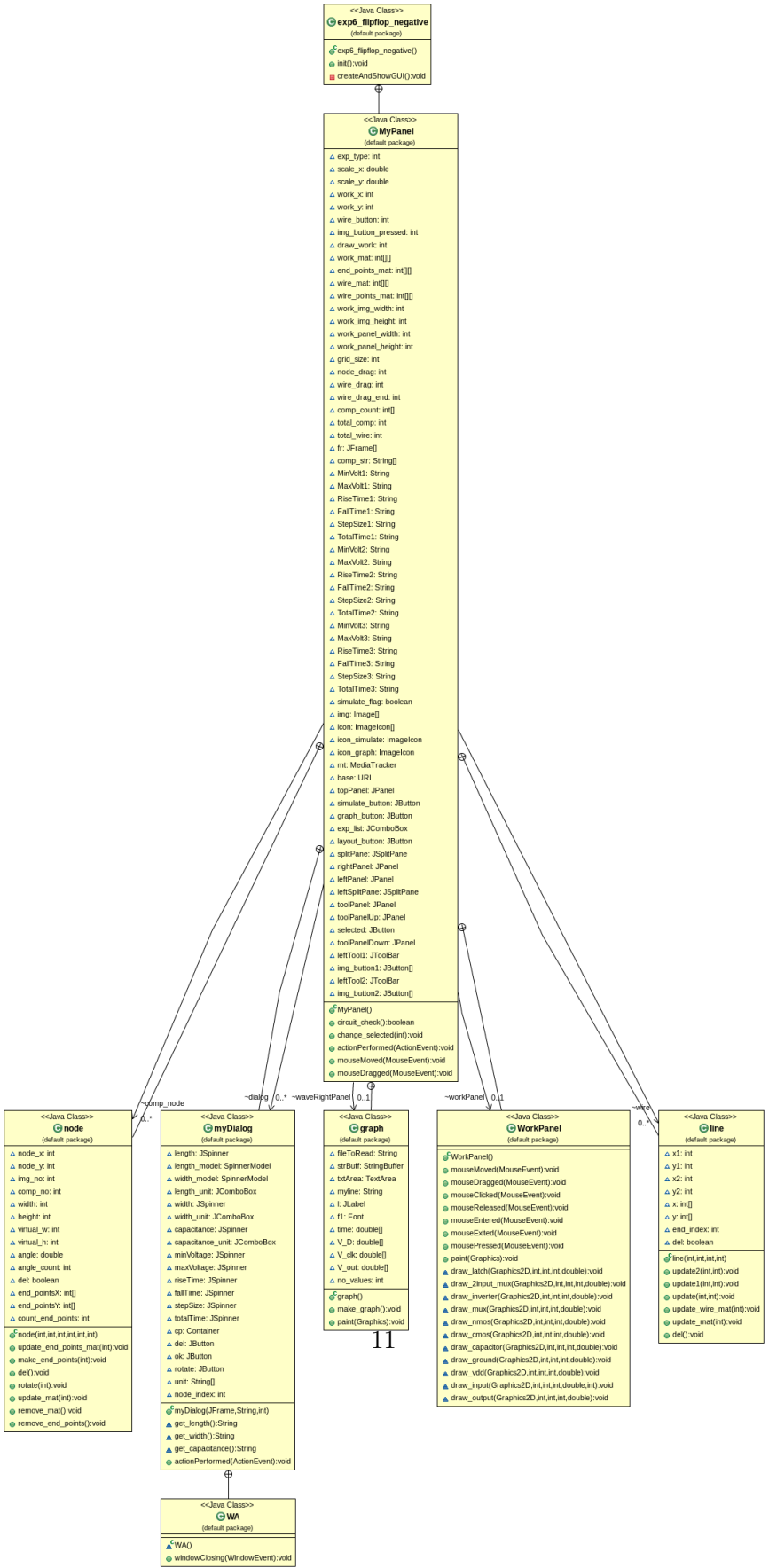
Table 1:

S.no	Metric	Java	JavaScript
1.	Lines of code	12,747	55
2.	Cyclomatic complexity	6.27	1
3.	Maintainability index	12.35	45.5308


<<Java Class>>
exp6_latch_positive
(default package)
exp6_latch_positive()
init():void
createAndShowGUI():void
<<Java Class>>
MyPanel
(default package)
exp_type: int
scale_x: double
scale_y: double
work_x: int
work_y: int
wire_button: int
img_button_pressed: int
draw_work: int
work_mat: int[][]
end_points_mat: int[][]
wire_mat: int[][]
wire_points_mat: int[][]
work_img_width: int
work_img_height: int
work_panel_width: int
work_panel_height: int
grid_size: int
node_drag: int
wire_drag: int
wire_drag_end: int
comp_count: int[]
total_comp: int
total_wire: int
fr: JFrame[]
comp_str: String[]
MinVolt1: String
MaxVolt1: String
RiseTime1: String
FallTime1: String
StepSize1: String
TotalTime1: String
MinVolt2: String
MaxVolt2: String
RiseTime2: String
FallTime2: String
StepSize2: String
TotalTime2: String
MinVolt3: String
MaxVolt3: String
RiseTime3: String
FallTime3: String
StepSize3: String
TotalTime3: String
simulate_flag: boolean
img: Image[]
icon: ImageIcon[]
icon_simulate: ImageIcon
icon_graph: ImageIcon
mt: MediaTracker
base: URL
topPanel: JPanel
simulate_button: JButton
graph_button: JButton
exp_list: JComboBox
layout_button: JButton
splitPane: JSplitPane
rightPanel: JPanel
leftPanel: JPanel
leftSplitPane: JSplitPane
toolPanel: JPanel
toolPanelUp: JPanel
selected: JButton
toolPanelDown: JPanel
leftTool1: JToolBar
img_button1: JButton[]
leftTool2: JToolBar
img_button2: JButton[]
MyPanel()
circuit_check():boolean
change_selected(int):void
actionPerformed(ActionEvent):void
mouseMoved(MouseEvent):void
mouseDragged(MouseEvent):void
~dialog
0..1
~workPanel
0..1
~wire
0..*
~wave
~rightPanel
0..1
<<Java Class>>
graph
(default package)
fileToRead: String
strBuff: StringBuffer
txtArea: TextArea
myline: String
l: JLabel
f1: Font
time: double[]
V_D: double[]
V_clk: double[]
V_out: double[]
no_values: int
graph()
make_graph():void
paint(Graphics):void
<<Java Class>>
myDialog
(default package)
length: JSpinner
length_model: SpinnerModel
width_model: SpinnerModel
length_unit: JComboBox
width: JSpinner
width_unit: JComboBox
capacitance: JSpinner
capacitance_unit: JComboBox
minVoltage: JSpinner
maxVoltage: JSpinner
riseTime: JSpinner
fallTime: JSpinner
stepSize: JSpinner
totalTime: JSpinner
cp: Container
del: JButton
ok: JButton
rotate: JButton
unit: String[]
node_index: int
myDialog(JFrame,String,int)
get_length():String
get_width():String
get_capacitance():String
actionPerformed(ActionEvent):void
<<Java Class>>
WA
(default package)
WA()
windowClosing(WindowEvent):void
<<Java Class>>
