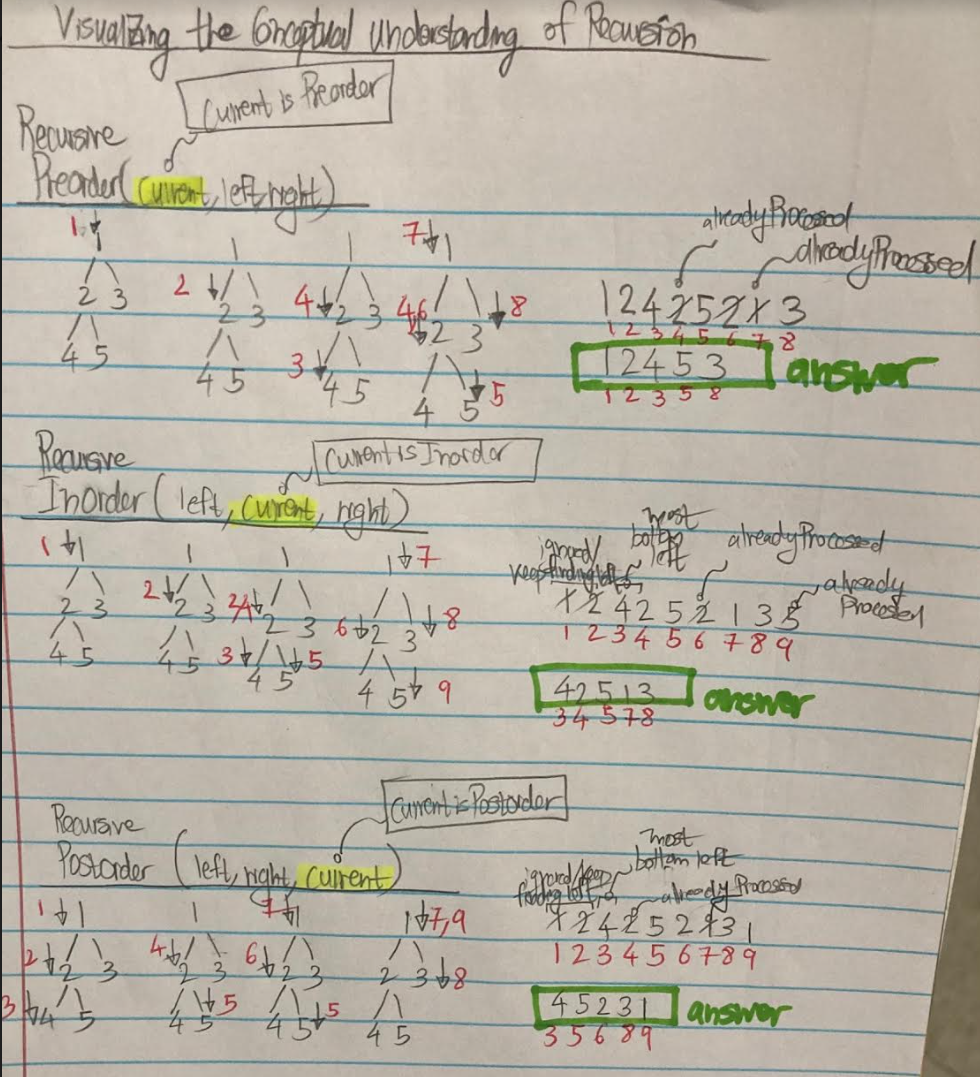
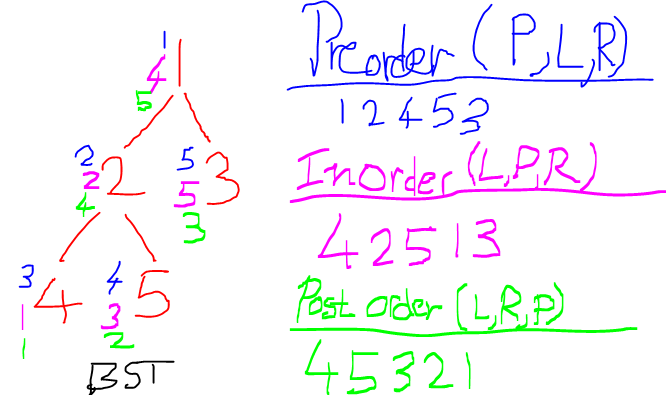
