ITI 1120 Fall 2013 - Assignment 3

Available: Sunday, Oct 20, 2013
Due: Sunday Nov 4, 2014, 10:00 pm

Instructions

This assignment is to be done INDIVIDUALLY. Follow the instructions in the lab manual for submitting assignments through the Virtual campus. The following are specific instructions for this assignment:

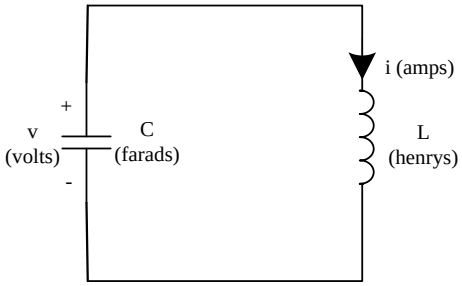
1) For question 1, provide a Word file A3Q1.docx file with the algorithms developed for the question.
2) For question 2, provide a Word file A3Q2.docx file with the algorithms developed for the question.
3) For question 3, submit the Java files A3Q3.java and Circuit.java.
4) Zip all the .docx and .java files in a3_xxxxxx.zip, where xxxxxx is your student number, and submit it through the Virtual Campus.

Your algorithms should be developed using the format used in class. In the Java code, use the coding structures presented in class. Do not use other structures, such as the switch and break statements.

Marking Scheme (total 100 marks)

- **Regulations and Standards: 5 marks**
- Question 1: 30 marks
- Question 2: 25 marks
- Question 3: 40 marks

Exploring an L-C electric circuit



Consider the electrical circuit to the right that contains the capacitor C, and the inductor L. The capacitor and inductor are devices that can store electrical energy: the capacitor stores energy in an electric field which is reflected in the voltage across it and the inductor stores energy in a magnetic field which is reflected by the current that passes through it. Thus when the current *i* is zero, the capacitor contains all of the energy in the circuit (the voltage *v* will be at its maximum) and when the voltage *v* is zero, the inductor will contain all the energy (the current *i* will be at its maximum). Energy is exchanged continuously between the 2 devices at a frequency of resonance.

The equations that relate the voltage *v*, the current *i*, the capacitor *C* and the inductor *L* are:

$$\frac{dv(t)}{dt} = -\frac{1}{C} i(t) \qquad \frac{di(t)}{dt} = \frac{1}{L} v(t) \qquad (1)$$

Applying rules of mathematics, circuit and calculus, the following equations can be obtained:

$$i(t) = I_0 \cos\left(\frac{1}{\sqrt{LC}} t + \phi\right) \qquad v(t) = -\frac{1}{\sqrt{LC}} L I_0 \sin\left(\frac{1}{\sqrt{LC}} t + \phi\right) \qquad (2)$$

where I_0 and ϕ are constants related to the initial values of *v* and *I*, that is *v(0)* and *i(0)*, the values of voltage and current at time zero (*t = 0.0*). Note that the analytical solution given by equations (2) show that the voltage and current are sinusoidal functions, which indicates that energy is exchanged between the capacitor and the inductor

In this assignment, you will be working with a numerical solution of the first equations (1) which is used when an analytical solution is not available. The Euler method is a numerical method used to evaluate differential equations using the computer (http://en.wikipedia.org/wiki/Euler_method). It can be applied to equations (1) to calculate the values of *v(t)* and *i(t)* à different values of *t*. First a time step *h* is defined, which determines how time *t* is incremented. The time t_0 is defined as the time $t = 0.0$ and then time $t_k = t_{k+1}$ for $k = 1, 2, 3, \ldots$ Thus if $h = 0.001$, the values for *t* are $t_0 = 0.0$, $t_1 = 0.001$, $t_2 = 0.002$, $t_3 = 0.003$, …. The values of v_k and i_k at these times of *t* are calculated with the following equations (obtained by applying the Euler method to equations (1)):

$$v_{n+1} = v_n - h\frac{1}{C} i_n \qquad i_{n+1} = i_n + h\frac{1}{L} v_n \qquad (3)$$

In summary:

a) The value of t_0 is 0.0, the time step *h* must be given as well as the initial values of voltage v_0 and current i_0.

b) Time is incremented with the time step, i.e., $t_k = h + t_{k-1}$ pour $k = 1, 2, 3, 4, \ldots$

c) The voltage *v(t)* is calculated at the times values with $v_{k+1} = v_k - h\frac{1}{C} i_k$

d) The current *i(t)* is calculated at the times values with $i_{k+1} = i_k + h\frac{1}{L} v_k$

e) Note that t_1, v_1 and i_1 are calculated using the values t_0, v_0 et i_0; that t_2, v_2 and i_2 are calculated using the values t_1, v_1 and i_1; that t_3, v_3 and i_3 are calculated using the values t_2, v_2 et i_2; and so on.

