Chris Banci

ID: 010031304

3/19/16

Linear Linked Lists - HW # 2

2)

//Adds an object to the front of a list

Public void addFirst(object o) {

ObjectListNode p = new ObjectListNode(o, list);

If (list == null);

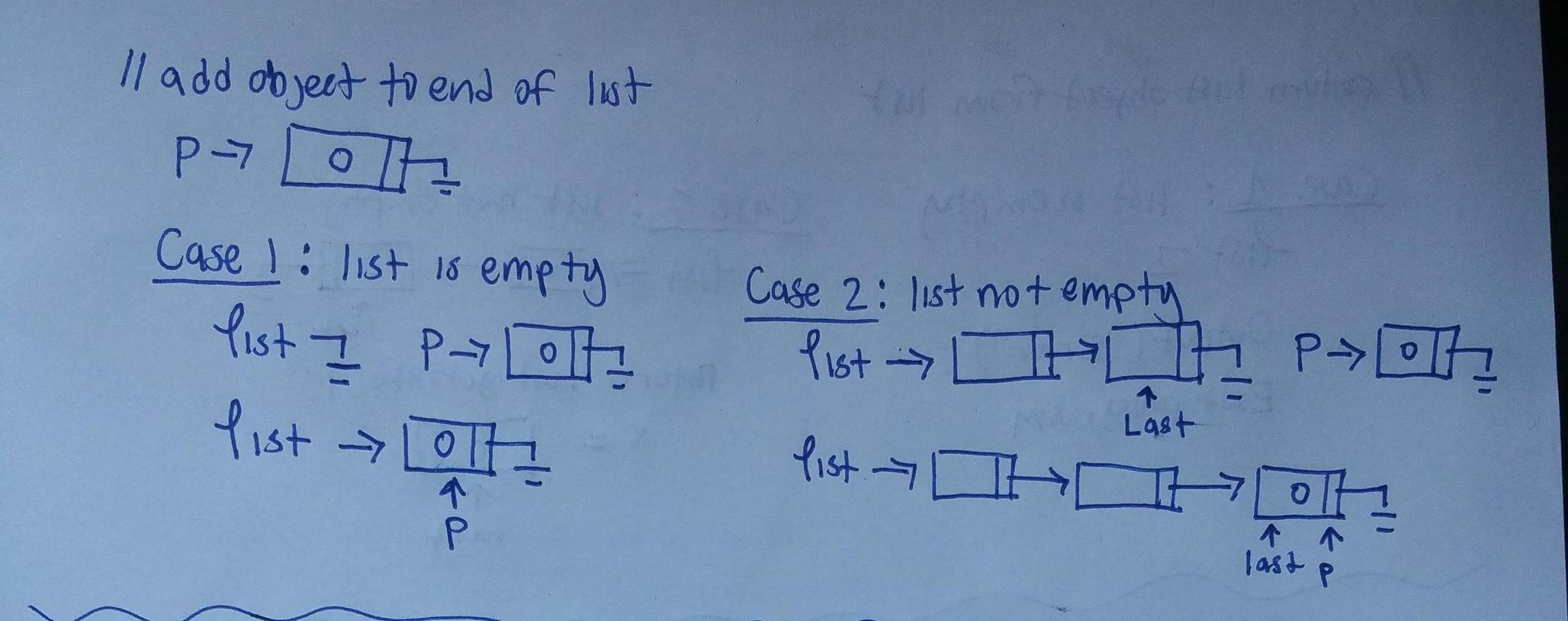
last n = p;

List = p;

----------------------------------------------------------

**Case 1:** if the list is empty, list points to the new object node.

**Case 2**: if list is not empty, last node in the lists sets next to the new object node.



// Adds a node to the front of the list

public void addFirst(ObjectListNode p)

if (p == null) {  
 System.out.println(“RunTime Error: addFirst()”);

System.exit(1)

}

p.setNext(list);

If (list == null)

last = p;