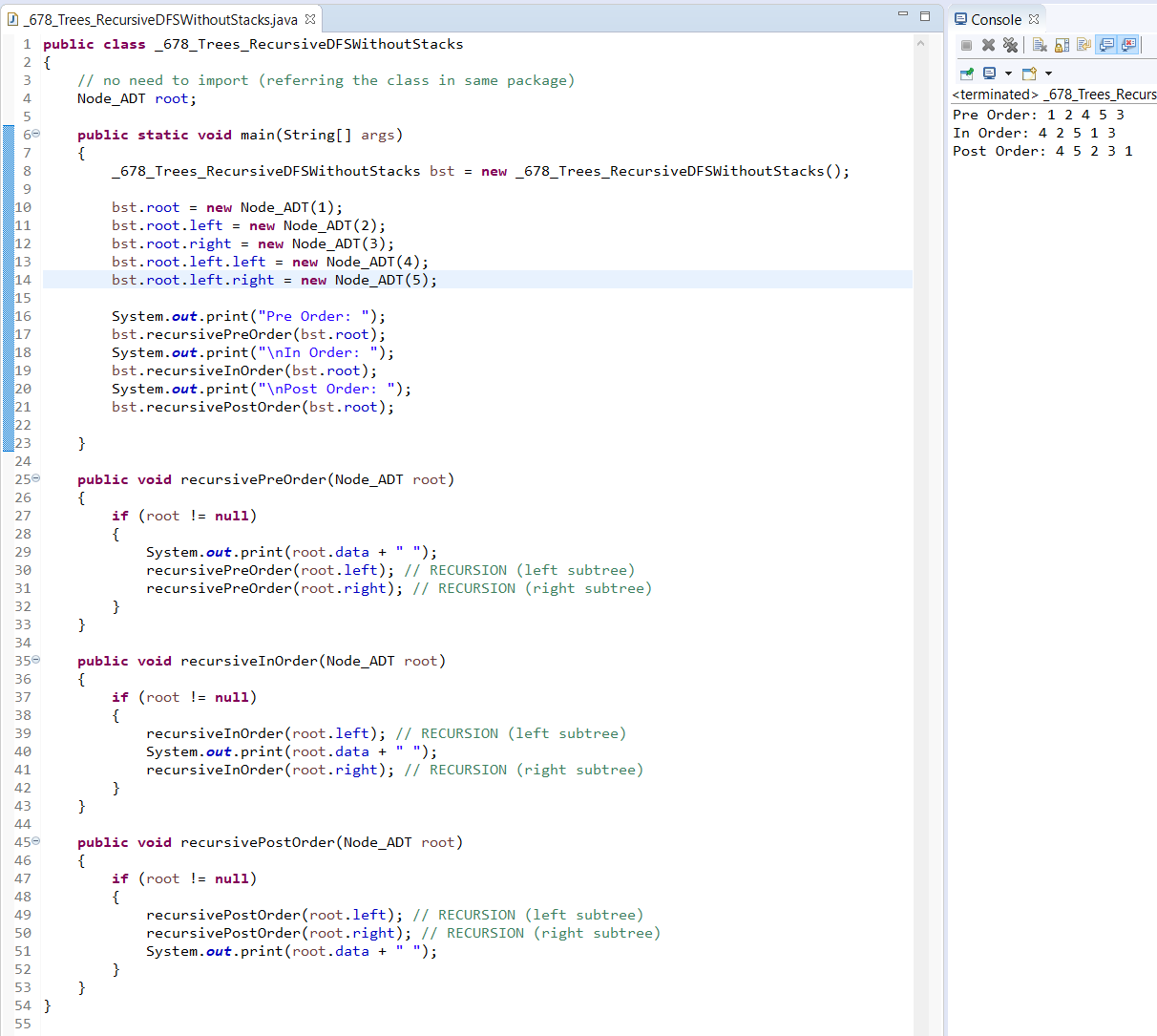
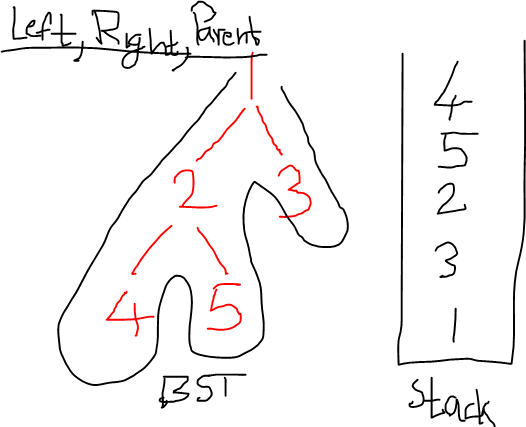