The following table lists the configuration variables for the circuit and the range of acceptable values for these variables:

Variable	**Unit**	**Minimum**	**Maximum**
v0	volts	0.0	5.0
i0	amps	0.0	0.005
C	farads	$1 \text{ X } 10^{-9}$	$1 \text{ X } 10^{-7}$
L	henrys	$1 \text{ X } 10^{-3}$	$1 \text{ X } 10^{-1}$

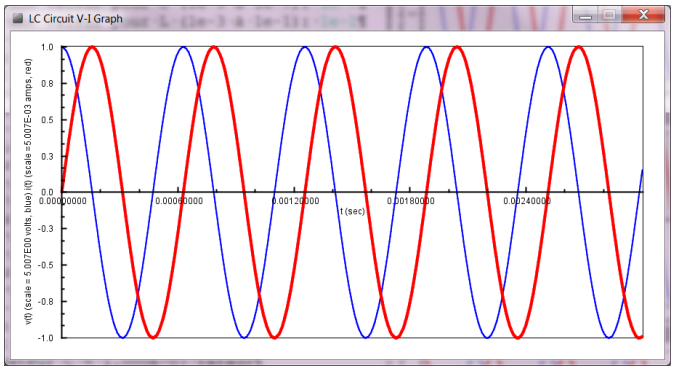
In addition to the values for the above configuration variables, the user shall provide values for the time step *h* and the maximum time value for plotting a graph (see the examples below).

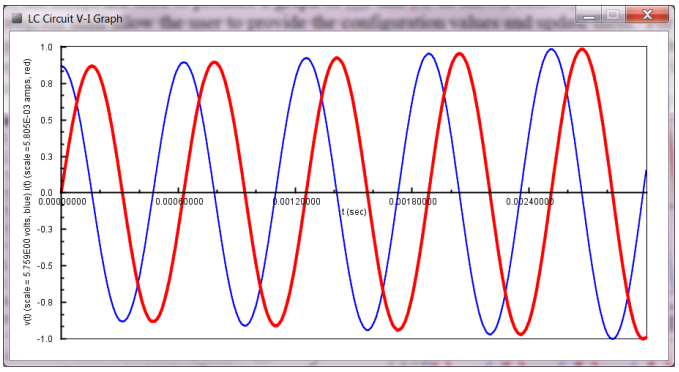
The objective of this assignment is to develop software that will calculate a sequence of values for time *t*, the voltage *v(t)*, and the current *i(t)*. These real values are stored in three different arrays. The content of the arrays are used to produce a graph of *v(t)* and *i(t)* as shown below.

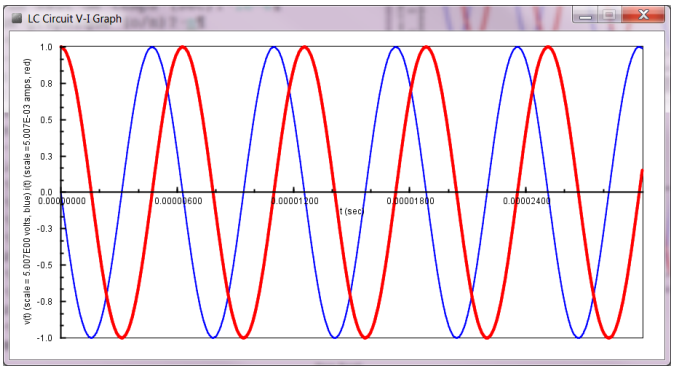
The program shall allow the user to provide the configuration values and update them. For each set of configuration values, the user is allowed to produce one or more graphs with different time values (time step and maximum time). An example of the program output is provided on the next page. Use it for the development of your program.

Example Output

```
ITI1120 Fall 2013, Assignment 3, Question 3
Name: Gilbert Arbez, Student# 81069665
Enter a value for v0 (0.0 to 5.0): -5.0
Bad value: -5.0
Enter a value for v0 (0.0 to 5.0): 5.0
Enter a value for i0 (0.0 to 0.005): 1
Bad value: 1.0
Enter a value for i0 (0.0 to 0.005): 0.0
Enter a value for C (1e-9 to 1e-7): 1e-7
Enter a value for L (1e-3 to 1e-1): 1e1
Bad value: 10.0
Enter a value for L (1e-3 to 1e-1): 1e-1
-----------------------------------------
Initial voltage = 5.0 volts
Initial current = 0.0 amps
Capacitor C = 1.000E-07 farads
Inductor L = 1.000E-01 henrys
-----------------------------------------
Reset configuration values (y/n)? n
Create a graph (y/n)? y
Enter a maximum time (sec): 0.003
Enter a time step (sec): 1e-8
Create a graph (y/n)? y
Enter a maximum time (sec): 0.003
Enter a time step (sec): 1e-6
Create a graph (y/n)? n
Do you want to quit (y/n)? n
-----------------------------------------
Initial voltage = 5.0 volts
Initial current = 0.0 amps
Capacitor C = 1.000E-07 farads
Inductor L = 1.000E-01 henrys
-----------------------------------------
Reset configuration values (y/n)? y
Enter a value for v0 (0.0 to 5.0): 0.0
Enter a value for i0 (0.0 to 0.005): 0.005
Enter a value for C (1e-9 to 1e-7): 1e-9
Enter a value for L (1e-3 to 1e-1): 1e-3
Create a graph (y/n)? 0.00003
Do you want to quit (y/n)? n
-----------------------------------------
Initial voltage = 0.0 volts
Initial current = 0.005 amps
Capacitor C = 1.000E-09 farads
Inductor L = 1.000E-03 henrys
-----------------------------------------
Reset configuration values (y/n)? n
Create a graph (y/n)? y
Enter a maximum time (sec): 0.00003
Enter a time step (sec): 1e-10
Create a graph (y/n)? n
Do you want to quit (y/n)? y
Program Terminated
```







