User Manual

By Charlie Woodman

1. Table of Contents

Table of Contents

2. Introduction

FlowJava is an application that allows users to create Java applications by building flowcharts. It is designed to help users learn how to program and create basic programs in Java. The application uses a specialised flowchart language that can be used to properly represent simple Java programs. Once you have built a flowchart you can run the program it represents or download it as a Java file.

Note: FlowJava should not be used to build professional Java applications. It is intended to be used as an educational tool to help introduce the user to the Java programming language. FlowJava does not support creating classes which are a powerful part of the Java language. Once a user is confident in their use of FlowJava they are encouraged to research more complex IDE's. That being said, FlowJava is Turing complete and therefore has the same computing capabilities as any other Turing complete programming software.

3. Prerequisites

3.1 Environment

FlowJava can be run on any desktop with Java compatibility.

To run the application you must use a JDK and not a JRE and you must run it from a CLI application that can use Java e.g. on Command Prompt on a Windows machine you could run the FlowJava.jar file using the command "java -jar ~location of FlowJava.jar~/ FlowJava.jar". If you are using windows make sure you have the java environment variables set up on your system.
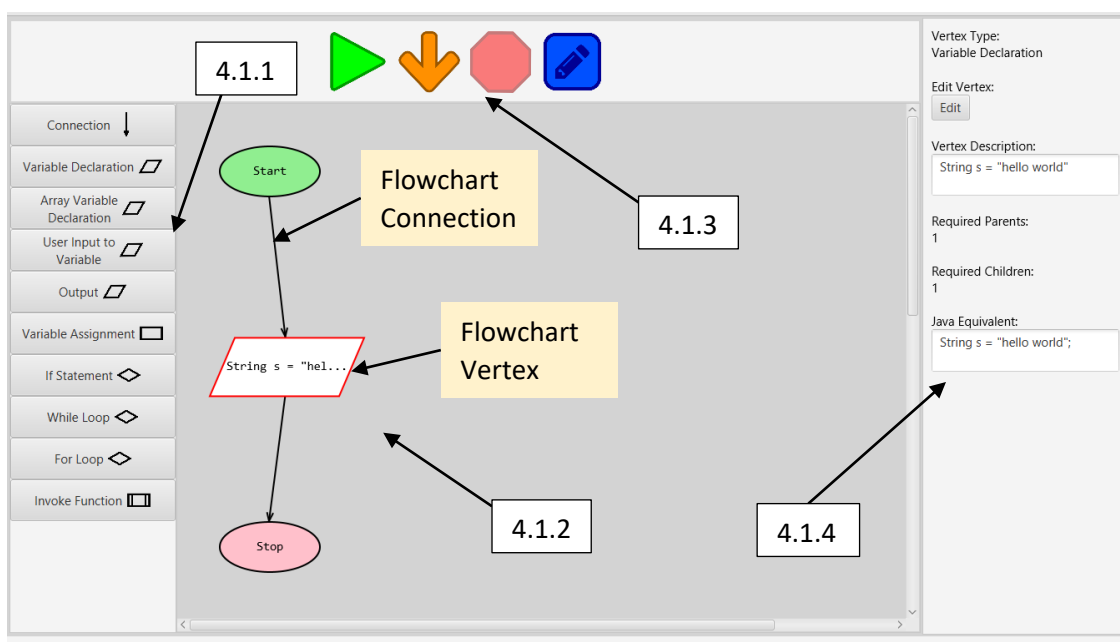
3.2 Prior Knowledge

Users should have knowledge of flowcharts and some basic programming concepts to use the application. If you are completely inexperienced with these topics you can use FlowJava to aid in your research.

4. User Interface

4.1 Main scene

The main scene of FlowJava :

4.1.1 Left Sidebar

The left sidebar contains the buttons used to add new nodes to your flowchart.