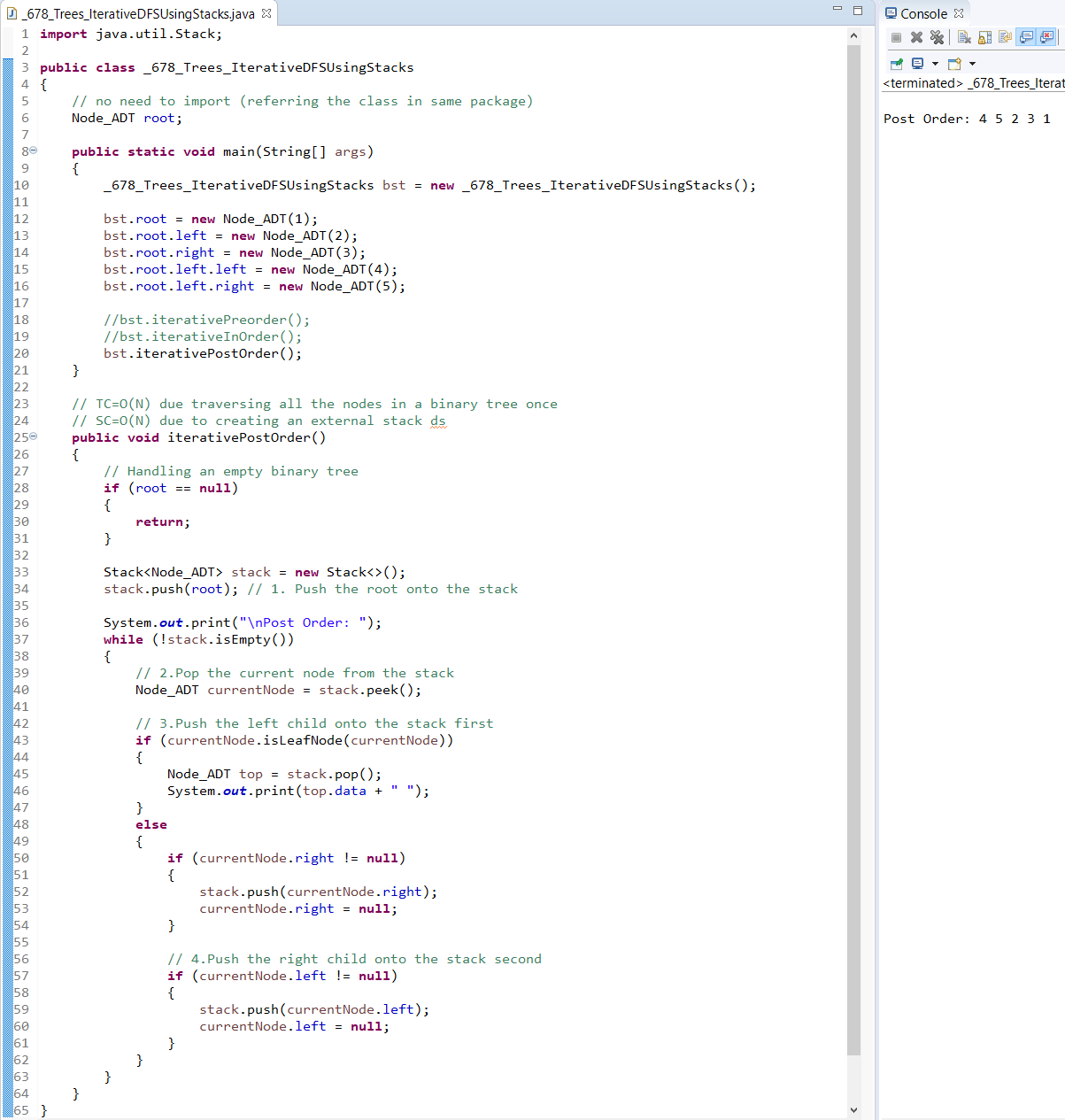