Notes:

- Numerical methods are not perfect and contain errors. Note on the preceding page that the second graph does not show waves that are not sinusoidal. This happens when selecting a time step that is too large which augments the effect of the errors in the numerical equations. But be careful, if the time step is too small, the program will be slow and can even run out of memory.
- To help with your development of the algorithms, the following predefined algorithm is available:

  GIVENS : tArr *(reference to an array with the time values t0, t1, t2, ...)*
  vArr *(reference to an array with the voltage values v0, v1, v2, ...)*
  iArr *(reference to an array with the current values i0, i1, i2, ...)*
  n *(number of elements in each of the arrays)*

  RESULTS : *(none)*

  HEADER : graphVI(tblT, tblV, tblI, n)

  This algorithm defines how to create a graph with the two curves *v(t)* and *i(t)*.

- The above algorithm can be translated to the following method provided in the class `CircuitGraph`

```
/**
 * Method:  graphVI
 * Description:  Creates a graph with two curves: on for the voltage
 *               (in blue) and one for the current (in red).
 * Parameters:
 *          tArr – reference to array of time values
 *          vArr – reference to array of the values v(t)
 *          iArr - reference to array of the values i(t)
 *  Assumptions:
 *      The minimum value of time is found at the beginning of tArr.
 *      The maximum value of time is fount at the end of tArr.
 *      The values of vArr and iArr are both positive and negative.
 */
public static void graphVI(double [] tArr, double [] vArr, double [] iArr)
```

  To use this class and method, you must add the following line to the start of the Java file.

```
import iti1120graph.CircuitGraph;
```

  The provided method contains a bug or rather the library Processing used to create the method contains a bug. It is possible to create multiple graphs with the software of this assignment. But when one of the graphs is closed (that is a Processing window), the whole program is terminated as well as all graphs are closed.

- The method `graphVI` is implemented in the library *iti1120graph.jar* using a graph library *GW Optics* (http://www.gwoptics.org/processing/gwoptics_p5lib/) developed using the *Processing* programing environment (http://processing.org/). Three Java libraries (.jar files) are provided and contain the necessary software to create a graph:
  - *core.jar* : The *Processing* library.
  - *gwoptics.jar* : The *GW Optics* library.
  - *iti1120graph.jar* : The library that contains the class `CircuitGraph` to be used in your software.
  - Annex B describes how to add these library files to DrJava.

Question 1 – Design of the User Interface

Appendix A provides two algorithms for interacting with the user; the main algorithm and the algorithm *getCircuitConf*. The main algorithm interacts with the user as follows.

- Request from the user the circuit configuration values by calling the algorithm *getCircuitConf*. This is done at the very start of the program and within the outer loop of the algorithm.
- The outer loop of the algorithm allows the user to repeat two steps. The first is to prompt the user to reset the values of the components. To reset the configuration values, the algorithm *getCircuitConf* is called. The second step is to create a graph as described in the next point.
- An inner loop is used to create one or more graphs. To create a graph, the user is asked to provide a maximum time and a time step for simulation. These values are used to generate a time array by calling the algorithm *genTimeArr*. The algorithm *genVArr* is called to create and fill another array with the values of *v(t)*. Then the algorithm *genIArr* is called to create and fill another array with the values of *i(t)*. These called algorithms are developed in Question 2. Finally the graph is created with a call to the predefined algorithm *graphVI*. The inner loop prompts the user to create another graph and thus allows the user to create a number of graphs for different time values before resetting the configuration values.

Note also that the configuration values (*v0, i0, C, L*) are all stored in an array. The four constants VIX, CIX, LIX, and RIX are defined globally so that all algorithms can use these constants as indices into the configuration value array.

The *getCircuitConf* algorithm is used to prompt the user for values of each component in the circuit. You will notice that for each such value, there is a pattern in the steps taken to interact with the user:

- Setup a loop that monitors the value of flag (a Boolean variable)
- Prompt the user for a value (and store in the array)
- Check the value provided by the user
- If the value is valid (within expected range), set flag to false (to break out of loop)
- If the value is not valid, print an error message.

