Group members info

Fullname	**email**
Burkman, Brendan	bburkman@depaul.edu
Chaidez, Arturo	achaide3@depaul.edu
Peraza, Eliecer	eperazaa@depaul.edu

Requirements Modeling

Intro

CalcPlus is an application built from scratch, which includes features such as arithmetic operations, unit conversions, and more over time. During its development, techniques to assess and ensure software quality and proper testing have been consequently applied. CalcPlus will allow the user to perform the basic arithmetic operations, as well as unit conversion in an intuitive, effective, efficient and easy manner. CalcPlus analysis, design and implementation, and enhancements will be developed using design patterns, quality metrics, software tests, and any other tool/approach deemed necessary to achieve a product of great quality and error-free.

Functional/Non Functional requirements

Functional Requirements (FR)

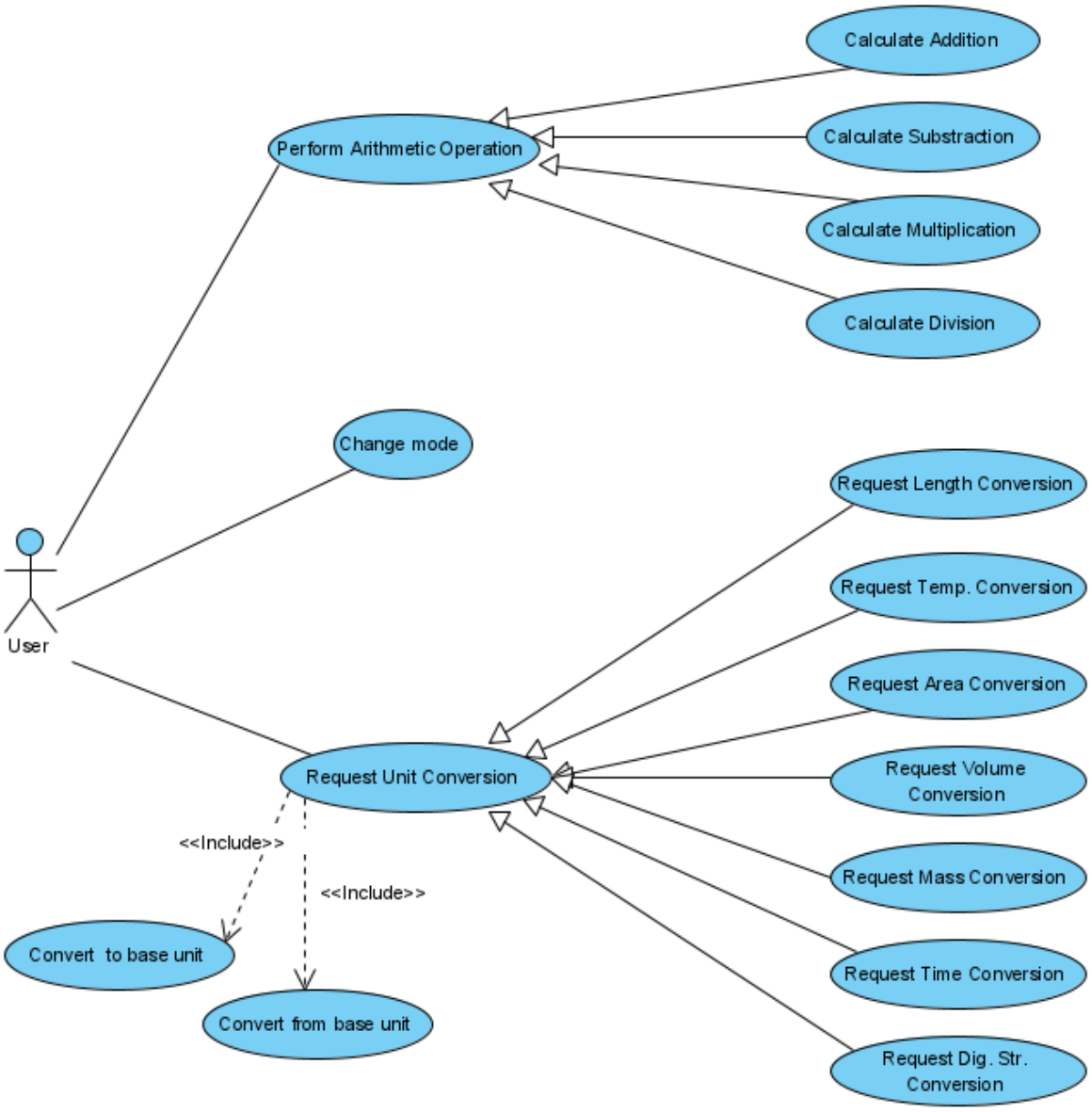
- **FR001** The user should be able to perform the addition operation. The system should show the results of the addition.
- **FR002** The user should be able to perform the subtraction operation. The system should show the results of the subtraction operation.
- **FR003** The user should be able to perform the multiplication operation. The system should show the results of the multiplication operation.
- **FR004** The user should be able to perform the division operation. The system should show the results of the division operation.
- **FR005** The user should be able to perform the percentage operation. The system should show the results of the percentage operation.
- **FR006** The user should be able to request a unit conversion based on length. The system should show the results of such conversion.
- **FR007** The user should be able to request a unit conversion based on temperature. The system should show the results of such conversion.
- **FR008** The user should be able to request a unit conversion based on area. The system should show the result of such conversion.
- **FR009** The user should be able to request a unit conversion based on volume. The system should show the results of such conversion.
- **FR010** The user should be able to request a unit conversion based on mass. The system should show the results of such conversion.