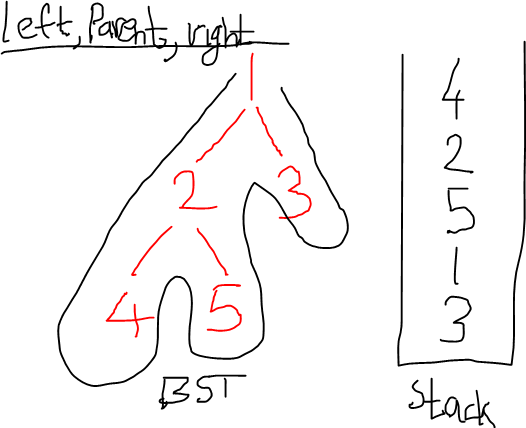
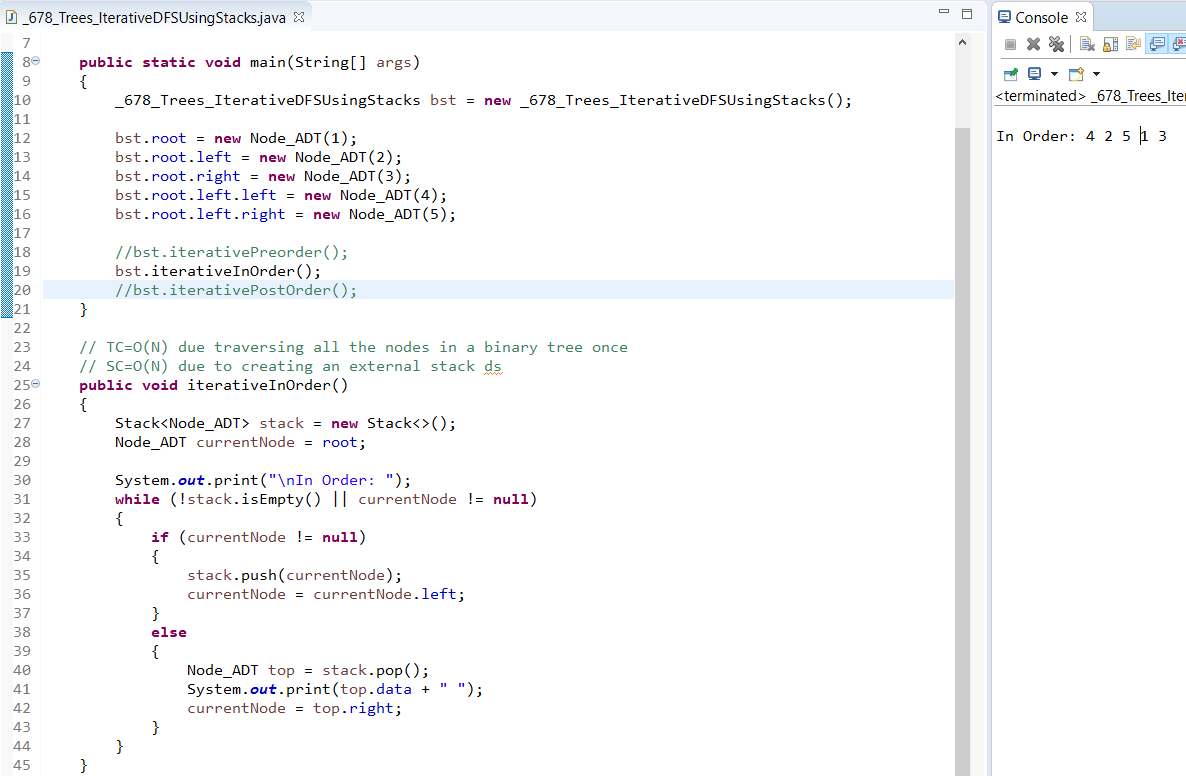