Note that these algorithms are rather complex and long. It should then be possible to identify tasks in each algorithm that can be captured in other algorithms. Then the original algorithms can call these new algorithms and consequently the original algorithms will become simpler.

a) Change the main algorithm into the following algorithms:
   i. An algorithm that prompts the user to reset the configuration values. This algorithm is given the reference to the existing configuration array. First the current values are displayed (see the example output). If the user chooses to reset the configuration values, a reference to a new array (created with a call to the *getCircuitConf* algorithm) is returned, otherwise the reference to the original array is returned.
   ii. An algorithm that allows the user to create one or more graphs.
   iii. Update the *main* algorithm to make calls to the above new algorithms.

b) Change the *getCircuitConf()* algorithm into the following algorithms.
   i. Define an algorithm that receives as givens a prompt string, a minimum value and a maximum value. The algorithm prompts the user until it obtains a valid value. The result of the algorithm is the valid value read in from the user.
   ii. Revise the algorithm *getCircuitConf* to make the necessary calls the algorithm developed above.

Question 2 – Design of the Circuit Software

The circuit software consists of algorithms computing values of *t*, *v(t)*, and *i(t)* used in creating a graph.

a) Develop an algorithm, *genTimeArray*, that generates an array of time values given a maximum length of time (*tMax*) and a time step (*h*). Thus the array will contain the values $t_0, t_1, t_2, \ldots, t_{max}$, where $t_0 = 0$, and $t_k = t_{k-1} + tStep$ for $k = 1, 2, 3, \ldots$.

b) Develop an algorithm, *genVArr*, that computes the values of *v(t)* with the following givens: the number of values to computes (*n*, hint: the value of *n* is provided by the algorithm *genTimeArray*), a time step *h*, and a reference to an array that contains the configuration values. The values of *v(t)* are stored in a new array. The result of the algorithm is a reference to this new array. The values of *v(t)* are calculate with the numerical equations described in the assignment introduction (Exploring the L-C Circuit). Note that it is not necessary to use an array for storing the values of *i(t)*. Rather all that is needed is to maintain a value for i_{k-1}.

c) Develop an algorithm, *genIArr*, that computes the values of *i(t)* with the following givens: the number of values to computes (*n*, hint: the value of *n* is provided by the algorithm *genTimeArray*), a time step *h*, and a reference to an array that contains the configuration values. The values of *i(t)* are stored in a new array. The result of the algorithm is a reference to this new array. The values of *i(t)* are calculate with the numerical equations described in the assignment introduction (Exploring the L-C Circuit). Note that it is not necessary to use an array for storing the values of *v(t)*. Rather all that is needed is to maintain a value for v_{k-1}.

Question 3 - Implementation and Testing

a) Translate the algorithms developed in question 1 to Java methods and place them into the class A3Q3, stored in the file *A3Q3.java*.

b) Translate the algorithms developed in question 2 to Java methods and store them in the class Circuit, stored in the file *Circuit.java*. Define the constants VIX, CIX, LIX and RIX in this class outside the methods using the following syntax (shown for VIX):

```
public static final int VIX = 0;  // Index to the battery value in
                                  // the components value array
```

The keyword *final* makes the variable *VIX* a constant with the value 0. These constants will be available to all methods in the *Circuit* class. As well, the constants can be accessed from outside the class using the expressions like *Circuit.VIX* (this is similar to the constant *Math.PI* available in the *Math* class).