- **FR011** The user should be able to request a unit conversion based on time. The system should show the results of such conversion.
- **FR012** The user should be able to request a unit conversion based on digital storage. The system should show the results of such conversion.
- **FR013** The user should be able to enter any real number, either integers or the decimal form representation of the number.
- **FR014** The user should be able to select between the basic calculator or the unit converter at the beginning of the use.
- **FR015** The user should be able to change the mode of operation between the basic calculator or the unit converter at any time.
- **FR016** The user should be able to get appropriate ERROR messages when the entered data warrants it. Ex. division by zero.
- **FR017** The user should be able to exit the application at any time.
- **FR018** The user should be able to perform the change signs operation. The system should show the results of the change signs operation.
- **FR019** The calculator's initial value should be 0.
- **FR020** The user should be able to carry multiple operations. Every next operation should trigger the execution of the previous operation and provide partial results.
- **FR021**. The calculator should show a precision of at least 2 decimals.

Nonfunctional Requirements (NFR)

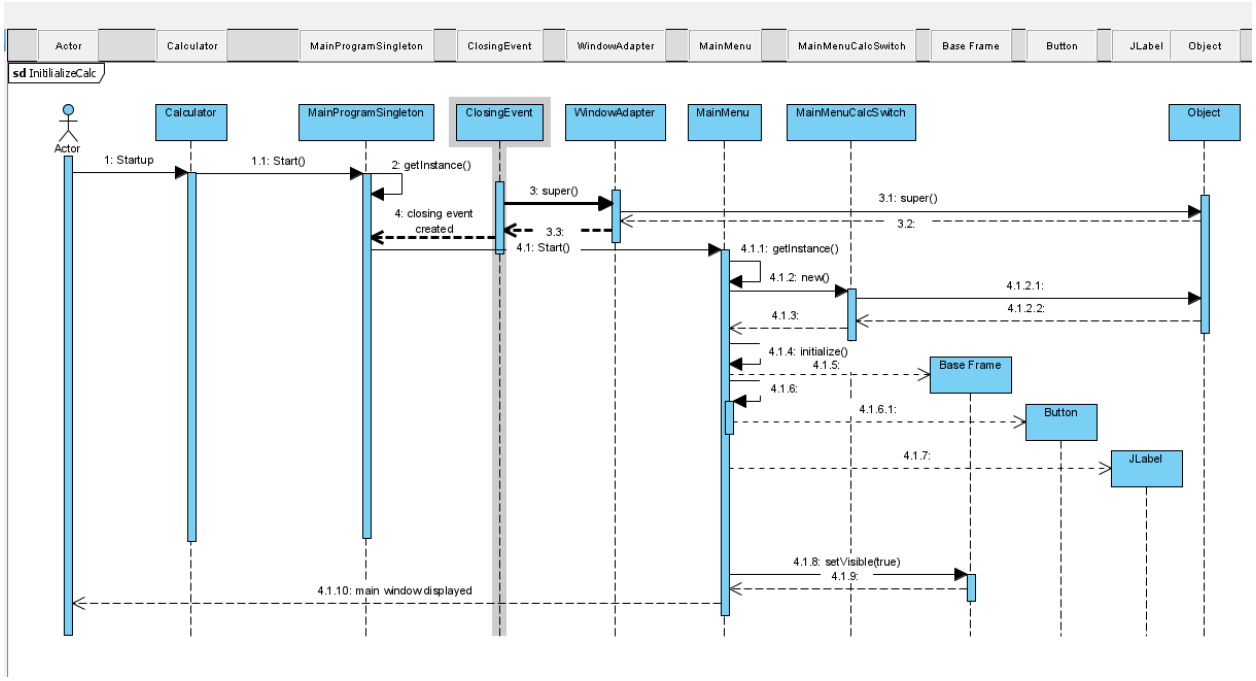
- **NFR001** The applications must be developed in JAVA and should be able to be run on a JDK 11 or newer release.
- **NFR002** The application should be easy to use and intuitive. Most users should be able to perform most operations with basic or no training
- **NFR003** The response time for any operation should be almost imperceptible to the user. A response greater than one second should not be allowed.
- **NFR004** The application must include a graphic interface.
- **NFR005** The application should be portable, testable, maintainable, extensible, and scalable.
- **NFR006** No license must be used/required for the development of the application.
- **NFR007** When the app gets out of focus, then it should be able to keep the same state it was before losing focus.
- **NFR008** The app should open up in less than 3 seconds. The app should switch between calculator and converter in less than one second.
- **NFR009** The app should be able to add additional units for the converter.
- **NFR010** When input value for unit conversion, it should be easy to add the from and to units as well as the category. The reload of the units based on category should be almost instantaneous.
- **NFR011** The app should not make a wrong calculation for either the calculator or the converter.
- **NFR012** User should be aware that the application can be run on a desktop environment or any other device with jvm configured.
- **NFR013** Invalid operations should be reported as they occurred.
- **NFR014** App must not exceed 5 MB of memory.
- **NFR015** App should be able to work with large numbers based on Java Double precision.
- **NFR016** The layout for the calculator should stick to the standard layout for most calculators, with proper size and shape, so the user can be used to that familiarity.
- **NFR017** No security requirement has been defined for this application.