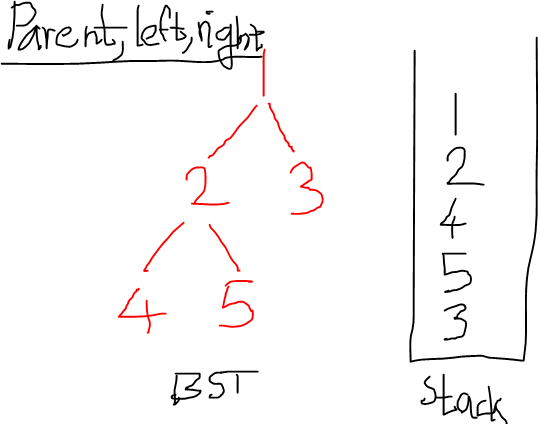
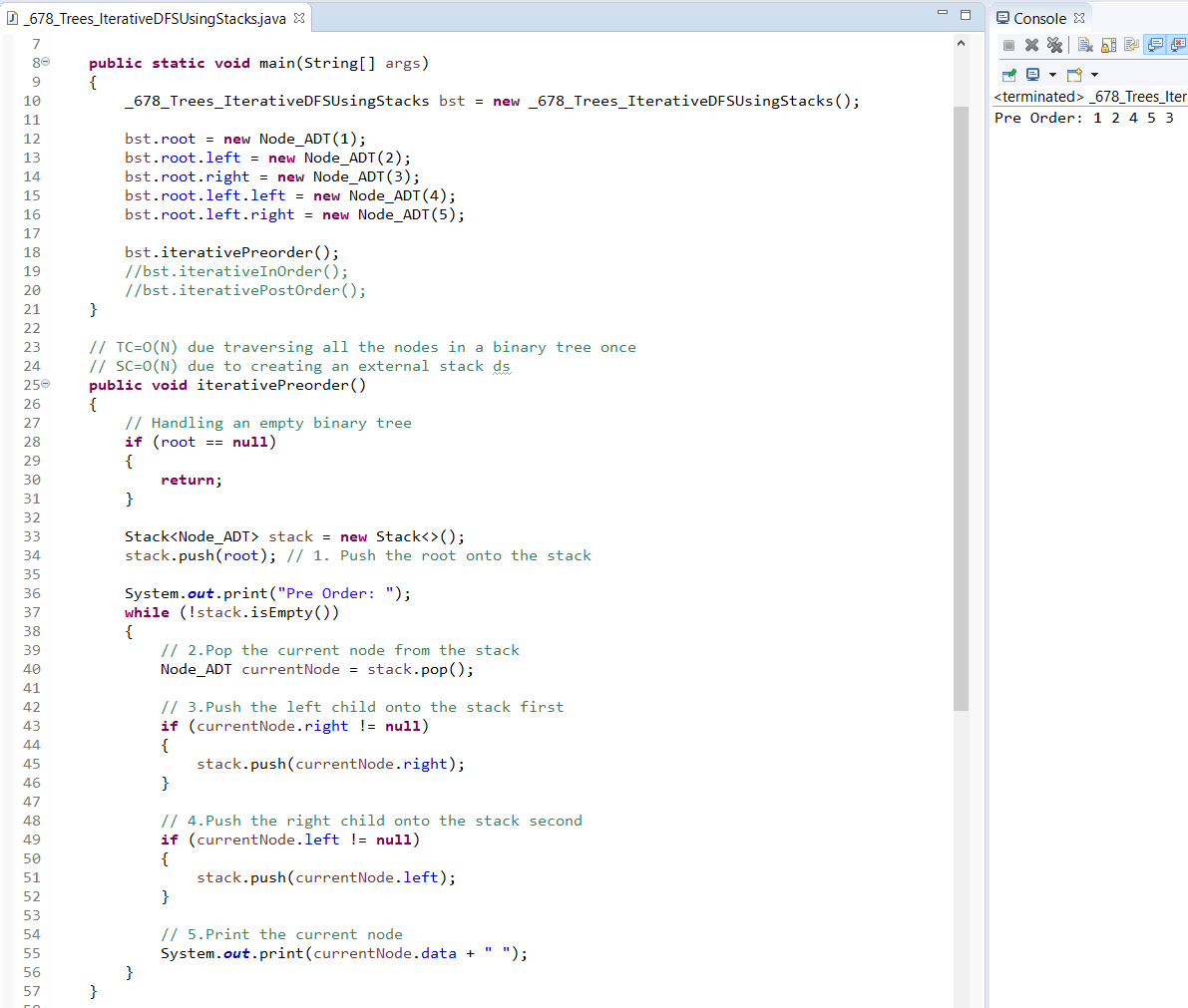
1st focus on developing a good conceptual understanding via visualization