Appendix A - Algorithms for Question 1

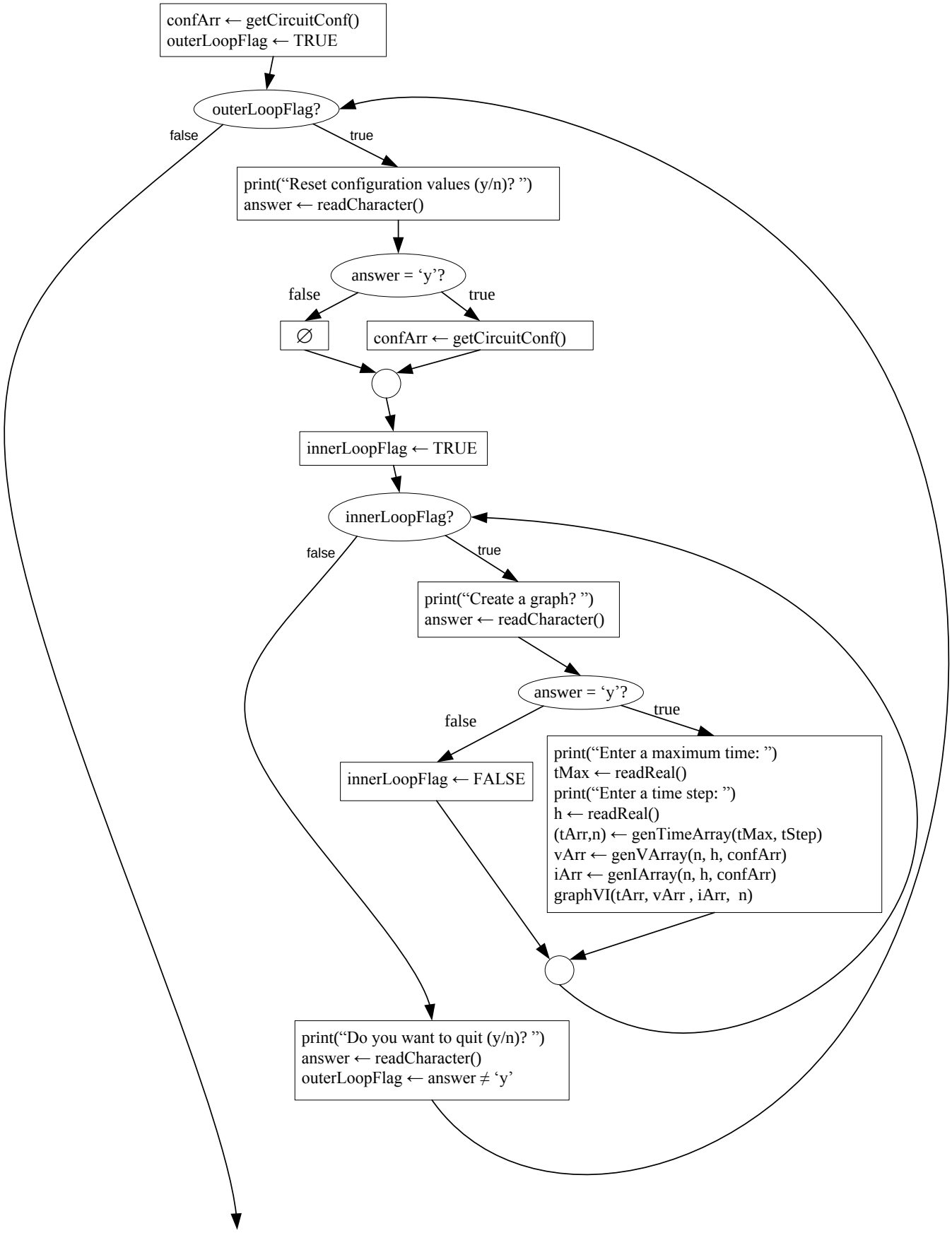
Global constants, these constants are accessible to all algorithms

VIX	*(constant 0 – index to the initial voltage value in configuration value array)*
IIX	*(constant 1 – index to the initial current value in configuration value array)*
CIX	*(constant 2 – index of capacitor value in configuration value array)*
LIX	*(constant 3 – index of inductance value in configuration value array)*

GIVENS: *(none)*
RESULTS: *(none)*
INTERMEDIATES:

confArr	*(reference to component value array)*
outerLoopFlag	*(flag to control outer loop)*
innerLoopFlag	*(flag to control inner loop)*
answer	*(character to record y/n answers from user)*
tMax	*(variable for maximum time of simulation)*
h	*(time step for simulation)*
tArr	*(reference to time array)*
vArr	*(reference to voltage array)*
iArr	*(reference to current array)*
n	*(number of elements in the arrays referenced by tArr, vArr, and iArr)*

HEADER:
main()
BODY: (see next page)


confArr ← getCircuitConf()
outerLoopFlag ← TRUE
outerLoopFlag?
false
true
print(“Reset configuration values (y/n)? ”)
answer ← readCharacter()
answer = ‘y’?
false
true
∅
confArr ← getCircuitConf()
innerLoopFlag ← TRUE
innerLoopFlag?
false
true
print(“Create a graph? ”)
answer ← readCharacter()
answer = ‘y’?
false
true
innerLoopFlag ← FALSE
print(“Enter a maximum time: ”)
tMax ← readReal()
print(“Enter a time step: ”)
h ← readReal()
(tArr,n) ← genTimeArray(tMax, tStep)
vArr ← genVArray(n, h, confArr)
iArr ← genIArray(n, h, confArr)
graphVI(tArr, vArr , iArr, n)
print(“Do you want to quit (y/n)? ”)
answer ← readCharacter()
outerLoopFlag ← answer ≠ ‘y’