WorkPanel
(default package)
WorkPanel()
mouseMoved(MouseEvent):void
mouseDragged(MouseEvent):void
mouseClicked(MouseEvent):void
mouseReleased(MouseEvent):void
mouseEntered(MouseEvent):void
mouseExited(MouseEvent):void
mousePressed(MouseEvent):void
paint(Graphics):void
draw_2input_mux(Graphics2D,int,int,int,double):void
draw_inverter(Graphics2D,int,int,int,double):void
draw_mux(Graphics2D,int,int,int,double):void
draw_nmos(Graphics2D,int,int,int,double):void
draw_cmos(Graphics2D,int,int,int,double):void
draw_capacitor(Graphics2D,int,int,int,double):void
draw_ground(Graphics2D,int,int,int,double):void
draw_vdd(Graphics2D,int,int,int,double):void
draw_input(Graphics2D,int,int,int,double,int):void
draw_output(Graphics2D,int,int,int,double):void
<<Java Class>>
line
(default package)
x1: int
y1: int
x2: int
y2: int
x: int[]
y: int[]
end_index: int
del: boolean
line(int,int,int,int)
update2(int,int):void
update1(int,int):void
update(int,int):void
update_wire_mat(int):void
update_mat(int):void
del():void
~comp_node
0..*
<<Java Class>>
node
(default package)
node_x: int
node_y: int
img_no: int
comp_no: int
width: int
height: int
virtual_w: int
virtual_h: int
angle: double
angle_count: int
del: boolean
end_pointsX: int[]
end_pointsY: int[]
count_end_points: int
node(int,int,int,int,int,int)
update_end_points_mat(int):void
make_end_points(int):void
del():void
rotate(int):void
update_mat(int):void
remove_mat():void
remove_end_points():void


<<Java Class>>
exp6_latch_negative
(default package)
exp6_latch_negative()
init():void
createAndShowGUI():void
<<Java Class>>
MyPanel
(default package)
exp_type: int
scale_x: double
scale_y: double
work_x: int
work_y: int
wire_button: int
img_button_pressed: int
draw_work: int
work_mat: int[][]
end_points_mat: int[][]
wire_mat: int[][]
wire_points_mat: int[][]
work_img_width: int
work_img_height: int
work_panel_width: int
work_panel_height: int
grid_size: int
node_drag: int
wire_drag: int
wire_drag_end: int
comp_count: int[]
total_comp: int
total_wire: int
fr: JFrame[]
comp_str: String[]
MinVolt1: String
MaxVolt1: String
RiseTime1: String
FallTime1: String
StepSize1: String
TotalTime1: String
MinVolt2: String
MaxVolt2: String
RiseTime2: String
FallTime2: String
StepSize2: String
TotalTime2: String
MinVolt3: String
MaxVolt3: String
RiseTime3: String
FallTime3: String
StepSize3: String
TotalTime3: String
simulate_flag: boolean
img: Image[]
icon: ImageIcon[]
icon_simulate: ImageIcon
icon_graph: ImageIcon
mt: MediaTracker
base: URL
topPanel: JPanel
simulate_button: JButton
graph_button: JButton
exp_list: JComboBox
layout_button: JButton
splitPane: JSplitPane
rightPanel: JPanel
leftPanel: JPanel
leftSplitPane: JSplitPane
