CST8130: Data Structures

Assignment #1- Router Simulator

Test Plan

Mark Kaganovsky

040-789-903

Section #302 (8:00am to 10:00am Monday)

This test plan is in the same format used by Professor Stanley Pieda in our semester 2 course: CST8132 Object Oriented Programming

Main

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Functionality Being Tested** | **What are we testing?** | **Pre-Condition**  **Any special context or program state to take note of?** | **Test method**  **How will we test?** | **Post-Condition**  **Expected Results** | **Post-Condition**  **Actual Results** |
| void main() – opening the file | Does it prompt the user for the name of the packet file after “booting” up the router (instantiating it), open it, and then handle any errors. | A Router object has been created, and the user entered a valid router table size. | Run the program, enter a valid number greater than 0 to create the routing table.  Enter the name of a valid file, and the name of an invalid file. | If the file is valid, the program will continue executing.  If the file is invalid, the program will display an error message describing why the file could not be opened (does not exist, do not have read permissions, etc.). | Matches |
| void main() – Reading the file, and creating the appropriate packet. | Does it successfully read the first character of a line, determine the packet type, and create the corresponding packet (RoutingPacket or DataPacket) based on that character, dropping the packet if it is invalid? | A Router object has been created, and the user entered a valid router table size.  The user has entered the name of a valid file, and that file was successfully opened. | Create a file in the following format:  “  z x x x x n x x x x n y  z x x x x n x x x x n y  z x x x x n x x x x n y  ”  Every line represents a packet.  Where valid values for ‘z’ is either a ‘p’ or a ‘d’.  Where ‘x’ represents some integer number (they don’t have to be the same), valid values are 0 to 255.  Where n is the network mask for an ip represented by the previous 4 integers; valid values are 8, 16, or 24.  Run the program; enter a valid number greater than 0 to create the routing table.  Enter the name of a valid readable file.  Step through with debugger; see if invalid packets are dropped, and if the appropriate Packet subclass (RoutingPacket or DataPacket) objects are created (and dropped if the destination ip or source ip were invalid). | For a file containing the following:  p 1 1 1 1 8 1 1 1 2 16 e0  p 2 2 2 2 16 2 2 2 3 24 e1  d 2 2 2 2 16 1 1 1 1 8 hello  z 1 1 1 1 24 2 2 2 2 16 fail  d 1 1 1 257 3 2 2 2 2 133 fail  Packets 1 and 2 should cause a RoutingPacket object to be created.  Packet 3 should cause a data packet to be created.  Packet 4 should be dropped because of its invalid packet type.  Packet 5 should be dropped because packet.readPacket(Scanner) will return false (meaning an invalid packet was read. | Matches |

Router

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Functionality Being Tested** | **What are we testing?** | **Pre-Condition**  **Any special context or program state to take note of?** | **Test method.**  **How will we test?** | **Post-Condition**  **Expected Results** | **Post-Condition**  **Actual Results** |
| Router() | Does it ask the user for the max router size, and successfully handle invalid input? | Main.main() has started, it is trying to create a new instance of the Router class. | Start the program.  Input numbers greater than 0, but less than the maximum value of an int.  Input numbers less than or equal to zero, and greater than the maximum value of an int.  Input non-numeric characters. | If the user inputs between 0 and the max value of an int, input should be accepted, and a new array of RoutingTableEntrys should be created based on that number.  If the user nputs an out of range value, ask them to try again.  If the user inputs any non-numeric characters, ask them to try again. | Matches. |
| void processPackets(  Packet packet) | Does it process the packet properly?  Add to routing table if it’s a RoutingPacket and if it does not already exist in the table.  Send the data out of the correct port if the entry already exists. | main() has read the first character of the line (one packet per line), determined the packet type, created the appropriate packet, and has just sent it to this method. | Start the program in debug mode and make it read from a file containing valid packets. See if it does the correct action for:  RoutingPacket – added to routing table if the entry does not already exist.  DataPacket – Sent out of the correct port if the destination IP is in the routing table. | For a file containing the following packets:  p 192 168 1 2 24 192 168 1 4 24 e0  p 192 168 1 2 24 192 168 1 4 24 e0  d 192 168 1 2 24 192 168 1 4 24 123456778123  d 192 168 1 4 24 192 168 2 3 24 packet1  Packet 1 should be added to the routing table.  Packet 2 should be dropped since it already exists in the table.  Packet 3 should be sent out port e0.  Packet 4 should be sent out port 0. | Matches. |
| void displayTable() | Does it output a neatly formatted routing table? | File has been fully read. main() has called this method to simulate the router being shut down. | Start the program, make it read from a packet file. | The table should have two columns. The first one contains the network IP for a destination, and the subnet mask. The second column should contain the port to get to that network.  Ex:  192.168.1.0 /24 e0  8.0.0.0 8 e1 | Matches. |