4.1.1.1 Connection Button

To add a connection to your flowchart click the connection button, then click the parent vertex of the connection, then finally click the child vertex of the connection (If you are unfamiliar with graph terminology go to https://towardsdatascience.com/graph-theory-132122ac38f2).

4.1.2 Flowchart Pane

The flowchart pane is where you can manipulate nodes to create flowcharts. You can click on elements on the flowchart pane to select them. If a selected element is removable, the delete key can be used to remove them. The start and stop vertices are not removable.

4.1.3 Toolbar

The toolbar has 4 buttons: the run button, the convert button, the terminate button and the functions manager button.

4.1.3.1 Run Button

The run button can be used to run the flowchart program currently on the flowchart pane. Flowchart programs must be correctly structured and syntactically correct to run.

Note: the actual code that is executed when pressing the run button is not the same as the code that is outputted from a conversion. The code that is run is modified to integrate it into the FlowJava application. The most notable difference is how input and outputs are handled. With programs run in FlowJava, input windows are used. However, in converted programs "System.in" and "System.out" are used for simplicity.

4.1.3.2 Convert Button

The convert button can be used to convert the program currently on the flowchart pane into a Java file. Flowchart programs must be correctly structured and syntactically correct to convert.

Note: FlowJava considers programs with unreachable statements (e.g. while(false){}) to be syntactically correct. However, many Java compilers consider these to be erroneous and may not compile if your programs contain them.

4.1.3.3 Terminate Button

The terminate button allows you to terminate a program that is currently running early. This can be used to stop a run of a program with an infinite loop. The button is only enabled when programs are running.

Note: Infinitely looping programs should be avoided in practice as there may not always be a way to terminate them like in FlowJava. The terminate button is implemented in FlowJava as the user is expected to make mistakes as part of the learning process.

4.1.3.4 Functions Manager Button

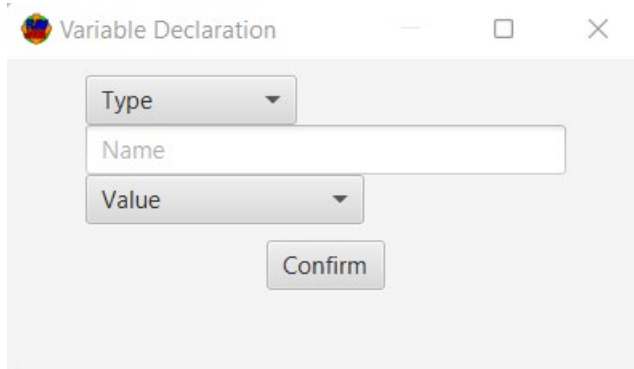The functions manager button is used to open the functions manager window.

4.1.4 Right Sidebar

The right sidebar is used to display information on the currently selected flowchart element. If the flowchart element is editable then the edit button will appear here.