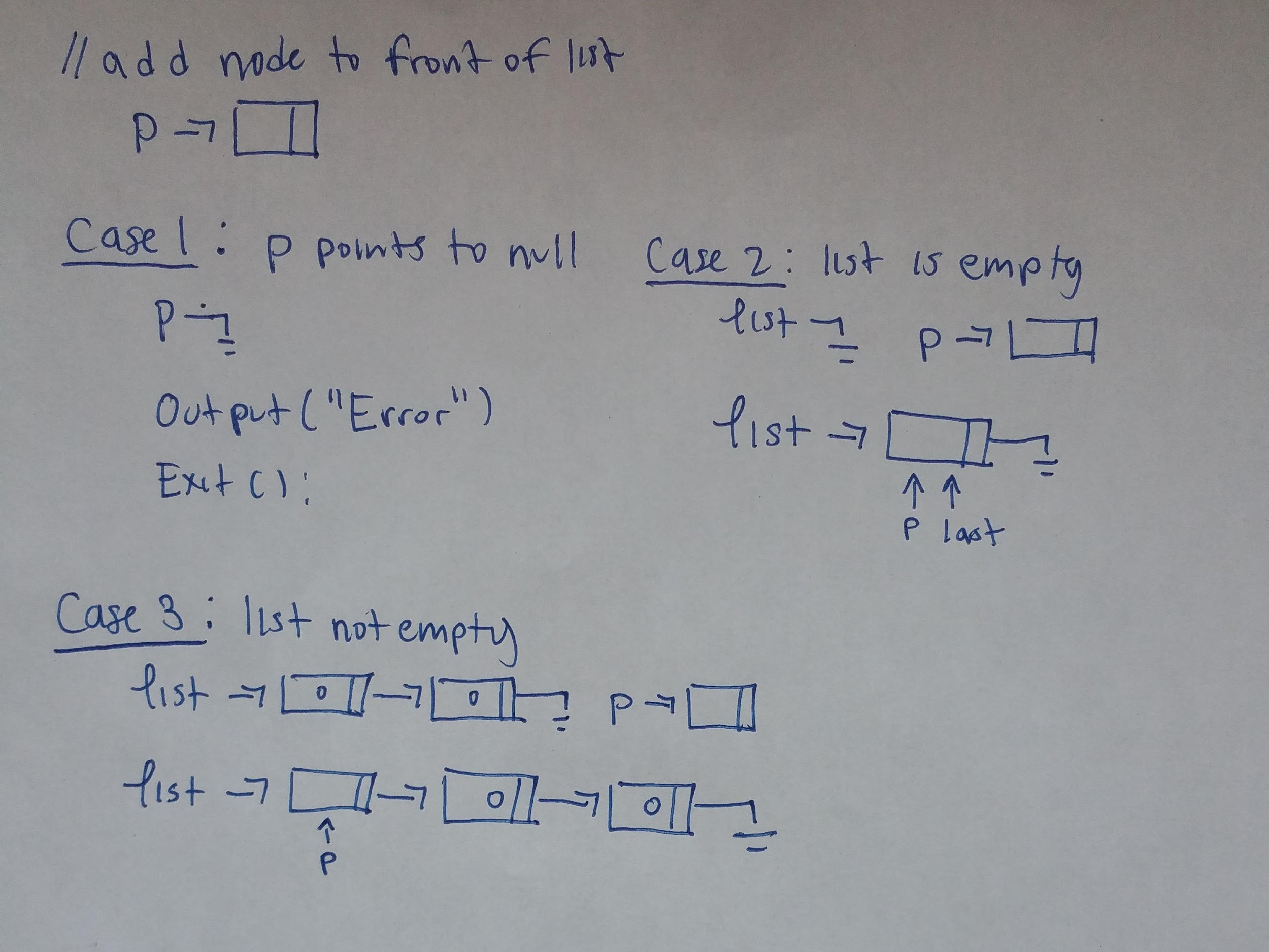
list = p;

}

--------------------------------------------------------------------------

**Case 1**: p pointing to null is a special case. If you were to add p to the front of the list, the other nodes in the list would be lost.  
**Case 2**: if the list is empty, list points to new node.

**Case 3**: if the list is not empty, new node p is set next to node pointed by list, then list points to p.



//Returns the last object in the list

public Object getLast() {

If (list == null) {  
 System.out.println(“RunTime Error: getLast()”);

}

return last.getInfo();

}

-------------------------------------------------------------------------------

**Case 1**: The list being empty is a special case. If the list is empty, there isn't an object that can be returned.

**Case 2:** if the list is empty, the last node pointed by last is returned.