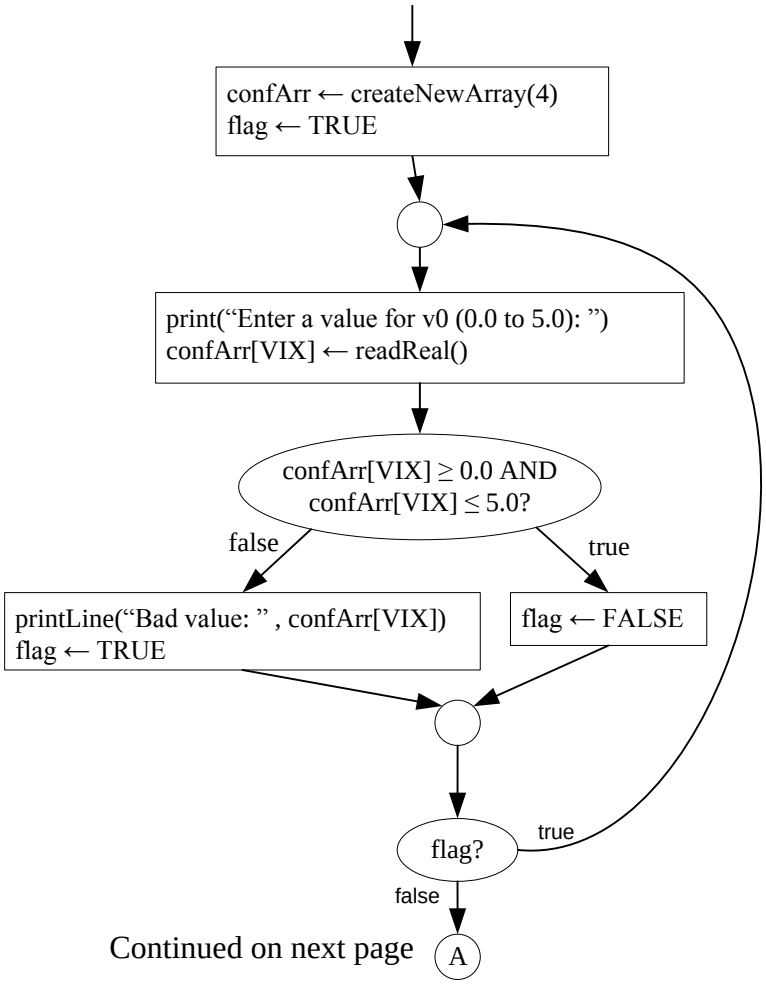
GIVENS: *(none)*
RESULTS:
confArr *(Reference to component array with values)*

INTERMEDIATES:
flag *(flag to monitor when user has entered a valid value)*

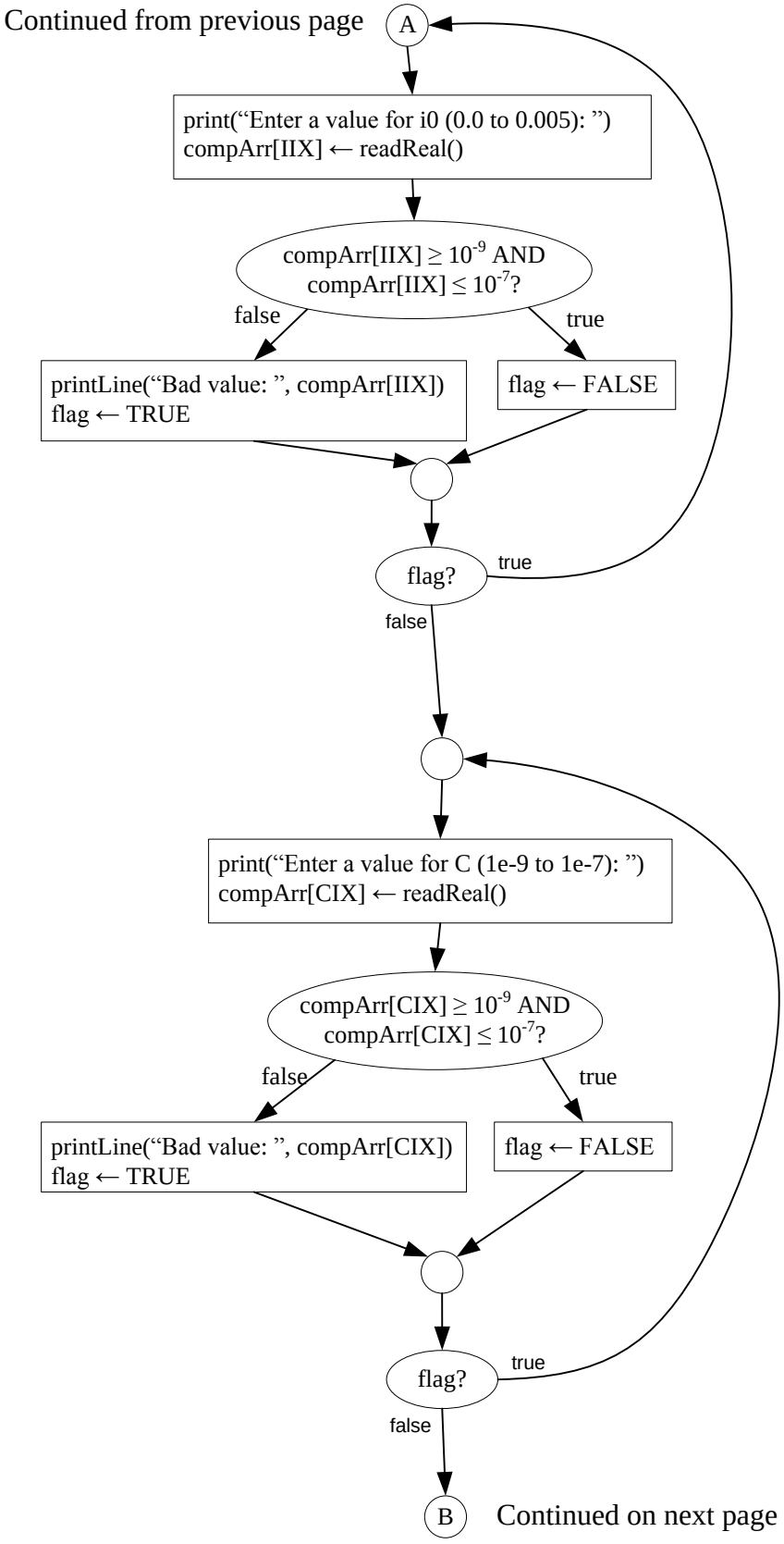
CONSTRAINTS:
confArr[VIX] has values ranging between 0.0 and 5 volts
confArr]RIX] has values ranging between 0.0 and 0.005 amps
confArr[CIX] has values ranging between 10^{-9} and 10^{-7} farads
confArr[LIX] has values ranging between 10^{-3} and 10^{-1} henrys