toolPanel: JPanel
toolPanelUp: JPanel
selected: JButton
toolPanelDown: JPanel
leftTool1: JToolBar
img_button1: JButton[]
leftTool2: JToolBar
img_button2: JButton[]
MyPanel()
circuit_check():boolean
change_selected(int):void
actionPerformed(ActionEvent):void
mouseMoved(MouseEvent):void
mouseDragged(MouseEvent):void
~waveRightPanel
0..1
~dialog
0..*
~wire
0..*
~workPanel
0..1
<<Java Class>>
graph
(default package)
fileToRead: String
strBuff: StringBuffer
txtArea: TextArea
myline: String
l: JLabel
f1: Font
time: double[]
V_D: double[]
V_clk: double[]
V_out: double[]
no_values: int
graph()
make_graph():void
paint(Graphics):void
<<Java Class>>
myDialog
(default package)
length: JSpinner
length_model: SpinnerModel
width_model: SpinnerModel
length_unit: JComboBox
width: JSpinner
width_unit: JComboBox
capacitance: JSpinner
capacitance_unit: JComboBox
minVoltage: JSpinner
maxVoltage: JSpinner
riseTime: JSpinner
fallTime: JSpinner
stepSize: JSpinner
totalTime: JSpinner
cp: Container
del: JButton
ok: JButton
rotate: JButton
unit: String[]
node_index: int
myDialog(JFrame,String,int)
get_length():String
get_width():String
get_capacitance():String
actionPerformed(ActionEvent):void
<<Java Class>>
WA
(default package)
WA()
windowClosing(WindowEvent):void
<<Java Class>>
line
(default package)
x1: int
y1: int
x2: int
y2: int
x: int[]
y: int[]
end_index: int
del: boolean
line(int,int,int,int)
update2(int,int):void
update1(int,int):void
update(int,int):void
update_wire_mat(int):void
update_mat(int):void
del():void
<<Java Class>>
WorkPanel
(default package)
WorkPanel()
mouseMoved(MouseEvent):void
mouseDragged(MouseEvent):void
mouseClicked(MouseEvent):void
mouseReleased(MouseEvent):void
mouseEntered(MouseEvent):void
mouseExited(MouseEvent):void
mousePressed(MouseEvent):void
paint(Graphics):void
draw_2input_mux(Graphics2D,int,int,int,double):void
draw_inverter(Graphics2D,int,int,int,double):void
draw_mux(Graphics2D,int,int,int,double):void
draw_nmos(Graphics2D,int,int,int,double):void
draw_cmos(Graphics2D,int,int,int,double):void
draw_capacitor(Graphics2D,int,int,int,double):void
draw_ground(Graphics2D,int,int,int,double):void
draw_vdd(Graphics2D,int,int,int,double):void
draw_input(Graphics2D,int,int,int,double,int):void
draw_output(Graphics2D,int,int,int,double):void
~comp_node
0..*
<<Java Class>>
node
(default package)
node_x: int
node_y: int
img_no: int
comp_no: int
width: int
height: int
virtual_w: int
virtual_h: int
angle: double
angle_count: int
del: boolean
end_pointsX: int[]
end_pointsY: int[]
count_end_points: int
node(int,int,int,int,int,int)
update_end_points_mat(int):void
make_end_points(int):void
del():void
rotate(int):void
update_mat(int):void