2nd code the visualized conceptual understanding



Recursive functions require arguments (cannot use the instance variable named root here)



Iterative functions don’t require arguments (we can use the instance variable named root here within the body of the function)

Note: created a new variable called currentNode so that we can avoid changing the root

Note: the order of pushing into the stack is the opposite of the preorder traversal

Top pointer (this is where we do the push and pop operations on the stack)

Iterative functions don’t require arguments (we can use the instance variable named root here within the body of the function)

Note: created a new variable called currentNode so that we can avoid changing the root

Note: the order of pushing into the stack is the opposite of the inorder traversal

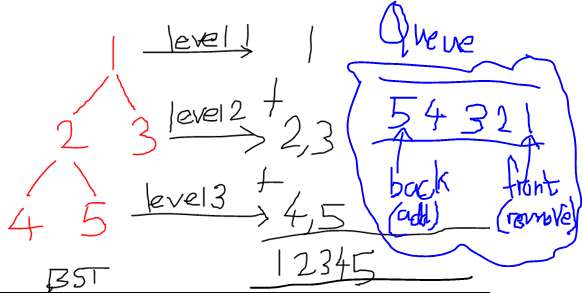
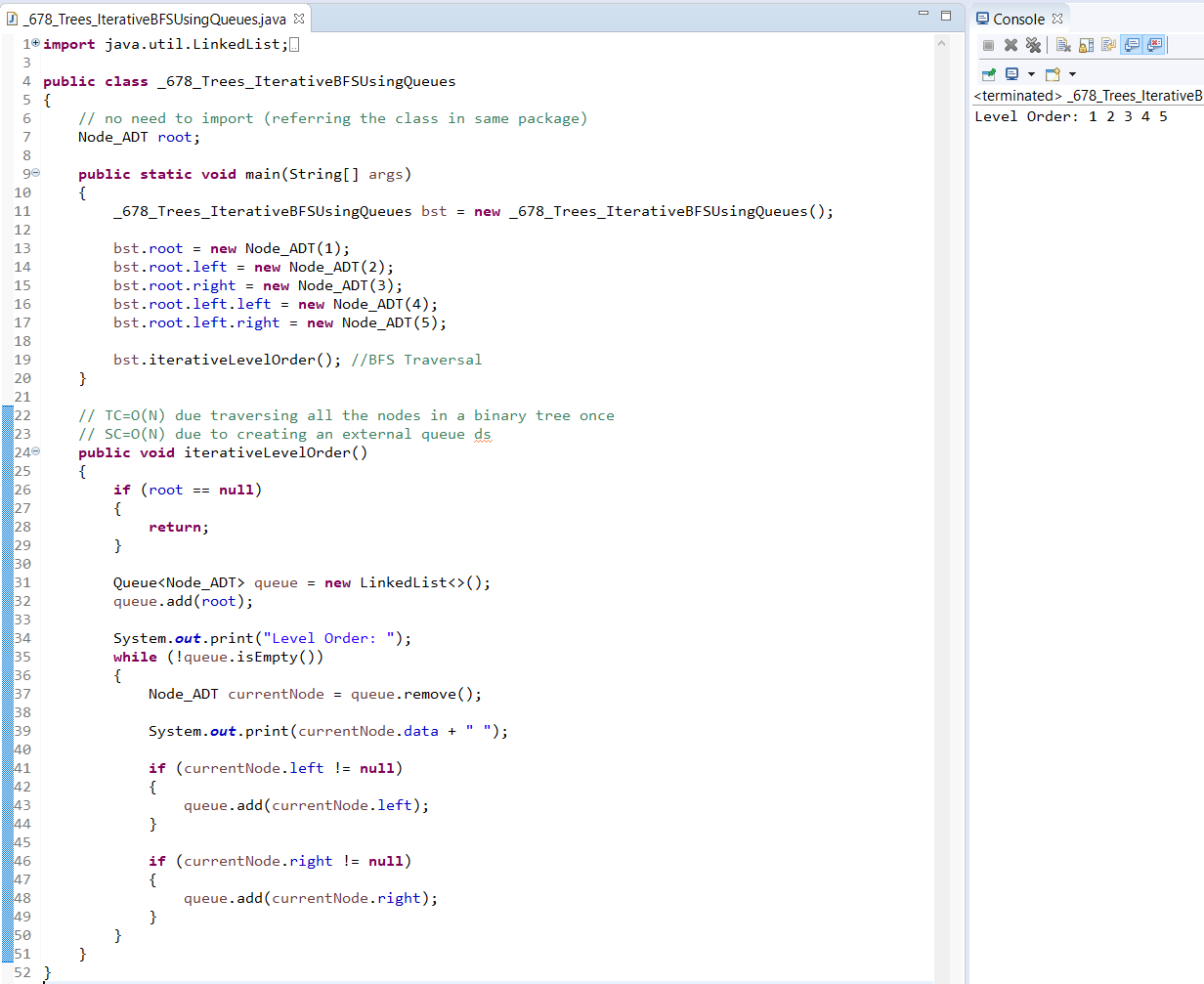
Top pointer (this is where we do the push and pop operations on the stack)

Iterative functions don’t require arguments (we can use the instance variable named root here within the body of the function)

Note: created a new variable called currentNode so that we can avoid changing the root

Note: the order of pushing into the stack is the opposite of the postorder traversal

Top pointer (this is where we do the push and pop operations on the stack)



Iterative functions don’t require arguments (we can use the instance variable named root here within the body of the function)

Note: created a new variable called currentNode so that we can avoid changing the root

Note: the order of adding into the queue is the opposite of the level order traversal