**Technical Manual for Zeus Implementing the Vehicle Routing**

**Problem with Time Window Constraints (VRPTW)**

Eric McAlpine, Marc Sensenich, Aaron Weckerly

December 4th, 2014

**Table of Contents**

1. **Overview 3**
2. **System Requirements 3**
3. **Assumptions 3**
4. **Installation of the Java Development Kit 4**
5. **Installation of the Eclipse IDE 5**
6. **VRPTW Packages and Class Files 6**
7. **Setting up the Zeus Project 10**
8. **Running the VRPTW in Zeus 15**
   1. **Selecting a Heuristic 16**
   2. **Utilizing the User Interface 17**
   3. **Recursive Processing 18**
9. **Developing Heuristics 19**
10. **Overview**

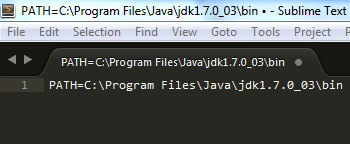
This project utilizes the Zeus framework to compute results for the Vehicle Routing Problem with Time Windows. This document will provide an in depth look into the technical details for the implementation of this project on another machine. This document will step through the setup of the development environment, outline the details of how to run a heuristic that is currently programmed into the project, demonstrate the functionality of the graphical user interface, inform the reader of how to process a large set of files using a single function call, and outline to the user of how they are able to implement their own heuristics utilizing the framework and examples provided.

1. **System Requirements**
   1. Some sort of Intel processor
   2. Microsoft Windows 7 or later
   3. Microsoft Excel or compatible software
   4. A Java Development Environment
      1. Java Development Kit 1.X.X
      2. Eclipse Luna IDE
2. **Assumptions**
   1. It is assumed that the reader of this manual has the necessary privileges to modify Windows environment variables
   2. It is assumed that the reader of this manual has internet access to retrieve necessary resources
   3. It is assumed that the user has a fundamental understanding of the Java Programming Language
   4. It is assumed that the reader has already obtained a copy of the Zeus VRPTW Workspace
3. **Installation of the Java Development Kit**

* **Installation of the Java Development Kit -** By following the steps outlined below the JDK will be downloaded and installed onto the current machine

1. Download the latest version of the [Java Development Kit 7](http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html)
2. Run the JDK 7 Installer and Follow the Installation Steps
   1. If the “Open” option was selected when initiating the download of the JDK, the installer will start up automatically
   2. If the “Save” option was selected when initiating the download of the, navigate to the directory where the JDK was saved to and double click on the file to run the installer

* **Setting the PATH variable for the Java Development Kit -** By following the steps outlined below a PATH variable will be set which will allow the system to recognize the path to the JDK
  + Temporarily Setting the PATH Variable with a Batch File
    1. Open up your favorite text editor
    2. Type in the following line
       - PATH = C:\<path to Java>\jdk1.7.X\_XX\bin
         * Note: please replace the Xs with the numerals of the Java Development Kit



* + 1. Save the file with the extension “<filename>.bat”
    2. After saving the file, navigate to the directory holding the file and double click the file to execute the batch file. This will automatically set the PATH variable.
       - Note: this will be temporary and when the system is rebooted the PATH variable needs to be set once more
  + Permanently Setting the PATH Variable
    1. On the Windows Taskbar: click the Windows Button and select “Control Panel”
    2. Within the “Control Panel” dialog box select “System”
    3. Within the “System” dialog box select “Advanced system settings”
    4. Within the “Advanced system settings” click on the “Environment variables....” button
    5. Within the “Environment variables” dialog box double-click on ‘Path’ or click on ‘Path’ and then the ‘Edit” button
    6. Update the ‘Variable value’ field with the path. If you are uncertain where to insert this, please append it to the of the current ‘Path’ variable

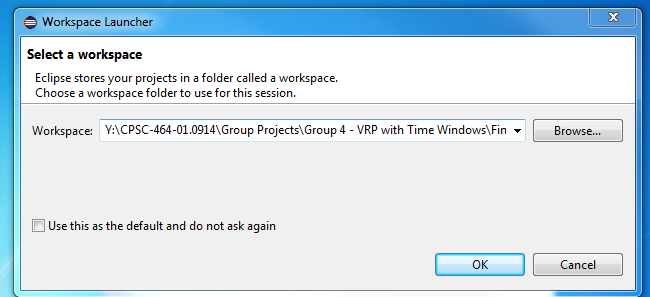
1. **Installation of the Eclipse IDE**

* **Downloading Eclipse Luna -** By following the steps outlined below the reader will have successfully downloaded the Eclipse Luna IDE.