Use case diagram

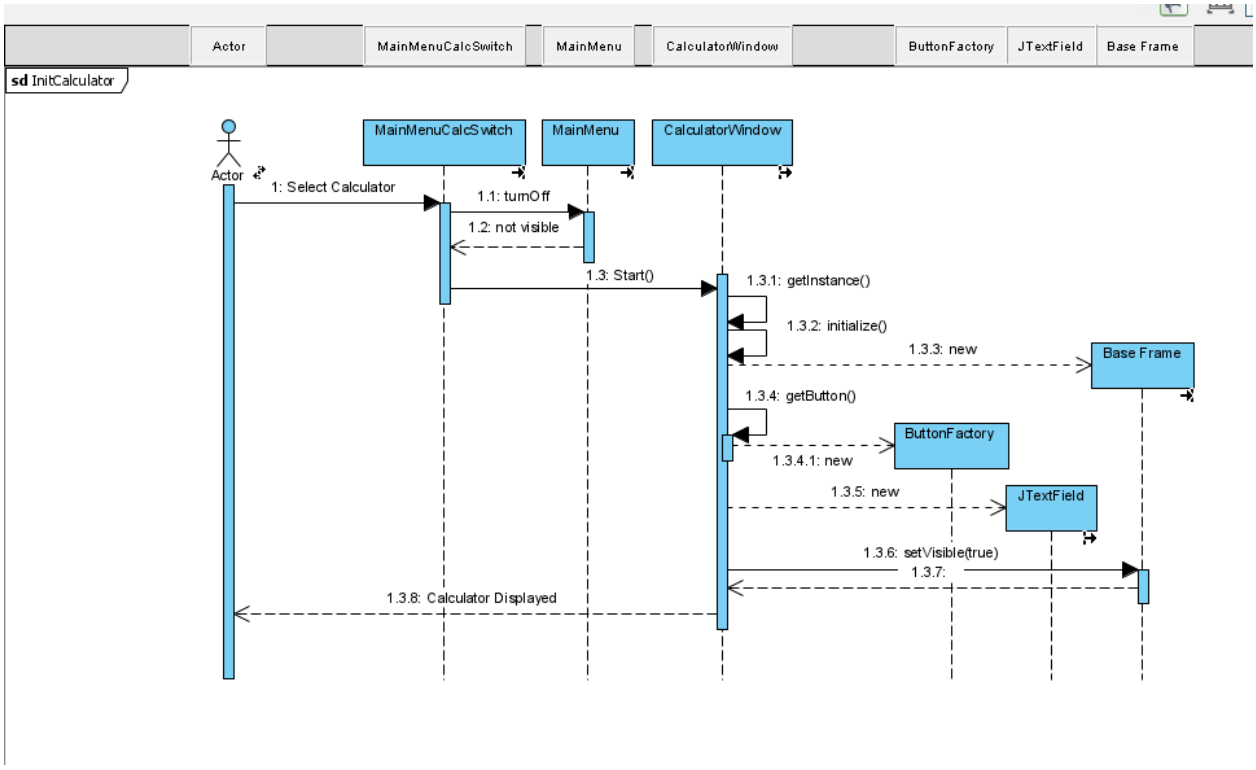


Sequence diagrams

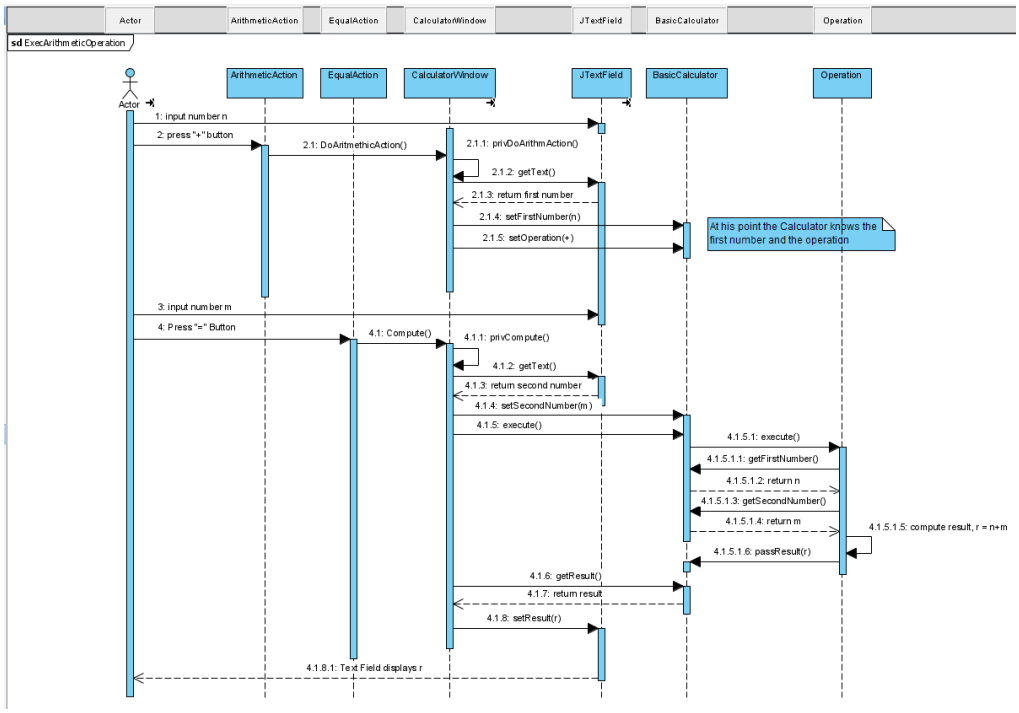
Initializing AppMenu



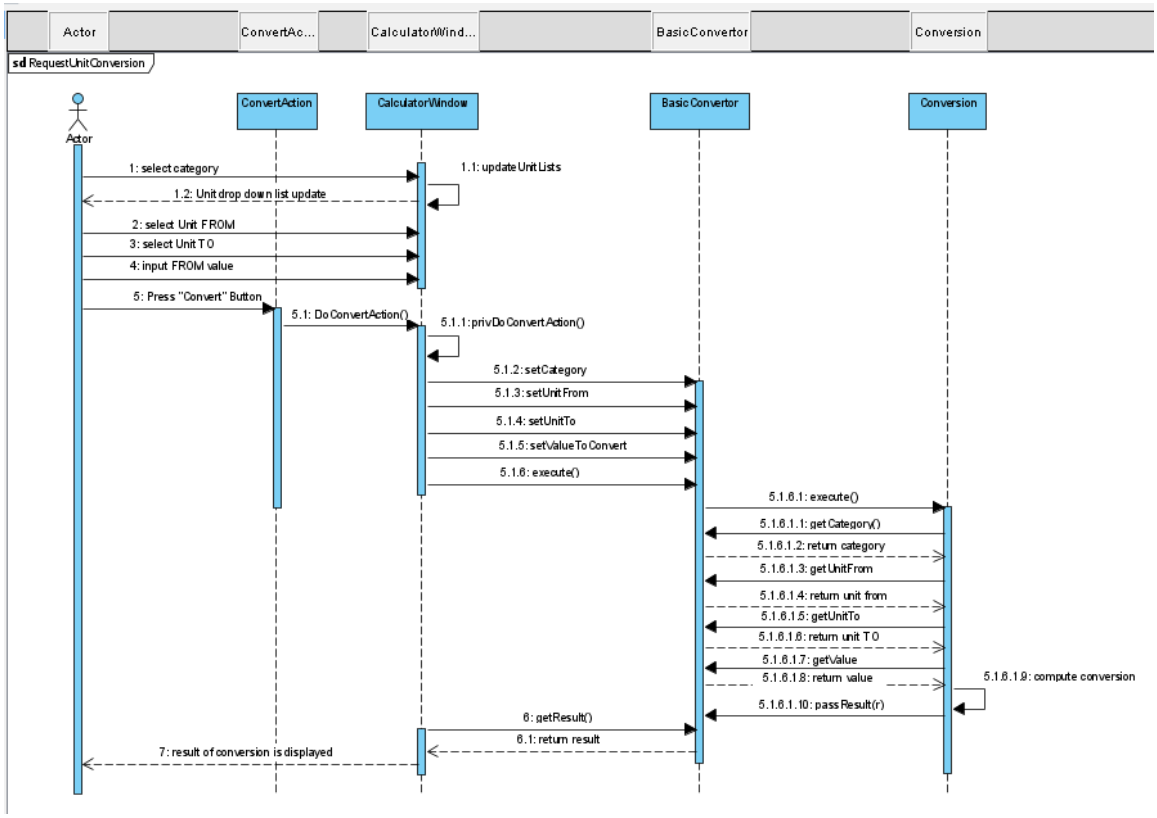
Initializing Calc



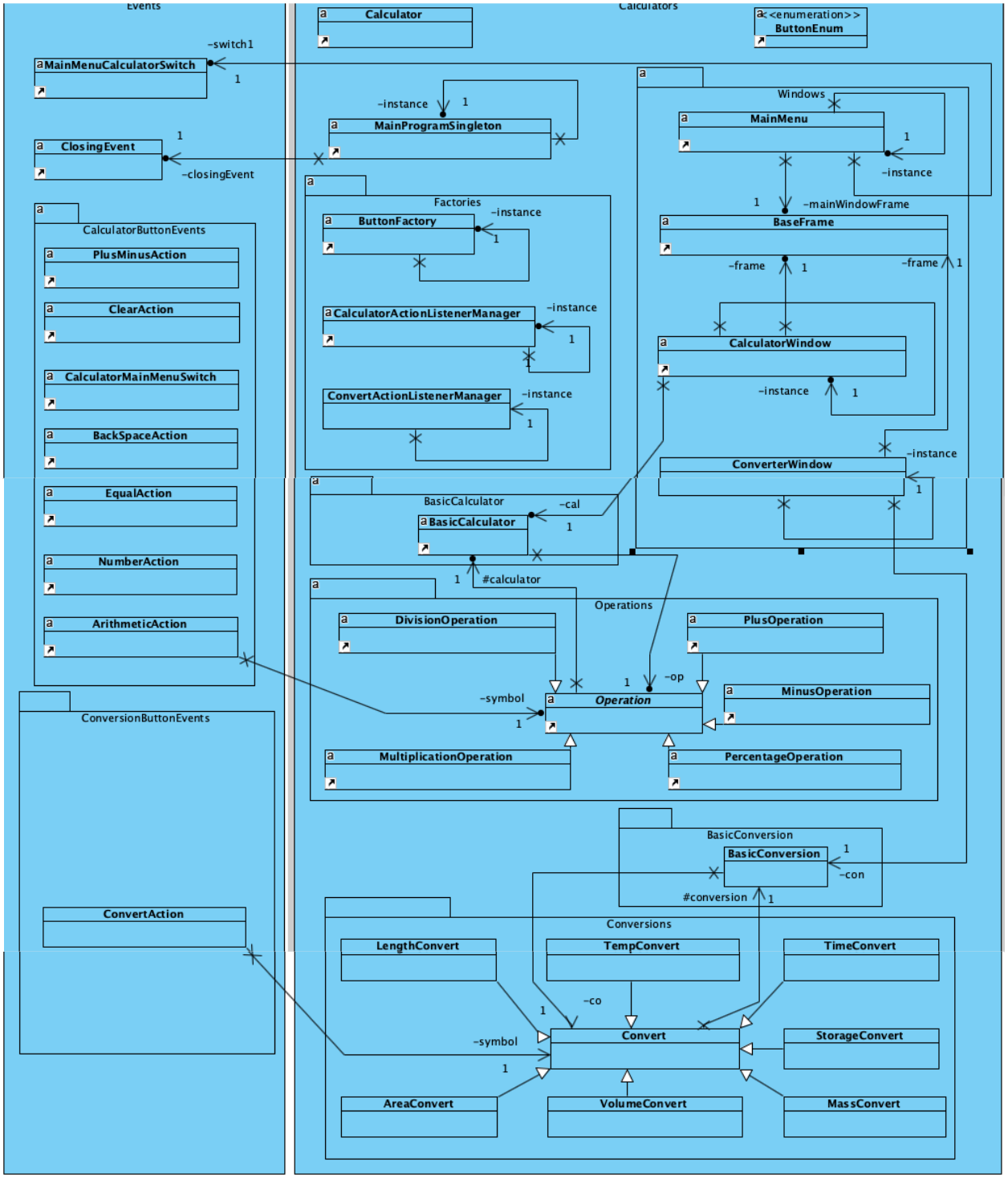
Performing an arithmetic operation

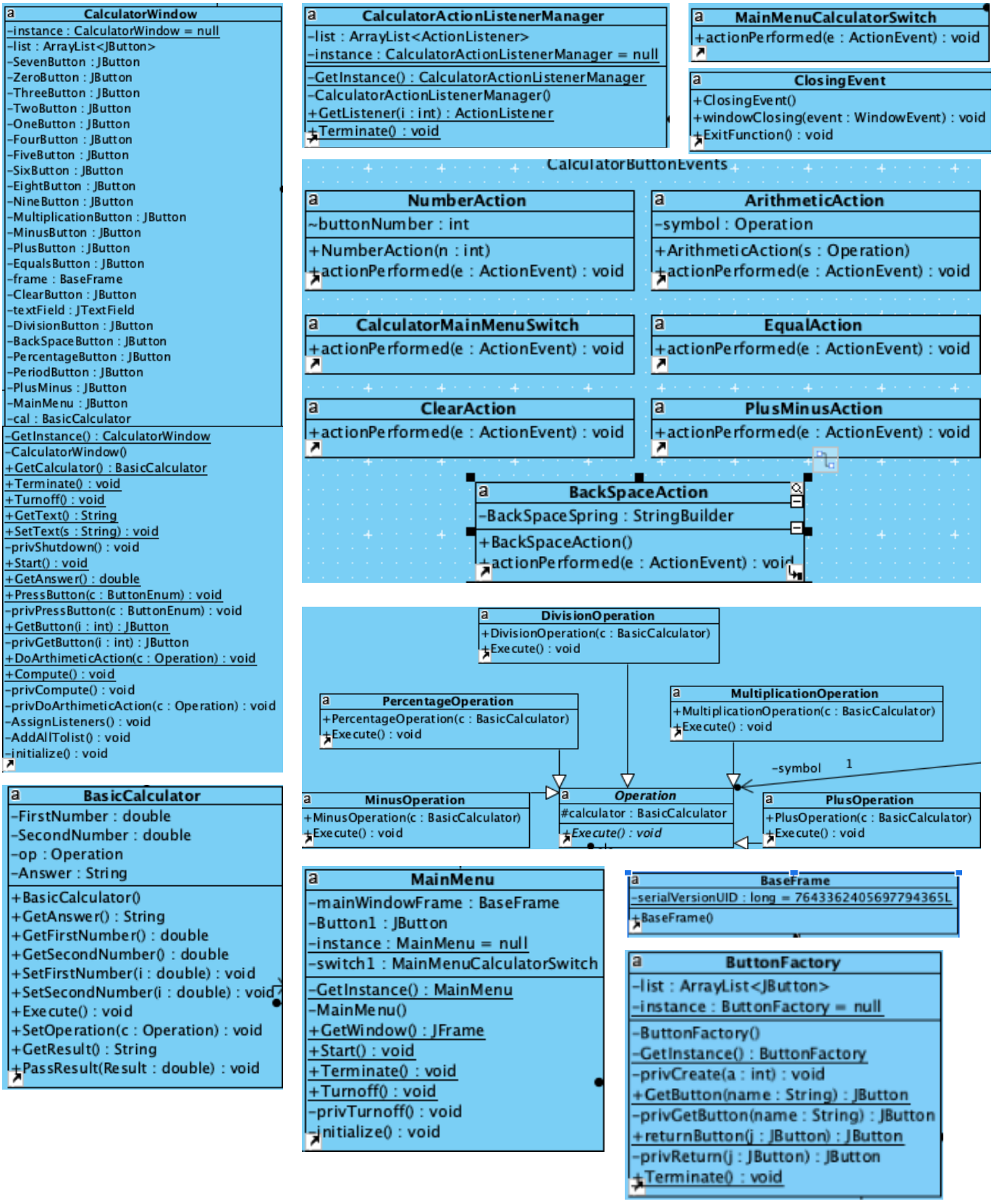


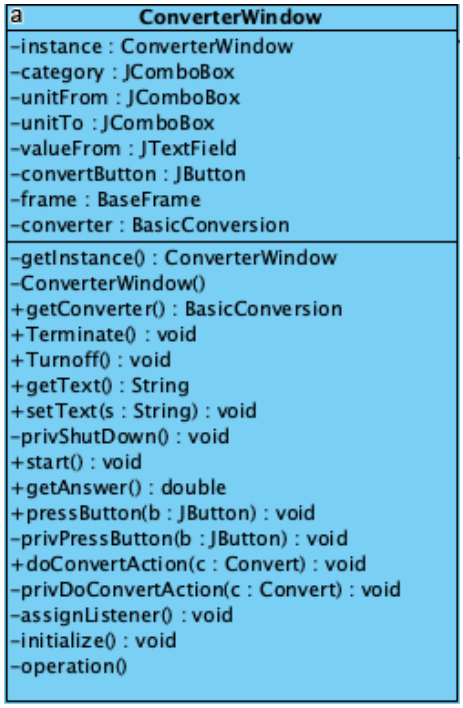
Requesting Unit Conversion

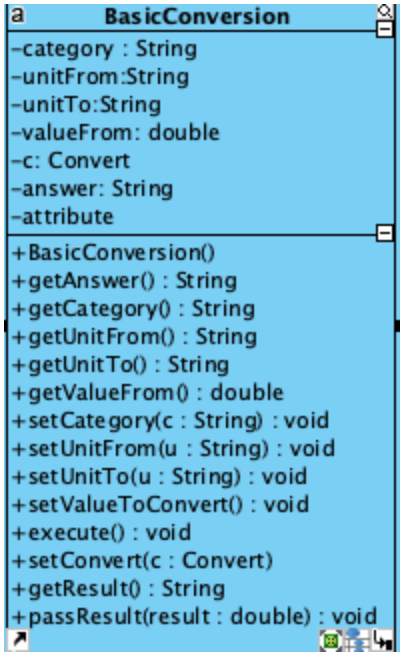


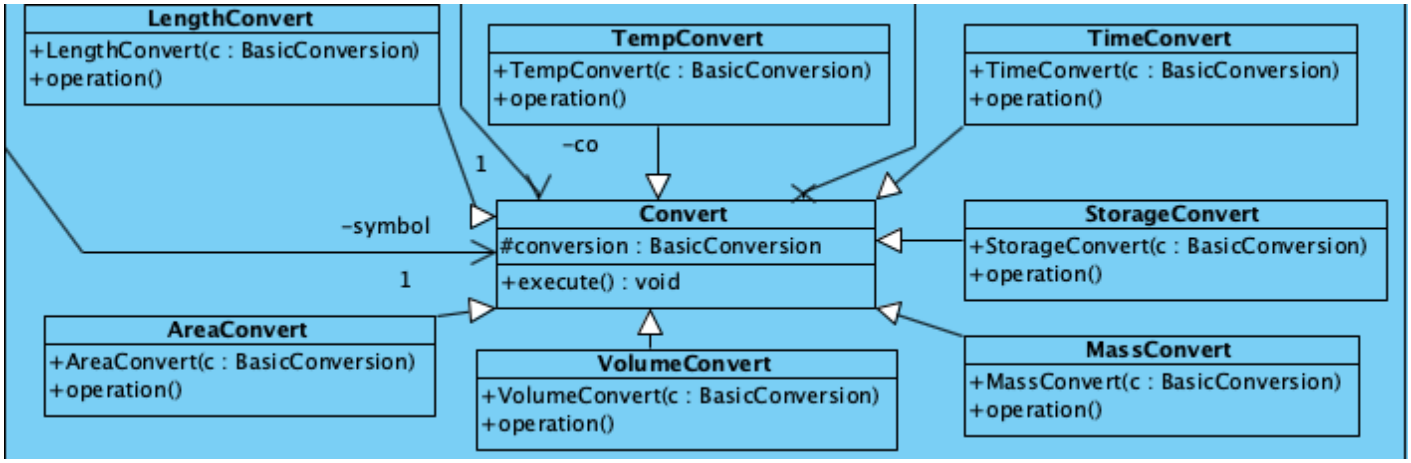
Class diagrams

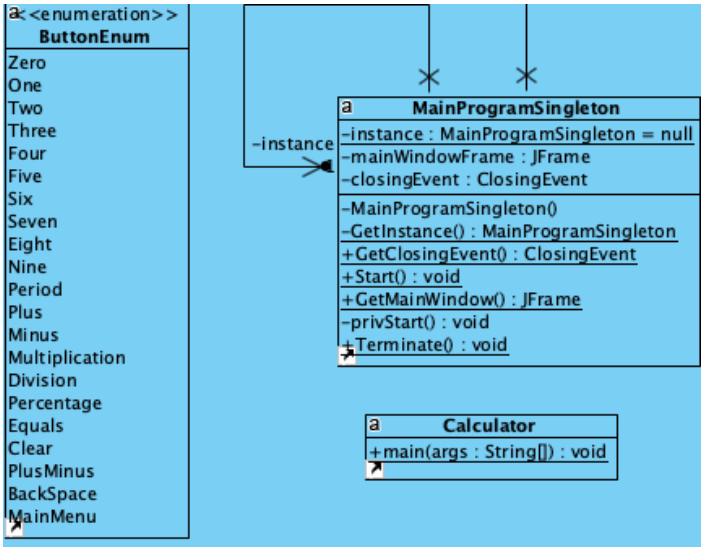



CalculatorWindow
-instance : CalculatorWindow = null
-list : ArrayList<JButton>
-SevenButton : JButton
-ZeroButton : JButton
-ThreeButton : JButton
-TwoButton : JButton
-OneButton : JButton
-FourButton : JButton
-FiveButton : JButton
-SixButton : JButton
-EightButton : JButton
-NineButton : JButton
-MultiplicationButton : JButton
-MinusButton : JButton
-PlusButton : JButton
-EqualsButton : JButton
-frame : BaseFrame
-ClearButton : JButton