4.2 New Vertex Window

The new vertex window will display when you click a button on the right sidebar to add a new vertex to the flowchart.
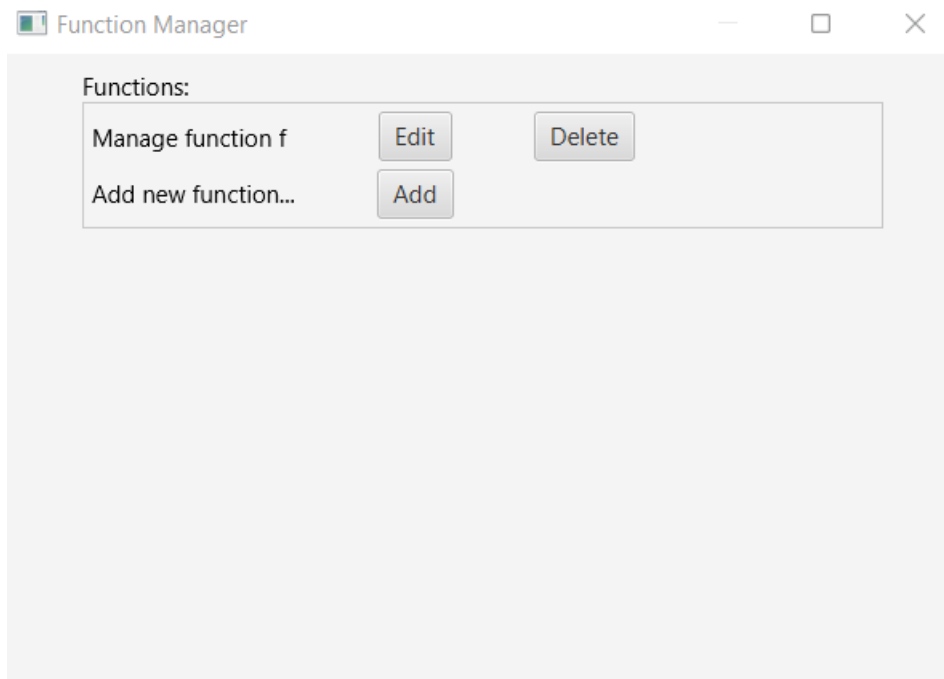
The new vertex window for a variable declaration node:

To add the new vertex to the flowchart first the user must fill in the input form with valid information. The input form has fields for the user to input the information needed for the new vertex.

Note: information inputted to the form which is considered valid may still have errors relating to the semantics of the rest of the program or the syntax of Java. If this is the case the user will be shown an error window when they try to run or convert the program.

4.3 Functions Manager Window

The functions manager window can be used to add, edit and delete functions for your program. The window:


Clicking the add button will open the function editor window. Once a function is created through the function editor window it will appear in the functions manager window. Functions can only be saved

and added to the program if they are considered valid. Any function shown in the functions manager window can be called with a function invocation vertex[see 5.9.1].

4.4 Function Editor Window

The function editor window:
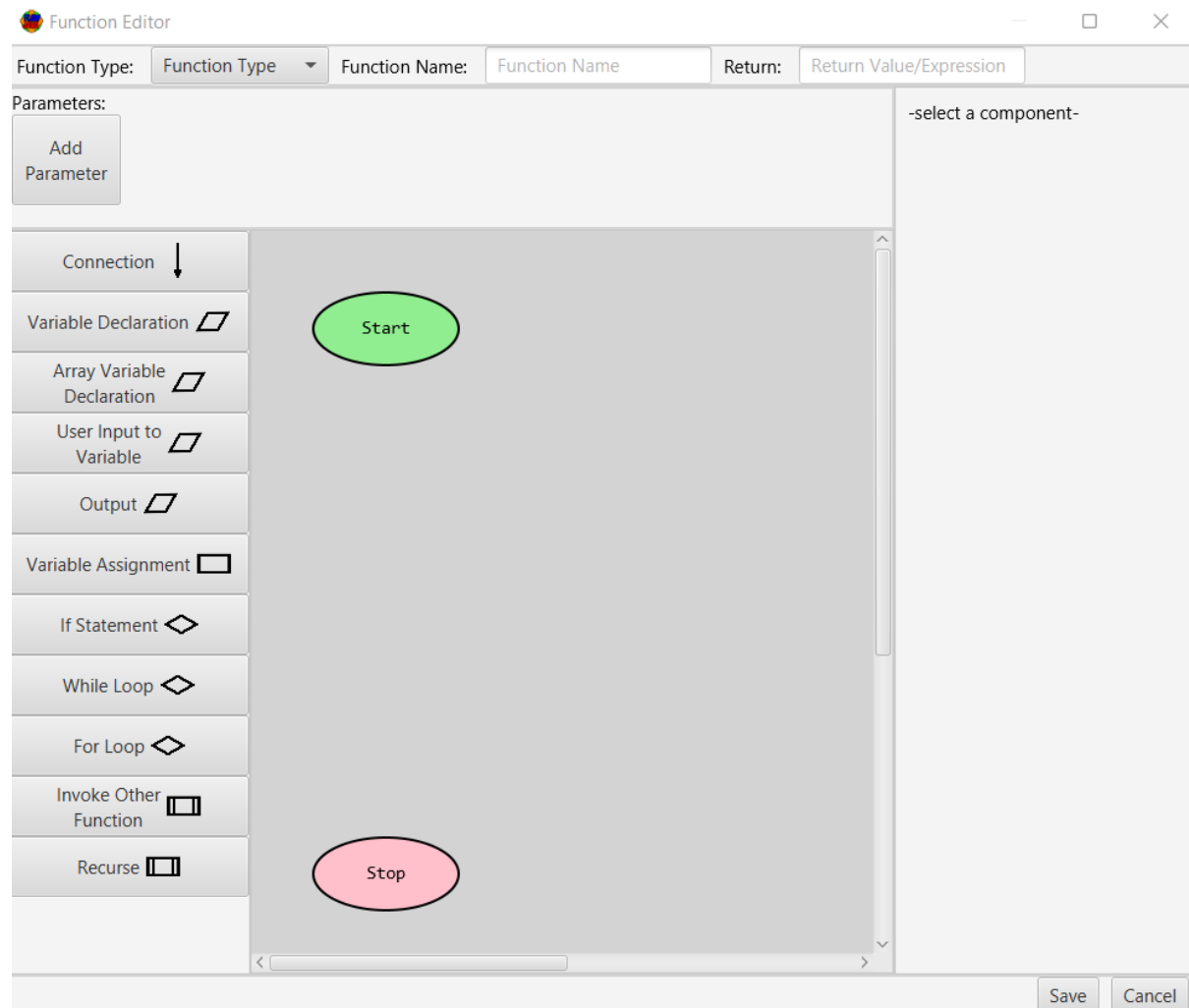
It is very similar to the main scene but specialised for creating functions for your programs. The buttons of the tool bar have been replaced with the add parameter button which opens up the parameter window.