1. Download the [Eclipse Luna IDE Zip](https://eclipse.org/downloads/)
2. Extract the contents of the Zip file to the directory of your choice.
   1. It is recommended to that you extract the archive in the root of the hard drive, e.g. “C:\eclipse”

* **Running Eclipse Luna for the First Time -** By following the steps outlined below the reader will have accessed Eclipse Luna for the first time and set up a default workspace

1. Run Eclipse Luna by executing C:\<path to eclipse parent directory>\eclipse\eclipse.exe
2. Identify the location for your Eclipse workspace
   1. Note: This is where local copies of your projects will be stored
   2. Note: This can be considered the default workspace, however you are not limited to only working within this workspace



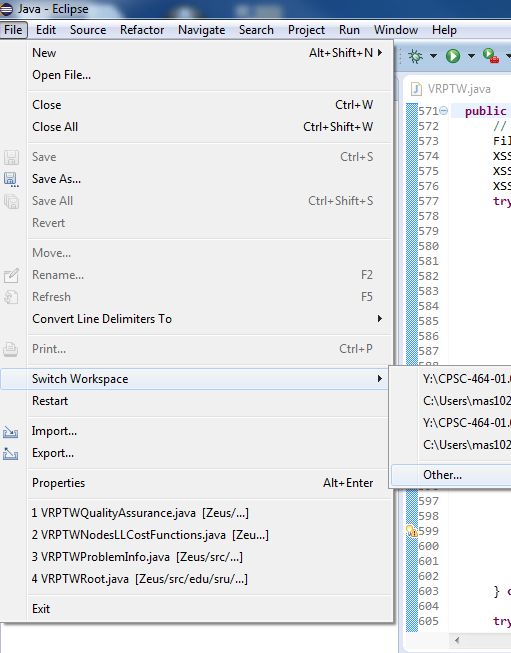
1. **VRPTW Packages and Class Files**

* **The following is a listing of and explanation class files utilized by Zeus in solving Vehicle Routing Problems with Time Window Constraints-** After completing this section the reader will have a base understanding of the class files utilized by Zeus in solving the VRPTW
  + Package: **edu.sru.thangiah.zeus.vrptw**
    - **VRPTWAttributes.java**
      * The VRPTWAttributes class extends the parent class the Zeus Core Attributes. This class holds all of the values calculated by the Cost Functions, both from the parent class and VRPTWAttributes
    - **VRPTWDepot.java** 
      * **This class maintains information regarding the depots utilized in the problem such as: geographical location, depot index, and trucks associated with a depot**
    - **VRPTWDepotLinkedList.java**
      * This class is an implementation of a linked list of VRPTWDepot(s). In the VRPTW there is only one depot that is stored in this list, however in other types of VRP and various datasets there is the potential for more than one depot
    - **VRPTWFeasiblity.java**
      * This class is utilized to check if a given route is feasible given the constraints placed when defining the problem. These constraints may include: distance, travel time, and capacity.
    - **VRPTWNodes.java**
      * This class represents a shipment to a customer placed on a route. An instance of this class holds numerous variables relating to this shipment and the problem itself; these include: wait time, insertion cost, and the nodes it is between in the route
    - **VRPTWNodesLinkedList.java**
      * This class is an implementation of a linked list of VRPTWNodes. This is a representation of a route being taken by a truck with the head and tail of the list being the depot.
    - **VRPTWProblemInfo.java**
      * This class holds problem specific info for the VRPTW and for Zeus itself. One of the key items that is held in this class ar the Cost Functions utilized within the VRPTW to calculate the cost of routes
    - **VRPTWSettings.java**
      * This is an inheriting class of Settings from Zeus Core. Since this is a set of global settings for Zeus there should be no problem specific information within this class.
    - **VRPTWShipment.java**
      * This class will store all of the information related to a shipment represented in the VRPTW
    - **VRPTWShipmentLinkedList.java**
      * This class is an implementation of a linked list of VRPTWShipments. This is a representation all of the customers that are to be serviced within the problem set
    - **VRPTWTruck.java**
      * This class holds all of the information about the trucks utilized within the problem. This information includes: the truck’s depot, a VRPTWNodesLinkedList representing the route, and the type of truck that it is