remove_mat():void
remove_end_points():void


<<Java Class>>
exp6_flipflop_positive
(default package)
exp6_flipflop_positive()
init():void
createAndShowGUI():void
<<Java Class>>
MyPanel
(default package)
exp_type: int
scale_x: double
scale_y: double
work_x: int
work_y: int
wire_button: int
img_button_pressed: int
draw_work: int
work_mat: int[][]
end_points_mat: int[][]
wire_mat: int[][]
wire_points_mat: int[][]
work_img_width: int
work_img_height: int
work_panel_width: int
work_panel_height: int
grid_size: int
node_drag: int
wire_drag: int
wire_drag_end: int
comp_count: int[]
total_comp: int
total_wire: int
fr: JFrame[]
comp_str: String[]
MinVolt1: String
MaxVolt1: String
RiseTime1: String
FallTime1: String
StepSize1: String
TotalTime1: String
MinVolt2: String
MaxVolt2: String
RiseTime2: String
FallTime2: String
StepSize2: String
TotalTime2: String
MinVolt3: String
MaxVolt3: String
RiseTime3: String
FallTime3: String
StepSize3: String
TotalTime3: String
simulate_flag: boolean
img: Image[]
icon: ImageIcon[]
icon_simulate: ImageIcon
icon_graph: ImageIcon
mt: MediaTracker
base: URL
topPanel: JPanel
simulate_button: JButton
graph_button: JButton
exp_list: JComboBox
layout_button: JButton
splitPane: JSplitPane
rightPanel: JPanel
leftPanel: JPanel
leftSplitPane: JSplitPane
toolPanel: JPanel
toolPanelUp: JPanel
selected: JButton
toolPanelDown: JPanel
leftTool1: JToolBar
img_button1: JButton[]
leftTool2: JToolBar
img_button2: JButton[]
MyPanel()
circuit_check():boolean
change_selected(int):void
actionPerformed(ActionEvent):void
mouseMoved(MouseEvent):void
mouseDragged(MouseEvent):void
~wire
0..*
~dialog
0..*
~workPanel
0..1
~waveRightPanel
0..1
~comp_node
0..*
<<Java Class>>
line
(default package)
x1: int
y1: int
x2: int
y2: int
x: int[]
y: int[]
end_index: int
del: boolean
line(int,int,int,int)
update2(int,int):void
update1(int,int):void
update(int,int):void
update_wire_mat(int):void
update_mat(int):void
del():void
<<Java Class>>
myDialog
(default package)
length: JSpinner
length_model: SpinnerModel
width_model: SpinnerModel
length_unit: JComboBox
width: JSpinner
width_unit: JComboBox
capacitance: JSpinner
capacitance_unit: JComboBox
minVoltage: JSpinner
maxVoltage: JSpinner
riseTime: JSpinner
fallTime: JSpinner
stepSize: JSpinner
totalTime: JSpinner
cp: Container
del: JButton
ok: JButton
rotate: JButton
unit: String[]
node_index: int
myDialog(JFrame,String,int)
get_length():String
get_width():String
get_capacitance():String
actionPerformed(ActionEvent):void
<<Java Class>>
WA
(default package)
WA()
windowClosing(WindowEvent):void
<<Java Class>>
WorkPanel
(default package)
WorkPanel()
mouseMoved(MouseEvent):void
mouseDragged(MouseEvent):void