-textField : JTextField
-DivisionButton : JButton
-BackSpaceButton : JButton
-PercentageButton : JButton
-PeriodButton : JButton
-PlusMinus : JButton
-MainMenu : JButton
-cal : BasicCalculator
-GetInstance() : CalculatorWindow
-CalculatorWindow()
+GetCalculator() : BasicCalculator
+Terminate() : void
+Turnoff() : void
+GetText() : String
+SetText(s : String) : void
-privShutdown() : void
+Start() : void
+GetAnswer() : double
+PressButton(c : ButtonEnum) : void
-privPressButton(c : ButtonEnum) : void
+GetButton(i : int) : JButton
-privGetButton(i : int) : JButton
+DoArthimeticAction(c : Operation) : void
+Compute() : void
-privCompute() : void
-privDoArthimeticAction(c : Operation) : void
-AssignListeners() : void
-AddAllTolist() : void
-initialize() : void
CalculatorActionListenerManager
-list : ArrayList<ActionListener>
-instance : CalculatorActionListenerManager = null
-GetInstance() : CalculatorActionListenerManager
-CalculatorActionListenerManager()
+GetListener(i : int) : ActionListener
+Terminate() : void
MainMenuCalculatorSwitch
+actionPerformed(e : ActionEvent) : void
ClosingEvent
+ClosingEvent()
+windowClosing(event : WindowEvent) : void
+ExitFunction() : void
CalculatorButtonEvents
NumberAction
~buttonNumber : int
+NumberAction(n : int)
+actionPerformed(e : ActionEvent) : void
ArithmeticAction
-symbol : Operation
+ArithmeticAction(s : Operation)
+actionPerformed(e : ActionEvent) : void
CalculatorMainMenuSwitch
+actionPerformed(e : ActionEvent) : void
EqualAction
+actionPerformed(e : ActionEvent) : void
ClearAction
+actionPerformed(e : ActionEvent) : void
PlusMinusAction
+actionPerformed(e : ActionEvent) : void
BackSpaceAction
-BackSpaceSpring : StringBuilder
+BackSpaceAction()
+actionPerformed(e : ActionEvent) : void
DivisionOperation
+DivisionOperation(c : BasicCalculator)
+Execute() : void
PercentageOperation
+PercentageOperation(c : BasicCalculator)
+Execute() : void
MultiplicationOperation
+MultiplicationOperation(c : BasicCalculator)