    - **VRPTWTruckLinkedList.java**
      * This class is an implementation of a linked list of VRPTWTrucks. This is a representation all of the trucks that are associated with a single depot
    - **VRPTWTruckType.java**
      * This class is a representation of a truck utilized within the problem. This has attributes such as: max duration, max capacity, and max distance
  + Package: **edu.sru.thangiah.zeus.vrptw.vrptwcostfunctions**
    - **VRPTWAbstractCostFunctions**
      * This class outlines the basic cost functions for all of the other classes in this package relating to total distance, total travel time, total wait time, and total service time
    - **VRPTWDepotCostFunctions**
      * This class implements the abstract cost functions for the VRPTWDepot class and finds the costs associated with a particular depot.
    - **VRPTWDepotLLCostFunctions**
      * This class implements the abstract cost functions for the VRPTWDepotLinkedList class, which finds the cost of all the entire solution.
    - **VRPTWNodesLLCostFunctions**
      * This class implements the abstract cost functions for the VRPTWNodesLinkedList class, which finds the costs associated with a route.
    - **VRPTWTruckCostFunctions**
      * This class implements the abstract cost functions for the VRPTWTruck class, which finds the cost associated with a truck.
    - **VRPTWTruckLLCostFunctions**
      * This class implements the abstract cost functions for the VRPTWTruckLinkedList class, which finds the costs associated a set of routes.
  + Package: **edu.sru.thangiah.zeus.vrptwqualityassurance**
    - **VRPTWQADepot**
      * This class is a recreated depot correlating to the original solution’s depot and is used to verify that the original solution meets the constraints given.
    - **VRPTWQADepotLinkedList**
      * This class contains a recreation of all the depots used in the solution to verify that the solution meets all of the constraints.
    - **VRPTWQANode**
      * This class is a recreated customer correlating to the original route, used to verify that the original route is staying within its constraints
    - **VRPTWQANodeLinkedList**
      * This class contains the routed shipments for a recreated route and is used to verify the integrity of the original created route.
    - **VRPTWQAShipment**
      * This class holds all the information for a given customer. The data should be identical to VRPTWShipment, but the redundancy reduces errors.
    - **VRPTWQAShipmentLinkedList**
      * This class holds the input data that the Quality Assurance file uses to compare to the routed solution.
    - **VRPTWQATruck**
      * This class mimics a VRPTWTruck in order to confirm the validity of the contained route as defined by the constraints in the input file.
    - **VRPTWQATruckLinkedList**
      * This class emulates a VRPTWTruckLinkedList for the purpose of double-checking the validity of the contained routes
    - **VRPTWQualityAssurance**
      * This class double checks the validity of the route by recreating the routes and re-reading the data file.

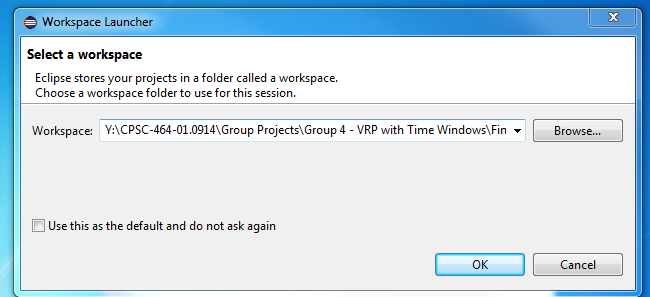
1. **Setting up the Zeus Java Project**

* **Setting up the Zeus workspace -** After completing this section the reader will have copied the Zeus workspace into the Eclipse IDE

1. Open the Zeus workspace in the Eclipse IDE
   1. If the reader already has Eclipse running
      1. Select ‘File’ from the navigation bar and then hover over ‘Switch workspace’. Click on ‘Other...’ in the drop-down menu.