mouseClicked(MouseEvent):void
mouseReleased(MouseEvent):void
mouseEntered(MouseEvent):void
mouseExited(MouseEvent):void
mousePressed(MouseEvent):void
paint(Graphics):void
draw_latch(Graphics2D,int,int,int,double):void
draw_2input_mux(Graphics2D,int,int,int,double):void
draw_inverter(Graphics2D,int,int,int,double):void
draw_mux(Graphics2D,int,int,int,double):void
draw_nmos(Graphics2D,int,int,int,double):void
draw_cmos(Graphics2D,int,int,int,double):void
draw_capacitor(Graphics2D,int,int,int,double):void
draw_ground(Graphics2D,int,int,int,double):void
draw_vdd(Graphics2D,int,int,int,double):void
draw_input(Graphics2D,int,int,int,double,int):void
draw_output(Graphics2D,int,int,int,double):void
<<Java Class>>
graph
(default package)
fileToRead: String
strBuff: StringBuffer
txtArea: TextArea
myline: String
l: JLabel
f1: Font
time: double[]
V_D: double[]
V_clk: double[]
V_out: double[]
no_values: int
graph()
make_graph():void
paint(Graphics):void
<<Java Class>>
node
(default package)
node_x: int
node_y: int
img_no: int
comp_no: int
width: int
height: int
virtual_w: int
virtual_h: int
angle: double
angle_count: int
del: boolean
end_pointsX: int[]
end_pointsY: int[]
count_end_points: int
node(int,int,int,int,int,int)
update_end_points_mat(int):void
make_end_points(int):void
del():void
rotate(int):void
update_mat(int):void
remove_mat():void
remove_end_points():void


<<Java Class>>
exp6_flipflop_negative
(default package)
exp6_flipflop_negative()
init():void
createAndShowGUI():void
<<Java Class>>
MyPanel
(default package)
exp_type: int
scale_x: double
scale_y: double
work_x: int
work_y: int
wire_button: int
img_button_pressed: int
draw_work: int
work_mat: int[][]
end_points_mat: int[][]
wire_mat: int[][]
wire_points_mat: int[][]
work_img_width: int
work_img_height: int
work_panel_width: int
work_panel_height: int
grid_size: int
node_drag: int
wire_drag: int
wire_drag_end: int
comp_count: int[]
total_comp: int
total_wire: int
fr: JFrame[]
comp_str: String[]
MinVolt1: String
MaxVolt1: String
RiseTime1: String
FallTime1: String
StepSize1: String
TotalTime1: String
MinVolt2: String
MaxVolt2: String
RiseTime2: String
FallTime2: String
StepSize2: String
TotalTime2: String
MinVolt3: String
MaxVolt3: String
RiseTime3: String
FallTime3: String
StepSize3: String
TotalTime3: String
simulate_flag: boolean
img: Image[]
icon: ImageIcon[]
icon_simulate: ImageIcon
icon_graph: ImageIcon
mt: MediaTracker
base: URL
topPanel: JPanel
simulate_button: JButton
graph_button: JButton
exp_list: JComboBox
layout_button: JButton
splitPane: JSplitPane
rightPanel: JPanel
leftPanel: JPanel
leftSplitPane: JSplitPane
toolPanel: JPanel
toolPanelUp: JPanel
selected: JButton
toolPanelDown: JPanel