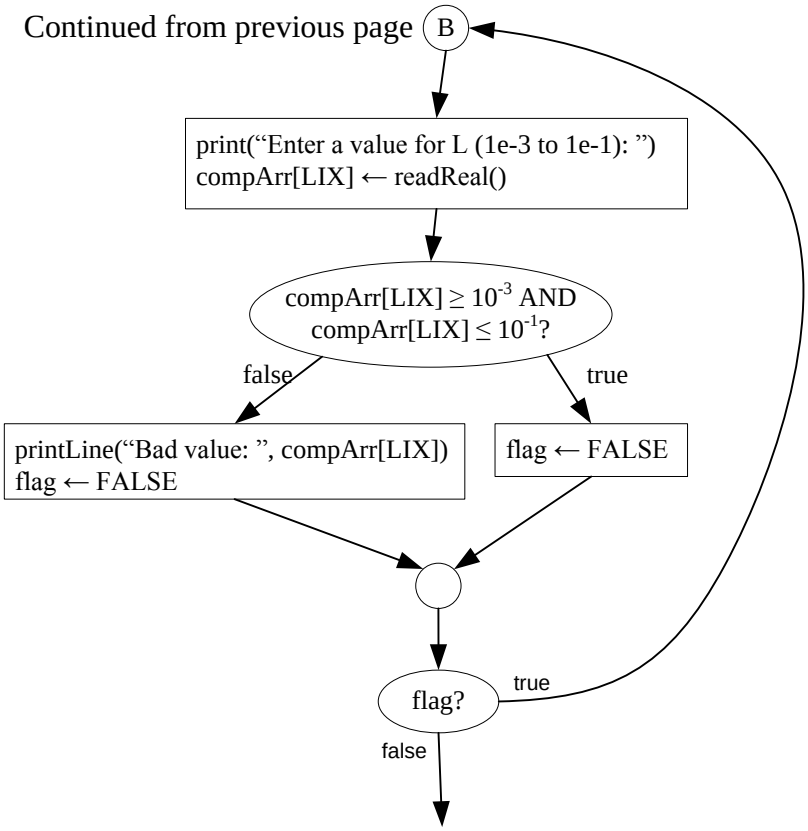
HEADER:
confArr ← getCircuitConf()

BODY:



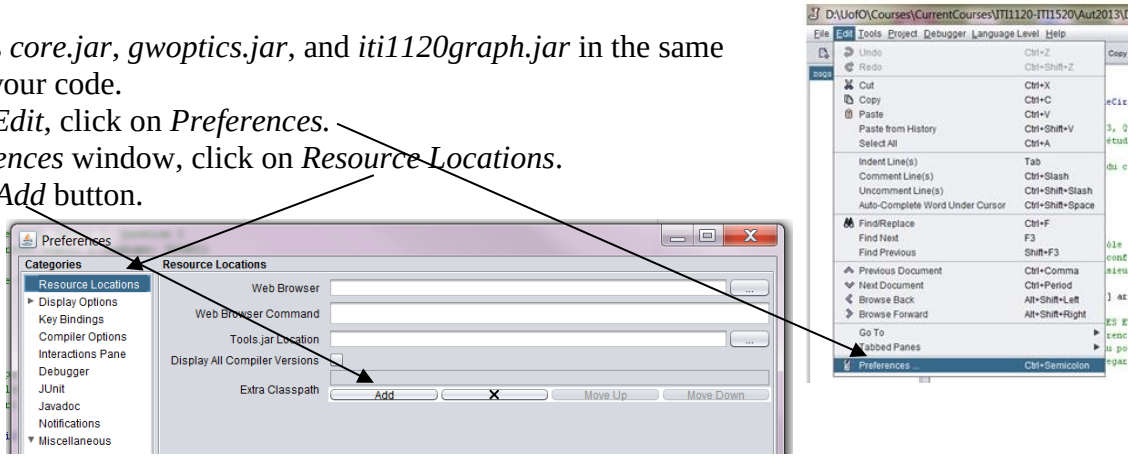
Continued on next page


Continued from previous page
A
print("Enter a value for i0 (0.0 to 0.005): ")
compArr[IIX] ← readReal()
compArr[IIX] ≥ 10-9 AND
compArr[IIX] ≤ 10-7?
false
true
printLine("Bad value: ", compArr[IIX])
flag ← TRUE
flag ← FALSE
flag?
true
false
print("Enter a value for C (1e-9 to 1e-7): ")
compArr[CIX] ← readReal()
compArr[CIX] ≥ 10-9 AND
compArr[CIX] ≤ 10-7?
false
true
printLine("Bad value: ", compArr[CIX])
flag ← FALSE
flag?
true
false
B
Continued on next page

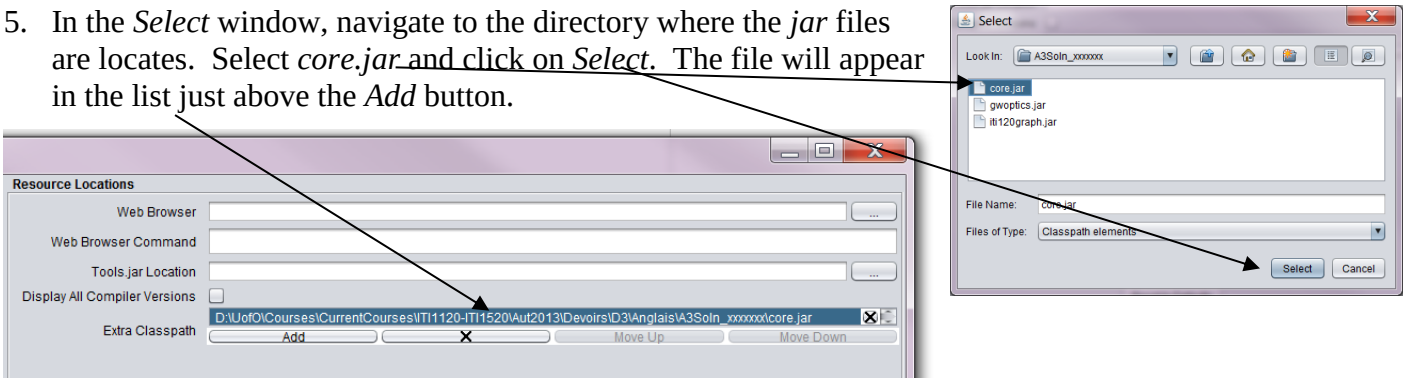

Continued from previous page
B
print("Enter a value for L (1e-3 to 1e-1): ")
compArr[LIX] ← readReal()
compArr[LIX] ≥ 10-3 AND
compArr[LIX] ≤ 10-1?
false
true
printLine("Bad value: ", compArr[LIX])
flag ← FALSE
flag ← FALSE
flag?
true
false

Annex B – Adding Java libraries (packages) to Dr Java

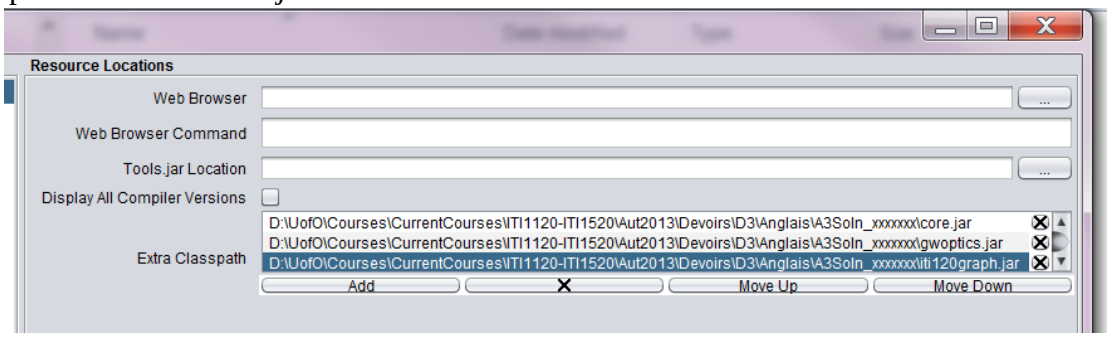
1. Save the files *core.jar*, *gwoptics.jar*, and *iti1120graph.jar* in the same directory as your code.
2. In the menu *Edit*, click on *Preferences*.
3. In the *Preferences* window, click on *Resource Locations*.
4. Click on the *Add* button.



5. In the *Select* window, navigate to the directory where the *jar* files are locates. Select *core.jar* and click on *Select*. The file will appear in the list just above the *Add* button.



6. Repeat to add the other *jar* files.



7. Click on *OK* to close the *Preferences* window.