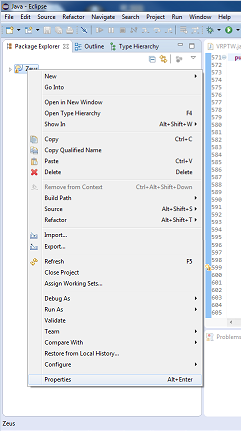


* + 1. Within the ‘Workspace Launcher’ type the path to the Zeus workspace in the ‘Workspace’ form element or if it is not known click on the ‘Browse...’ button and use the File structure to find the Zeus workspace folder

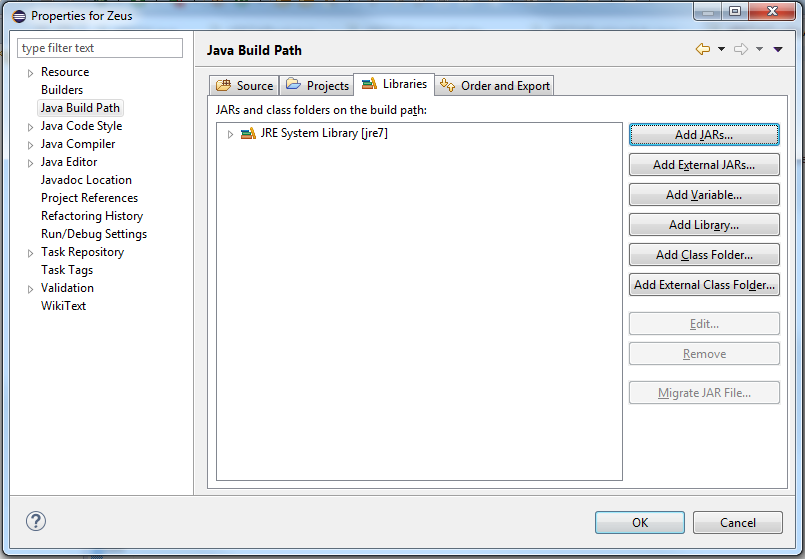


* + 1. After selecting the Zeus workspace click the ‘OK’ button and Eclipse will restart and the Zeus workspace will be loaded
  1. If the reader is not currently running Eclipse
     1. Launch the Eclipse IDE by clicking on the executable for Eclipse.
     2. Within the ‘Workspace Launcher’ type the path to the Zeus workspace in the ‘Workspace’ form element or if it is not known click on the ‘Browse...’ button and use the File structure to find the Zeus workspace folder
     3. After selecting the Zeus workspace click the ‘OK’ button and Eclipse will restart and the Zeus workspace will be loaded

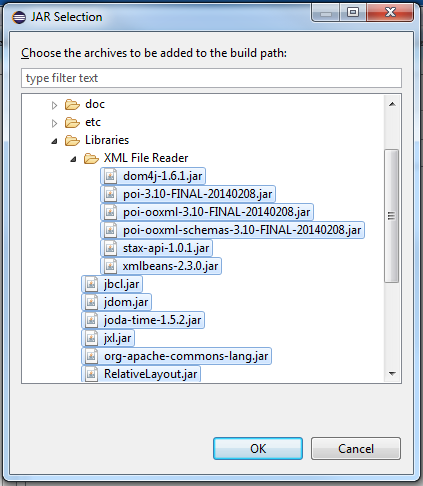
1. Importing the libraries utilized by Zeus
   1. Right-click on the project name in the Package Explorer and select ‘Properties’ in the drop-down menu or in the navigation menu click on ‘Project’ and select ‘Properties’ in the drop-down menu



* 1. In the ‘Properties for Zeus’ dialog box select ‘Java Build Path’ in the navigation menu on the left side of the dialog box
  2. Within the ‘Java Build Path’ pane, select the ‘Libraries’ tab
  3. On the right-hand side of the libraries tab click on the ‘Add JARs...’ button
     1. By selecting this option the path of the libraries will be relative to the project which allows for portability