There is an extra bar at the top for entering details of the function. The type of the function is the data type that is returned by the function. If the function doesn't return anything set the type to "void". The name of the function must be a valid function name for Java. The function name can't be "start", "run", "main", "getUserInput", "showAlert" or "cancelRun" as these function names are used in the program that runs user created programs in FlowJava.

There is an extra bar at the bottom with buttons for saving or cancelling your progress.

For a function to be saved and added to the program it must be considered valid.

Note: information inputted to the window which is considered valid may still have errors relating to the semantics of the rest of the program or the syntax of Java. If this is the case the user will be shown an error window when they try to run or convert the program from the main scene.

4.4.1 Parameter Window

The parameter window can be used to add or edit a parameter for a function. The window:
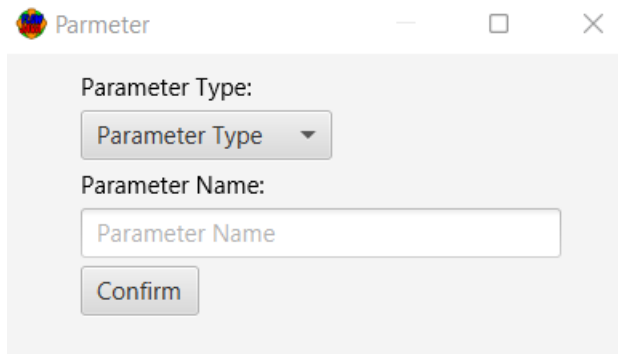

For a parameter to be added to the function it must be considered valid.

Note: information inputted to the window which is considered valid may still have errors relating to the semantics of the rest of the program or the syntax of Java. If this is the case the user will be shown an error window when they try to run or convert the program from the main scene.