leftTool1: JToolBar
img_button1: JButton[]
leftTool2: JToolBar
img_button2: JButton[]
MyPanel()
circuit_check():boolean
change_selected(int):void
actionPerformed(ActionEvent):void
mouseMoved(MouseEvent):void
mouseDragged(MouseEvent):void
~comp_node
0..*
~dialog
0..*
~waveRightPanel
0..1
~workPanel
0..1
~wire
0..*
<<Java Class>>
node
(default package)
node_x: int
node_y: int
img_no: int
comp_no: int
width: int
height: int
virtual_w: int
virtual_h: int
angle: double
angle_count: int
del: boolean
end_pointsX: int[]
end_pointsY: int[]
count_end_points: int
node(int,int,int,int,int,int)
update_end_points_mat(int):void
make_end_points(int):void
del():void
rotate(int):void
update_mat(int):void
remove_mat():void
remove_end_points():void
<<Java Class>>
myDialog
(default package)
length: JSpinner
length_model: SpinnerModel
width_model: SpinnerModel
length_unit: JComboBox
width: JSpinner
width_unit: JComboBox
capacitance: JSpinner
capacitance_unit: JComboBox
minVoltage: JSpinner
maxVoltage: JSpinner
riseTime: JSpinner
fallTime: JSpinner
stepSize: JSpinner
totalTime: JSpinner
cp: Container
del: JButton
ok: JButton
rotate: JButton
unit: String[]
node_index: int
myDialog(JFrame,String,int)
get_length():String
get_width():String
get_capacitance():String
actionPerformed(ActionEvent):void
<<Java Class>>
WA
(default package)
WA()
windowClosing(WindowEvent):void
<<Java Class>>
graph
(default package)
fileToRead: String
strBuff: StringBuffer
txtArea: TextArea
myline: String
l: JLabel
f1: Font
time: double[]
V_D: double[]
V_clk: double[]
V_out: double[]
no_values: int
graph()
make_graph():void
paint(Graphics):void
<<Java Class>>
WorkPanel
(default package)
WorkPanel()
mouseMoved(MouseEvent):void
mouseDragged(MouseEvent):void
mouseClicked(MouseEvent):void
mouseReleased(MouseEvent):void
mouseEntered(MouseEvent):void
mouseExited(MouseEvent):void
mousePressed(MouseEvent):void
paint(Graphics):void
draw_latch(Graphics2D,int,int,int,double):void
draw_2input_mux(Graphics2D,int,int,int,double):void
draw_inverter(Graphics2D,int,int,int,double):void
draw_mux(Graphics2D,int,int,int,double):void
draw_nmos(Graphics2D,int,int,int,double):void
draw_cmos(Graphics2D,int,int,int,double):void
draw_capacitor(Graphics2D,int,int,int,double):void
draw_ground(Graphics2D,int,int,int,double):void
draw_vdd(Graphics2D,int,int,int,double):void
draw_input(Graphics2D,int,int,int,double,int):void
draw_output(Graphics2D,int,int,int,double):void
<<Java Class>>
line
(default package)
x1: int
y1: int
x2: int
y2: int
x: int[]
y: int[]
end_index: int
del: boolean
line(int,int,int,int)
update2(int,int):void
update1(int,int):void
update(int,int):void
update_wire_mat(int):void
update_mat(int):void
del():void