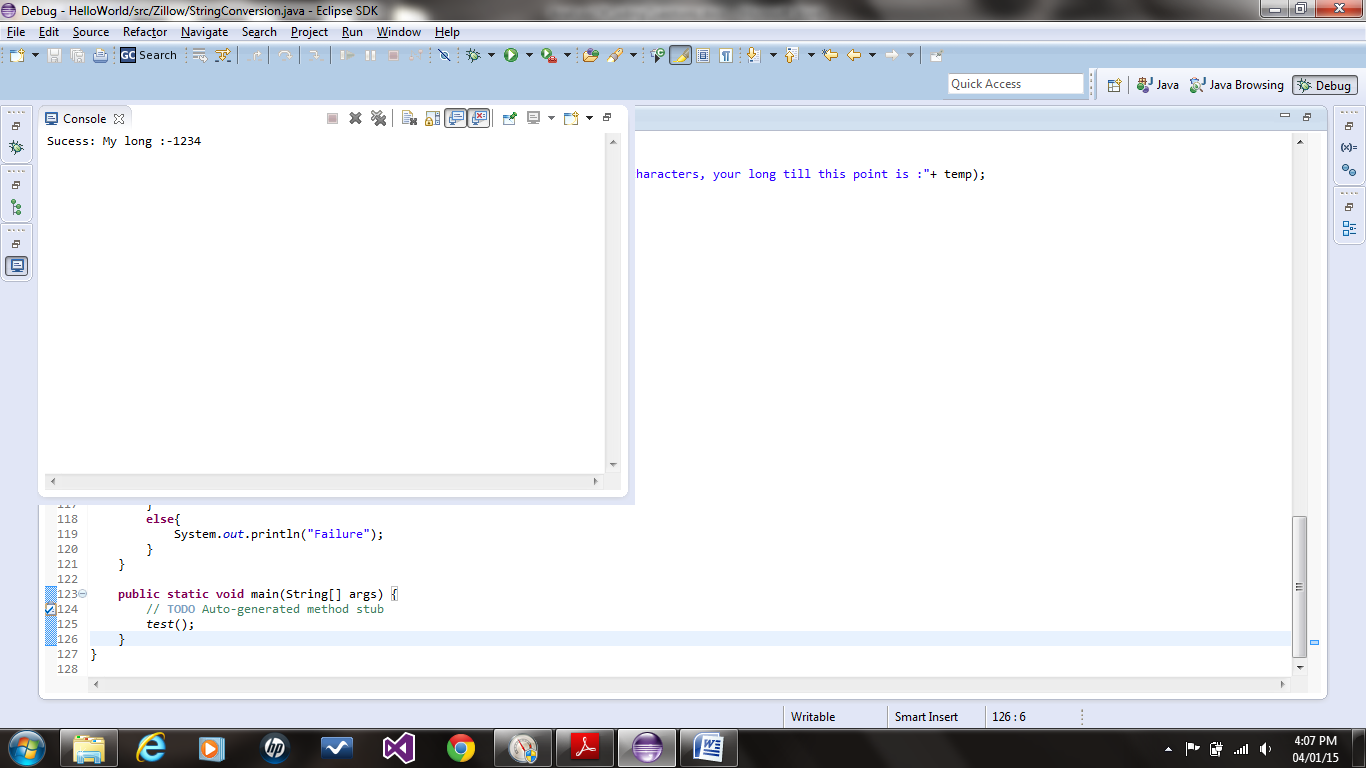
**Question 1)** Given a string, write a routine that converts the string to a long, without using the

built in functions that would do this. Describe what (if any) limitations the code has.

**Answer** – Code developed converts the string to long data type. Limitation of code is that whenever the string is not in proper format, then I throw exception displaying the reason for the error. For example,

If string is empty, string has no characters in it, or just a ‘+’ or ‘-‘ symbol, or string contains non numeric characters, or it’s not in range of long data type then I throw exception.

Below is the screenshot after running the program,



**Question 2)** Implement insert and delete in a tri-nary tree. A tri-nary tree is much like a binary

tree but with three child nodes for each parent instead of two -- with the left node being values

less than the parent, the right node values greater than the parent, and the middle nodes values

equal to the parent.

**Answer** – I have developed code which is independent of parent pointer, which helps in deleting the node from tree. So I am not storing the parent pointer in node as I am finding the parent at runtime. This increases the CPU cycles and overall complexity, but it provides a way to make code independent of whether parent pointer is present or not.

Also I have made my nodes responsible to manage the tree, instead of the calling function. Calling function will only be responsible to call the insert or delete at root node, and then rest of computation are done by my tree nodes. Code could have been much simpler if I allow my calling function to use recursion to find the node position for insert or delete. But instead I chose not to allow calling function to manipulate my tree nodes. As it provide much more abstraction from outer world. So my nodes are responsible for manipulating tree nodes for add or remove.

Screenshot for the test input is given below,