4.5 User Created Expression Error Report Window

The user created expression error report window, or error report window for short, is used to display syntactic or semantic errors in user created programs as a result of user error. The window:
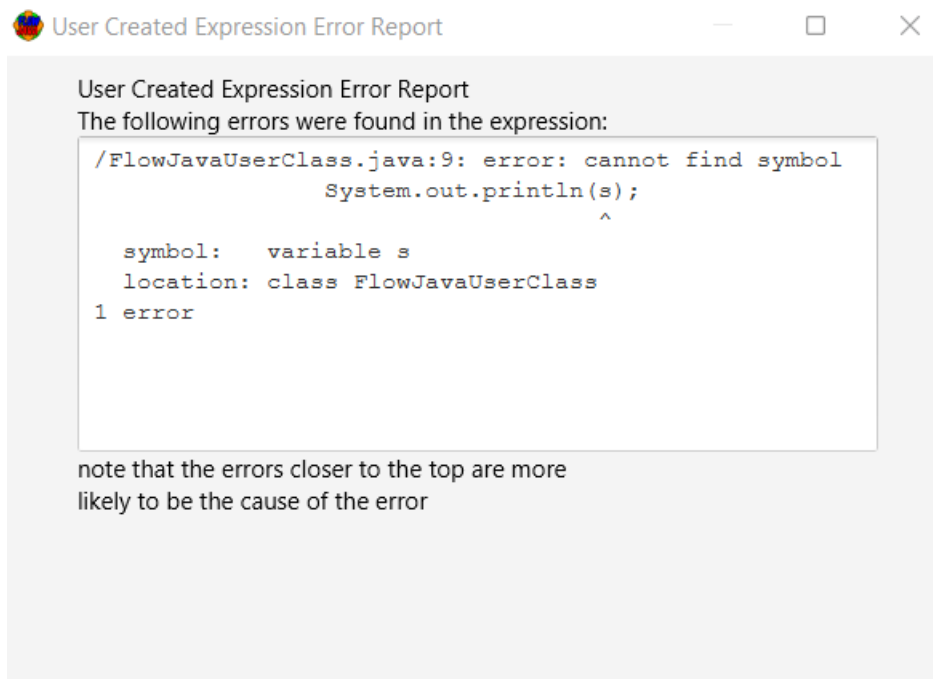
Inside the box is the error outputted directly from the Java compiler that FlowJava uses. The window will be displayed when the user tries to run or convert their program but there are syntactical or semantical errors in their program (including the functions) created by the user.

Note: you can use the Java equivalent box on the right sidebar to help locate where the error is in your flowchart program[see 4.1.4].

4.6 Create Expression Window

The create expression window can be used when creating some vertices to help create expressions for the input fields. The window:
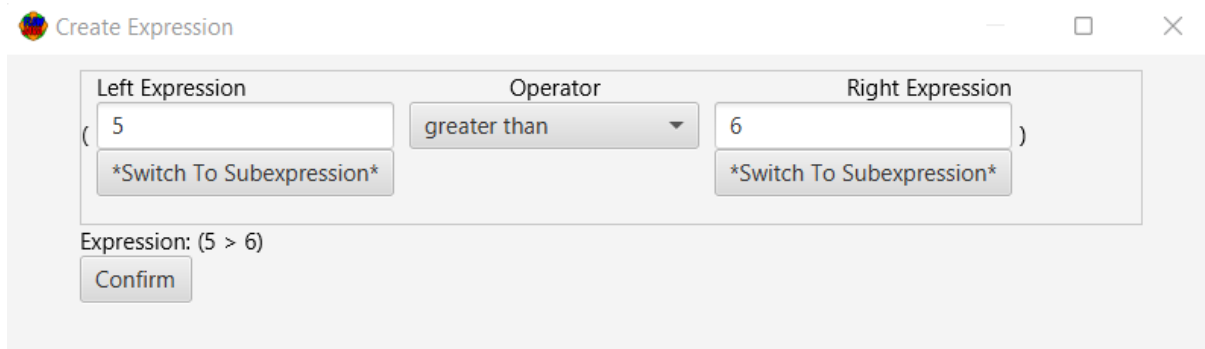
You can write full expressions out manually instead for the fields but the create expression window can assist with expression structures and operators. For information on Java operators go to https://docs.oracle.com/Javase/tutorial/Java/nutsandbolts/opsummary.html.