* 1. In the ‘JAR Selection’ dialog box expand the ‘Zeus’ tree. Within the ‘Zeus’ file structure select and expand the ‘Libraries’ directory and expand the ‘XML File Reader’ directory. Within these directories select the following JARs:
* dom4j-1.6.1
* poi-3.10-FINAL-20140208
* poi-ooxml-3.10-FINAL-20140208
* poi-ooxml-schema-3.10-FINAL-20140208
* stax-api-1.0.1
* xmlbeans-2.3.0
* jbcl
* jdom
* joda-time-1.52
* jxl
* org-apache-commons-lang
* RelativeLayout
* xercesImpl-2.9.1
* xmlbeans-2.4.0
* zeuscore
* zeusqualityassurance

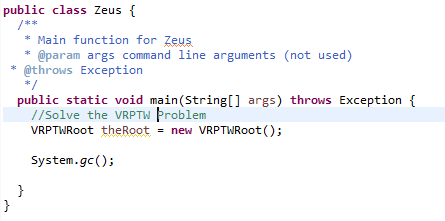


* 1. Click the ‘Ok’ button in ‘JAR Selection’ dialog box and then click the ‘Ok’ button in the ‘Properties for Zeus’ to finalize the import of the libraries

1. **Running the VRPTW in Zeus**

* **Setting up the Zeus class to run the Vehicle Routing Problem with Time Window Constraints -** After completing this section the reader will have set up the main class in Zeus to run the VRPTW

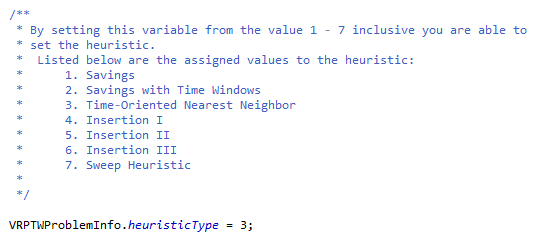
1. With the Eclipse IDE launched with the Zeus Workspace expand the Zeus project in the Package Explorer
   1. Note: if the Package Explorer is not in view on the workspace click ‘Window’ on the navigation bar, in the drop-down menu hover on ‘Show View’, in this menu click on ‘Package Explorer’ or press ‘Alt+Shift+Q, and P’ to open the view
2. With the ‘Zeus’ project expanded: expand the ‘src’ directory, within the ‘src’ directory expand the ‘edu.sru.thangiah.zeus’ package, and within that package double-click on ‘Zeus.java’ to open the source code
3. With the ‘Zeus.java’ file open in the editor ensure that the file has imported ‘edu.sru.thangiah.zeus.vrptw.VRPTWRoot’ and that in ‘public static void main()’ that it instantiates a new ‘VRPTWRoot’



1. If ‘Zeus.java’ is properly implemented then Zeus may be run for the VRPTW by either clicking on the ‘Run’ button or by pressing ‘Ctrl+F11’

* **Selecting different heuristics for the VRPTW in Zeus -** After completing this section the reader will learned how to set which heuristic will be utilized in the VRPTW

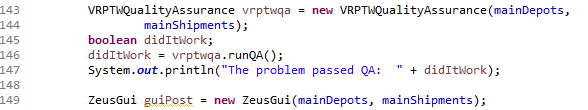
1. With the Eclipse IDE launched with the Zeus Workspace expand the Zeus project in the Package Explorer
   1. Note: if the Package Explorer is not in view on the workspace click ‘Window’ on the navigation bar, in the drop-down menu hover on ‘Show View’, in this menu click on ‘Package Explorer’ or press ‘Alt+Shift+Q, and P’ to open the view
2. With the ‘Zeus’ project expanded: expand the ‘src’ directory, within the ‘src’ directory expand the ‘edu.sru.thangiah.zeus.vrptw’ package, and within that package double-click on ‘VRPTWRoot.java’ to open the source code
3. With the ‘VRPTWRoot.java’ source code open in the editor navigate to line 49 ‘VRPTWProblemInfo.heuristicType = 1;’



1. Listed above this line is a list of heuristics available in Zeus for the VRPTW
2. By setting ‘VRPTWProblemInfo.heuristicType’ equal to a value from 1 - 7 the user may select which heuristic to use with the VRPTW
3. After selecting the heuristic, Zeus may be run for the VRPTW by either clicking on the ‘Run’ button or by pressing ‘Ctrl+F11’