+Execute() : void
-symbol
1
MinusOperation
+MinusOperation(c : BasicCalculator)
+Execute() : void
Operation
#calculator : BasicCalculator
+Execute() : void
PlusOperation
+PlusOperation(c : BasicCalculator)
+Execute() : void
BasicCalculator
-FirstNumber : double
-SecondNumber : double
-op : Operation
-Answer : String
+BasicCalculator()
+GetAnswer() : String
+GetFirstNumber() : double
+GetSecondNumber() : double
+SetFirstNumber(i : double) : void
+SetSecondNumber(i : double) : void
+Execute() : void
+SetOperation(c : Operation) : void
+GetResult() : String
+PassResult(Result : double) : void
MainMenu
-mainWindowFrame : BaseFrame
-Button1 : JButton
-instance : MainMenu = null
-switch1 : MainMenuCalculatorSwitch
-GetInstance() : MainMenu
-MainMenu()
+GetWindow() : JFrame
+Start() : void
+Terminate() : void
+Turnoff() : void
-privTurnoff() : void
-initialize() : void
BaseFrame
-serialVersionUID : long = 7643362405697794365L
+BaseFrame()
ButtonFactory
-list : ArrayList<JButton>
-instance : ButtonFactory = null
-ButtonFactory()
-GetInstance() : ButtonFactory
-privCreate(a : int) : void
+GetButton(name : String) : JButton
-privGetButton(name : String) : JButton
+returnButton(j : JButton) : JButton
-privReturn(j : JButton) : JButton
+Terminate() : void

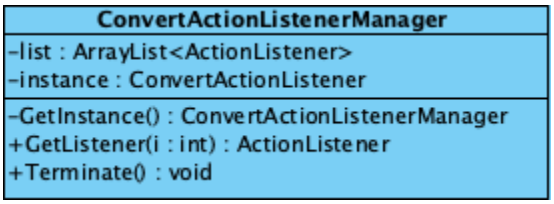

ConverterWindow
-instance : ConverterWindow
-category : JComboBox
-unitFrom : JComboBox
-unitTo : JComboBox
-valueFrom : JTextField
-convertButton : JButton
-frame : BaseFrame
-converter : BasicConversion
-getInstance() : ConverterWindow
-ConverterWindow()
+getConverter() : BasicConversion
+Terminate() : void
+Turnoff() : void
+getText() : String
+setText(s : String) : void
-privShutDown() : void
+start() : void
+getAnswer() : double
+pressButton(b : JButton) : void
-privPressButton(b : JButton) : void
+doConvertAction(c : Convert) : void
-privDoConvertAction(c : Convert) : void
-assignListener() : void
-initialize() : void
-operation()