4.6 Key shortcuts

The following is a list of key shortcuts usable in FlowJava:

- Shift + scroll wheel = zoom
- Ctrl + click = deselect node
- Delete = delete node
- Middle mouse hold = pan

5. Vertex Types

The different vertex types in FlowJava are: variable declaration, array variable declaration, user input to variable, output, variable assignment, if statement, end if, while loop, end while, for loop, end for, invoke function, invoke other function and recurse. For each vertex there is important information displayed on the right sidebar of the user interface when it is selected.

For a flowchart to be run or converted it must be structured correctly. For a flowchart to be structured correctly, each vertex must have the required number of children and parents.

5.1 Variable Declaration

Variable declarations vertexes are composed of the variables type, the variables name and an expression for the variables value. The variable types compatible with Java are String, Boolean, Character, Integer, Double, Float, Long and Short.

Note: FlowJava uses the non-primitive versions of these data types. For information on these types go to https://docs.oracle.com/Javase/7/docs/api/index.html and go to the Java.lang package.

Variable names must conform to the Java variable naming conventions, for information on this go to https://www.oracle.com/Java/technologies/Javase/codeconventions-namingconventions.html.

Variable values must be valid to their data type. Unlike dynamic typed languages like python, Java is strongly type. Meaning you can't declare a variable x as a string and then give it an integer value.

Like many fields for vertex creation, you can enter expressions into the value field. Expressions must be valid Java expressions, for information on this go to https://docs.oracle.com/Javase/tutorial/Java/nutsandbolts/expressions.html.

Note: information inputted for variable declarations which is considered valid may still have errors relating to the semantics of the rest of the program or the syntax of Java. If this is the case the user will be shown an error window when they try to run or convert the program from the main scene.

Variable names must be unique and cannot be "userInputBr", "userInputString", "running", "semaphore", "customAlert" or "isResultPresent" as these are used in the program that runs the user created programs. If you would like to look at the source code of FlowJava check out the GitHub repo: https://github.com/cw523-01/SussexFYP-FlowJava.

5.2 Array Variable Declaration

Array variable declaration vertices are similar to variable declaration vertices except they are for creating arrays. Array variables can be declared by specifying the elements or length of the array. When you declare an array variable by length each element is initially set to null. The length of the arrays is always fixed by the number of elements in the declaration. Arrays can be of any of the same types as normal variables and all elements of the array must be of the same type. For more details on arrays go to https://docs.oracle.com/Javase/tutorial/Java/nutsandbolts/arrays.html.

5.3 User Input to Variable

User input to variable vertices are the only way to get user input for programs run in FlowJava. The vertices consist of a variable type and name with the value of the variable being determined by the user when the program is run.

Warning: you can't use br.readline() in other vertex expressions in FlowJava to get user input. Users are encouraged to alter the Java version of their programs outside of FlowJava to implement non-compatible features of Java.

A user input must be assigned to a new variable but it can then be assigned to an existing variable using a variable assignment.

When running converted Java programs the mechanism for getting user input is much simpler.

5.4 Output

User input to variable vertices are the only way to output values for programs run in FlowJava. The vertices consist of a single value for the expression to be output.

Warning: you can't use System.out.println() in other vertex expressions in FlowJava to output values. Users are encouraged to alter the Java version of their programs outside of FlowJava to implement non-compatible features of Java.

5.5 Variable Assignment

Variable assignment vertices are used to assign a value to a variable. You can only use a variable assignment to assign a new value to a variable that has already been declared. The value assigned to the variable must be of the same data type as the variable.