* **Enabling and disabling the GUI -** After completing this section the reader will have learned how to enable and disable the GUI in Zeus for the VRPTW

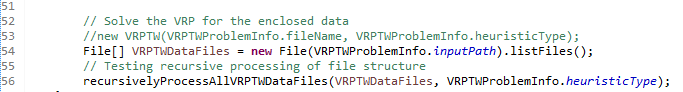
1. With the Eclipse IDE launched with the Zeus Workspace expand the Zeus project in the Package Explorer
   1. Note: if the Package Explorer is not in view on the workspace click ‘Window’ on the navigation bar, in the drop-down menu hover on ‘Show View’, in this menu click on ‘Package Explorer’ or press ‘Alt+Shift+Q, and P’ to open the view
2. With the ‘Zeus’ project expanded: expand the ‘src’ directory, within the ‘src’ directory expand the ‘edu.sru.thangiah.zeus.vrptw’ package, and within that package double-click on ‘VRPTW.java’ to open the source code
3. With the ‘VRPTW.java’ source code open in the editor navigate to line 127 ‘ZeusGui guiPost = new ZeusGui(mainDepots, mainShipments);’



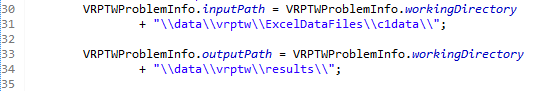
1. By commenting out this line the GUI will not be enabled and by leaving this line uncommented the GUI will be enabled
2. After selecting whether the GUI will be enabled, Zeus may be run for the VRPTW by either clicking on the ‘Run’ button or by pressing ‘Ctrl+F11’

* **Batch Processing of Problem Sets -** After completing this section the reader will have learned how to set Zeus to run entire problem sets during on run

1. With the Eclipse IDE launched with the Zeus Workspace expand the Zeus project in the Package Explorer
   1. Note: if the Package Explorer is not in view on the workspace click ‘Window’ on the navigation bar, in the drop-down menu hover on ‘Show View’, in this menu click on ‘Package Explorer’ or press ‘Alt+Shift+Q, and P’ to open the view
2. With the ‘Zeus’ project expanded: expand the ‘src’ directory, within the ‘src’ directory expand the ‘edu.sru.thangiah.zeus.vrptw’ package, and within that package double-click on ‘VRPTWRoot.java’ to open the source code
3. With the ‘VRPTWRoot.java’ source code open in the editor navigate to line 52 ‘new VRPTW(VRPTWProblemInfo.fileName, VRPTWProblemInfo.heuristicType);’



1. Comment out this line of code and uncomment lines 53 - ‘File[] VRPTWDataFiles = new File(VRPTWProblemInfo.inputPath).listFiles();’ and 55 - ‘recursivelyProcessAllVRPTWDataFiles(VRPTWDataFiles, VRPTWProblemInfo.heuristicType);’
2. By completing step 4, Zeus will be enabled to process multiple files in sequence, it is recommended that the GUI be disabled for this process.



1. On line 30 - 31 - ‘VRPTWProblemInfo.inputPath = VRPTWProblemInfo.workingDirectory+ “\\data\\vrptw\\ExcelDataFiles\\<root directory>\\";’ set the root directory of this process
2. After selecting the root directory, Zeus may be run for the VRPTW by either clicking on the ‘Run’ button or by pressing ‘Ctrl+F11’
3. **Developing Heuristics for the VRPTW in Zeus**

* **The Two Main Functions for Heuristics -** Upon completion of this section the user will understand the two methods utilized in heuristic development
  + Selection
    - The selection function declaration, getSelectShipment(), is found within the ShipmentLinkedList Class which is the parent class of VRPTWShipmentLinkedList. This method is a stub which is meant to be implemented by the inheriting class of VRPTWShipmentLinkedList, e.g. TimeOrientedNearestNeighbor
    - This method is responsible for finding the next shipment that is to be inserted into the emerging routes
  + Insertion
    - The insertion function declaration, getInsertShipment(), is found within the NodesLinkedList Class which is the parent class of VRPTWNodesLinkedList. This method is a stub which is meant to be implemented by the inheriting class of VRPTWNodesLinkedList, e.g. InsertionI
    - This method is responsible for inserting the shipment that was selected by inserting the shipment into a VRPTWNodes instantiation and inserting this VRPTWNodes instantiation into the emerging route
* **Developing a selection method for the VRPTW -** Upon completion of this section the user will be able develop their own selection method for the VRPTW