RoutingTableEntry

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Functionality Being Tested** | **What are we testing?** | **Pre-Condition**  **Any special context or program state to take note of?** | **Test method**  **How will we test?** | **Post-Condition**  **Expected Results** | **Post-Condition**  **Actual Results** |
| RoutingTableEntry() | Does it create a new instance of a destinationIP address and a portCode String. | None | Create an instance of this class. Step through with debugger, verify that the objects have been created. | An IPAddress object reference is given to destinationIP, and a new String is assigned to portCode. | Matches |
| void addEntry(IPAddress ip, String port) | Does it set this instances destinationIP and portCode variables to ip and port respectively. | A RoutingTableEntry object has been created. | Create an instance of RoutingTableEntry, an instance of an IPAddress object and a String instance. Call this method and step through with debugger to see if the reference match up. | destinationIP should equal the parameter ip.  portCode should equal the parameter port. | Matches |
| String searchForPort(IPAddress ip) | Does it test for equality between this entry’s destinationIP and the IPAddress passed to this method. | A RoutingTableEntry object has been created, and this object has been assigned a destination IPAddress and a portCode using addEntry() | Create an instance of RoutingTableEntry, call the addEntry() class and pass a valid IPAddress and portCode. Observe the return value. | If the IPAddress passed to this method is equal in value to the destinationIP, then the portCode should be returned, otherwise null should be returned. | Matches |

Packet

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Functionality Being Tested** | **What are we testing?** | **Pre-Condition**  **Any special context or program state to take note of?** | **Test method**  **How will we test?** | **Post-Condition**  **Expected Results** | **Post-Condition**  **Actual Results** |
| Packet() | Does it initialize the following fields:  destinationip, sourceip, packetData? | Main has determined the type of packet (RoutingPacket or DataPacket), and has created a new instance of either RoutingPacket or DataPacket. | Run the program and step through with debugger once reading of the packet file has started. | destinationIP and sourceIP should have a unique IPAddress object assigned to them. | Matches |
| boolean readPacket(Scanner) | Does it read in the address for the destinationIP and sourceIP, then set packetData to the rest of the line.  If the readAddress() method for destinationIP or sourceIP returns false, then this method needs to return false. | Main has created either a DataPacket or RoutingPacket, and has passed the fileReader Scanner to this method to fill in the packet information. | Run the program and step through with debugger once reading of the packet file has started. | If the destinationIP or sourceIP are invalid IPAddresses, then this method should return false. Otherwise it should return true. | Matches |
| IPAddress  getDestNetwork() | Does it return a reference to the network address of destinationIP? | A Packet object has been created. | Run the program and step through with debugger once reading of the packet file has started. Print out the returned value from this method. | The network portion of the destinationIP address should be returned.  Example:  If destinationIP holds the value: “192.168.1.2” with a subnet mask of /24, the IP returned from this method should be “192.168.1.0” | Matches |
| String getPacketData() | Does it return a reference to the data contained in this packet. | A Packet object has been created. | Print out the result returned from this method. | If the data in the packet was “e0”, then e0 should be printed. | Matches. |
| Abstract  boolean  processFoundPacket  (String) | Abstract method. Look at the subclasses DataPacket and RoutingPacket. | A Packet object has been created. The router has processed the packet and determined that it exists in the table. | See RoutingPacket and DataPacket. | See RoutingPacket and DataPacket. | See RoutingPacket and DataPacket. |
| Abstract  boolean  processNotFoungPacket (String) | Abstract method. Look at the subclasses DataPacket and RoutingPacket. | A Packet object has been created. The router has processed the packet and determined that it does not exist in the table. | See RoutingPacket and DataPacket. | See RoutingPacket and DataPacket. | See RoutingPacket and DataPacket. |

RoutingPacket

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Functionality Being Tested** | **What are we testing?** | **Pre-Condition**  **Any special context or program state to take note of?** | **Test method**  **How will we test?** | **Post-Condition**  **Expected Results** | **Post-Condition**  **Actual Results** |
|  |  |  |  |  |  |

DataPacket

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Functionality Being Tested** | **What are we testing?** | **Pre-Condition**  **Any special context or program state to take note of?** | **Test method**  **How will we test?** | **Post-Condition**  **Expected Results** | **Post-Condition**  **Actual Results** |
|  |  |  |  |  |  |

IPAddress

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Functionality Being Tested** | **What are we testing?** | **Pre-Condition**  **Any special context or program state to take note of?** | **Test method**  **How will we test?** | **Post-Condition**  **Expected Results** | **Post-Condition**  **Actual Results** |
|  |  |  |  |  |  |