5.6 If Statement

If statement vertices are used to branch the program using an if statement (if-then statement). The vertices have a single value for the expression to evaluate. The expression must evaluate to a

boolean to be valid in Java. For more information on this go to
https://docs.oracle.com/Javase/tutorial/Java/nutsandbolts/if.html.

Warning: If statements vertices must have and true and false branch. If you wish for one of these branches to be empty then make a direct connection to the end if branch that links to the if statement vertex.

Note: There is no explicit notion of an else if in FlowJava. However, to implement this you can use another if statement inside the false branch of an if statement.

5.6.1 End If
End if vertices are used so the user can explicitly state the end of an if branch. End if statements are not used in Java but they are implied by the curly braces ({}) used for if statements.
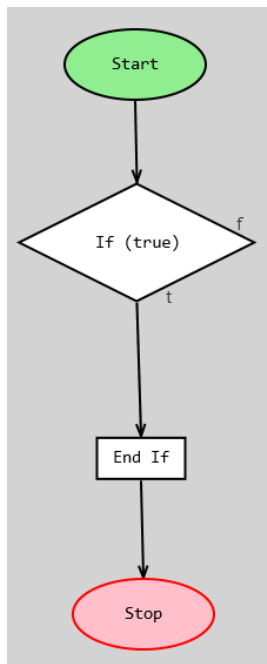
5.6.2 Structure
For a flowchart program in FlowJava to be considered valid, if statement vertices must be structured properly with end if vertices. The bodies of the two sperate if branches must all eventually lead to the end if vertex. E.g.
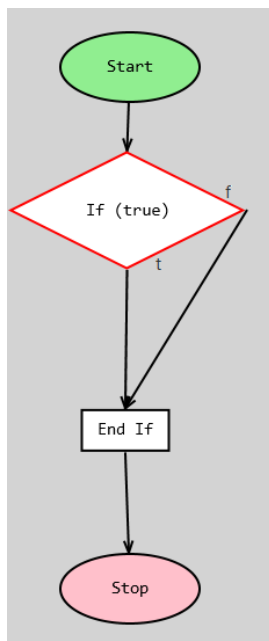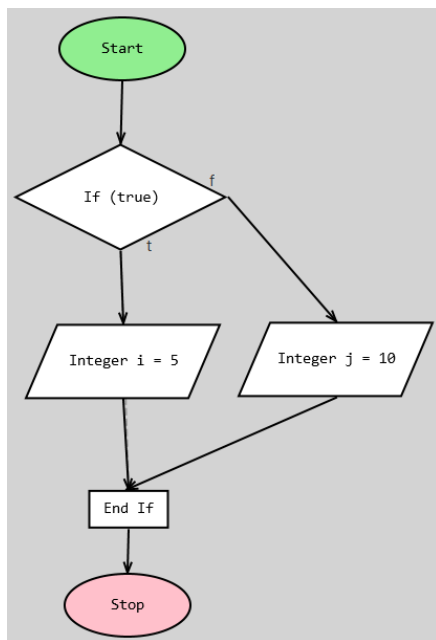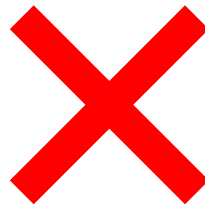

Valid

An end if vertex is connected to the if statement vertex it is connected to by a non-removable dotted line. If statements vertices must be structured properly only with their corresponding end if vertex.

5.7 While Loop

While loop vertices are used to loop the program using a while clause. The vertices have a single value for the expression to evaluate for looping. The expression must evaluate to a bool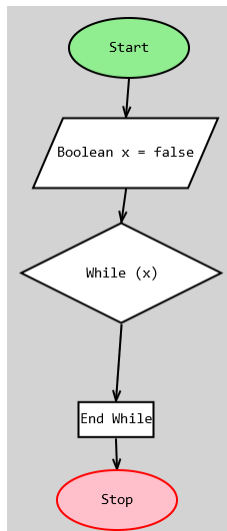ean to be valid in Java. For more information on this go to https://docs.oracle.com/Javase/tutorial/Java/nutsandbolts/while.html.