BasicConversion
-category : String
-unitFrom:String
-unitTo:String
-valueFrom: double
-c: Convert
-answer: String
-attribute
+BasicConversion()
+getAnswer() : String
+getCategory() : String
+getUnitFrom() : String
+getUnitTo() : String
+getValueFrom() : double
+setCategory(c : String) : void
+setUnitFrom(u : String) : void
+setUnitTo(u : String) : void
+setValueToConvert() : void
+execute() : void
+setConvert(c : Convert)
+getResult() : String
+passResult(result : double) : void


LengthConvert
+LengthConvert(c : BasicConversion)
+operation()
TempConvert
+TempConvert(c : BasicConversion)
+operation()
TimeConvert
+TimeConvert(c : BasicConversion)
+operation()
-co
1
-symbol
1
Convert
#conversion : BasicConversion
+execute() : void
StorageConvert
+StorageConvert(c : BasicConversion)
+operation()
AreaConvert
+AreaConvert(c : BasicConversion)
+operation()
VolumeConvert
+VolumeConvert(c : BasicConversion)
+operation()
MassConvert
+MassConvert(c : BasicConversion)
+operation()


<<enumeration>>
ButtonEnum
Zero
One
Two
Three
Four
Five
Six
Seven
Eight
Nine
Period
Plus
Minus
Multiplication
Division
Percentage
Equals
Clear
PlusMinus
BackSpace
MainMenu
-instance
MainProgramSingleton
-instance : MainProgramSingleton = null
-mainWindowFrame : JFrame
-closingEvent : ClosingEvent
-MainProgramSingleton()
-GetInstance() : MainProgramSingleton
+GetClosingEvent() : ClosingEvent
+Start() : void
+GetMainWindow() : JFrame
-privStart() : void
-Terminate() : void
Calculator
+main(args : String[]) : void


ConvertAction
+actionPerformed(e : ActionEvent) : void


ConvertActionListenerManager
-list : ArrayList<ActionListener>
-instance : ConvertActionListener
-GetInstance() : ConvertActionListenerManager
+GetListener(i : int) : ActionListener
+Terminate() : void

Tasks status

Type	Status	Title	Start date	End date	Time (d)	%
Milestone	Done	Project Kickoff	01/22/2021	01/22/2021		100%
Task	Done	Create Project Proposal	01/24/2021	02/02/2021	7	100%
Milestone	Done	Project - Phase I - Proposal	02/03/2021	02/03/2021		100%
Task	Done	Project Scope and Requirements Gathering	02/03/2021	02/10/2021	6	100%
Task	Done	Design	02/11/2021	02/16/2021	4	100%
Milestone	Done	Project - Phase II	02/21/2021	02/21/2021		100%
Task	In Progress	Prototype Dev A	02/21/2021	02/24/2021	3	95%
Task	In Progress	Development A - Basic calculator	02/22/2021	03/02/2021	7	90%
Task	In Progress	Testing & Revision A	03/02/2021	03/05/2021	4	25%
Task	In Progress	Quality Metric Analysis A	03/02/2021	03/05/2021	4	25%
Milestone	In Progress	Development Phase A	03/05/2021	03/05/2021		75%
Task	To be started	Prototype Dev B	03/02/2021	03/05/2021	3	0%
Task	To be started	Development B - Unit Converter	03/02/2021	03/10/2021	7	0%
Task	To be started	Testing & Revision B	03/10/2021	03/14/2021	3	0%
Task	To be started	Quality Metric Analysis B	03/10/2021	03/14/2021	3	0%
Milestone	To be started	Development Phase B	03/14/2021	03/14/2021		0%
Task	To be started	Refactoring - if needed	03/14/2021	03/16/2021	2	0%
Milestone	In Progress	Project - Phase III - Final Release	03/16/2021	03/16/2021		35%

QA Analysis Summary

DesignateJava did return potential code smells; however, the presence of them in the code is justifiable, so they shouldn't be catalogued as such. In some cases, they do acknowledge a class may serve a specific known purpose. They state to ignore the smell if this is the case.

For implementation code smells:

- There were repeated code smells of magic numbers. In Calculators, class MainMenu, method initialize, DesignateJava does not understand all these numbers are being used to create defined boundaries for the application. These boundary numbers are used only once and are unlikely to change. If they are to be changed, they only need to be changed once.
- Under Calculators.Windows, class CalculatorWindow, method ActionListener, the imputed numbers are specific numbers tied to certain buttons. Once again, this number will only need to be changed here.

For design code smells:

- For class basicCalculator, the smell is unutilized abstraction but states this smell may be ignored. This class is only getters and setters, which is why it was misunderstood. All the design smells from here are from similar situations.

For architecture smells:

- The code smell here is feature concentration in Events.CalculatorButtonEvents. It says there is Lack of Component Cohesion (LCC) = 1.0. These are arithmetic operations, which were separated into their own classes. The other way to do this, which is an actual code smell, is to use if/else statements or switch statements.

When looking at the metrics using Understand and DesigniteJava, they indicate quality code from both. In the three examples below, weighted methods per class (WMC) and depth of inheritance tree (DIT) are both low and either match or nearly match. Number of children (NOC) is acceptable as well. Understand's lack of cohesion methods (LCOM) is off, but we see in DesigniteJava the LCOM are great.

		WMC	DIT	NOC	LCOM
ButtonFactory	Understand	12	1	0	56
	DesigniteJava	12	0	0	0
CalculatorWindow	Understand	23	1	0	81
	DesigniteJava	23	0	0	.381
PlusOperation	Understand	2	2	0	0
	DesigniteJava	2	1	0	-1