1. With the Eclipse IDE launched with the Zeus Workspace expand the Zeus project in the Package Explorer
   1. Note: if the Package Explorer is not in view on the workspace click ‘Window’ on the navigation bar, in the drop-down menu hover on ‘Show View’, in this menu click on ‘Package Explorer’ or press ‘Alt+Shift+Q, and P’ to open the view
2. With the ‘Zeus’ project expanded: expand the ‘src’ directory, within the ‘src’ directory expand the ‘edu.sru.thangiah.zeus.vrptw’ package, and within that package double-click on ‘VRPTWShipmentLinkedList.java’ to open the source code
3. Within this Java file create a new class named for the insertion method similar to the example below:

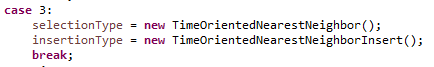


1. Inside of the class that was just created you must implement the following methods: getSelectShipment(VRPTWDepotLinkedList currDepotLL, VRPTWDepot currDepot, VRPTWShipmentLinkedList currShipLL, VRPTWShipment currShip) and toString() similar to the way they are below:





1. In the getSelectShipment() method the logic will be developed out that selects the next shipment. This method will then return this shipment as a VRPTWShipment
2. In the toString() method one will want to return a String value that lets the user know what this selection method is called, e.g. “Selection Type: Time Oriented Nearest Neighbor”
3. Upon completion of these methods this selection method may be utilized in the VRPTW by adding this to the switch statement in “VRPTW.java” similar to how it is below:



1. Setting ‘VRPTWProblemInfo.heuristicType’ equal to the case value given to the newly developed heuristic will allow for this to be run at the next run of Zeus
2. Zeus may be run for the VRPTW for the updated heuristic by either clicking on the ‘Run’ button or by pressing ‘Ctrl+F11’

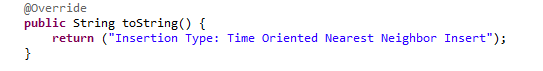
* **Developing a insertion method for the VRPTW -** Upon completion of this section the user will be able develop their own insertion method for the VRPTW

1. With the Eclipse IDE launched with the Zeus Workspace expand the Zeus project in the Package Explorer
   1. Note: if the Package Explorer is not in view on the workspace click ‘Window’ on the navigation bar, in the drop-down menu hover on ‘Show View’, in this menu click on ‘Package Explorer’ or press ‘Alt+Shift+Q, and P’ to open the view
2. With the ‘Zeus’ project expanded: expand the ‘src’ directory, within the ‘src’ directory expand the ‘edu.sru.thangiah.zeus.vrptw’ package, and within that package double-click on ‘VRPTWNodesLinkedList.java’ to open the source code
3. Within this Java file create a new class named for the insertion method similar to the example below:

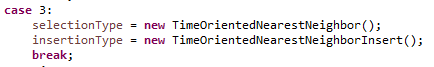


1. Inside of the class that was just created you must implement the following methods: getInsertShipment(VRPTWNodesLinkedList currNodeLL, VRPTWShipment theShipment) and toString() similar to the way they are below:





1. In the getInsertShipment() method the logic will be developed out that inserts the selected shipment as a node into the emerging route. This method will then return a boolean value indicating if the node was successfully inserted into the route
2. In the toString() method one will want to return a String value that lets the user know what this selection method is called, e.g. “Insertion Type: Time Oriented Nearest Neighbor Insert”
3. Upon completion of these methods this insertion method may be utilized in the VRPTW by adding this to the switch statement in “VRPTW.java” similar to how it is below:



1. Setting ‘VRPTWProblemInfo.heuristicType’ equal to the case value given to the newly developed heuristic will allow for this to be run at the next run of Zeus
2. Zeus may be run for the VRPTW for the updated heuristic by either clicking on the ‘Run’ button or by pressing ‘Ctrl+F11’