5.7.1 End While

End while vertices are used so the user can explicitly state the end of a while loop. End while statements are not used in Java but they are implied by the curly braces ({}) used for while loops.
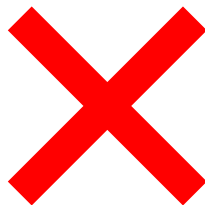
5.7.2 Structure

For a flowchart program in FlowJava to be considered valid, while loop vertices <u>must</u> be structured properly with end while vertices. The body of the while loop must eventually lead to the end if vertex. E.g.
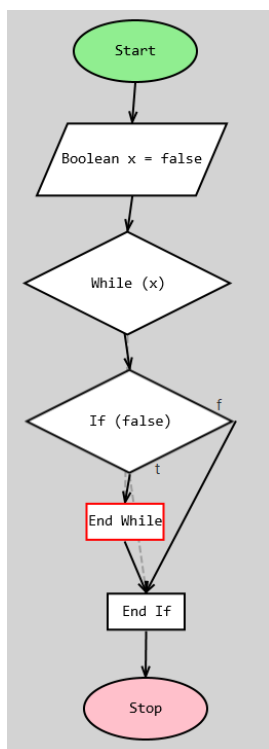
Valid

Invalid

5.8 For Loop

For loop vertices are used to loop the program using a for clause. There are three values for the three separate components of a for clause. The three components are the initialisation expression, condition expression (termination expression) and update expression (increment expression). The initialisation expression is executed before the loop is executed. The condition expression is evaluated each iteration to determine whether the program will continue looping. Each expression

must be valid in Java. The update expression is executed after each iteration. For more information on this go to https://docs.oracle.com/Javase/tutorial/Java/nutsandbolts/for.html.

5.8.1 End For
End for vertices are used so the user can explicitly state the end of a for loop. End for statements are not used in Java but they are implied by the curly braces ({}) used with for loops.

5.8.1 Structure
The structure requirements with for loop vertices and their end for vertex is identical to the requirements with while loop vertices and their end while vertex.

5.9 Invoke Function
Once a function is created using the function manager window you can invoke it from the main flowchart program using an invoke function vertex. The vertex has a value for the name of the function to call and a value for the parameter values. The values of the parameters in the function call must be valid for the parameters in the function's definition (e.g. same number of parameters and correct data types). For more information on this go to https://www.w3schools.com/Java/Java_methods_param.asp (functions are commonly called methods in Java documentation).

5.9.1 Invoke Other Function
Invoke other function vertices are identical to invoke function vertices except they can only be used within a function to call a different function.

5.9.2 Recurse
Recurse vertices are identical to invoke function vertices except they can only be used within a function to call the function it is inside. For more information on this go to https://www.w3schools.com/Java/Java_recursion.asp.

Warning: recursion is an easy way of causing an infinite loop so which should always be avoided.

6. Converted Programs
When you convert a flowchart program and downloaded it will download as a .Java file. These files can be edited in most text editing applications. Users are encouraged to download and modify their programs outside of FlowJava.

The converted programs use System.in and System.out for user input and output. This is a different mechanism to when the flowchart programs are run inside FlowJava. The reasoning behind this is that flowchart programs are specialised to run inside FlowJava. System.in and System.out are used in converted programs as it is one of the most basic forms of input and output in Java.

FlowJava does not support OOP (therefore all the functions in converted programs are static) which is a powerful tool in Java. Users are encouraged to use FlowJava to attain a grasp of the basics of Java and then move onto a different IDE that supports more complex Java features. For more information on OOP go to https://www.w3schools.com/java/java_oop.asp.

To run a converted program you must research how to run a java file on your system. This is commonly done by using an IDE or command line application.