# Short Report

## Implementation

To Implement my program, I used four classes, one for the trucks, one for the boxes, one for the algorithms & lastly one for testing. To store the trucks & boxes I used Array lists so that I could access the objects within loops, which would be used for the likes of first fit as I needed some method of storing the first box on the stack for each truck.

## Pseudo Code

### NEXT FIT

1. For all boxes i = 1, 2, …, n

2. If box fits on the current stack

3. Stack box on truck

4. End If

5. Else

5. The new width of the truck equals the current width minus the width of the box on the bottom of the current stack

6. If box fits within this new width

7. Start new stack on the truck

8. End If

9. Else

10. Create a new truck & fit the box on that truck

11. End else

12. End else

13. End for

14 Print number of trucks used

15. End method

### FIRST FIT

1. For all boxes i = 1, 2, …, n

2. For all trucks j = 1, 2, …, n

3. If box(i) fits in truck (j) then

4. If this is the first box on the stack then

5. Store this element within the current truck

6.End If

7: Stack box(i) on truck(j)

8. Break loop

9.End If

10.End for

11. If box(i) did not fit in any truck

10. The new width of the truck equals the current width minus the width of the box on the bottom of the current stack

11. If box fits within this new width

12. Start new stack on the truck

13. End If

14. Else

15. Create a new truck & fit the box on that truck

16. End else

17. End If

18. End for

19. Print trucks used

20.End method

## Testing

To test the program, I used a test class that would use a while loop to create a set amount of box objects, the length & width of these boxes are randomly generated within the loop between 1 – 60, 60 being the size of the truck so we would not get boxes bigger than our truck. We would only have one truck to begin with as all trucks will be the same size we simply just generate more when needed within the algorithm. I then simply called the firstFit & nextFit method, using system.nanoTime() to calculate how long each method would take to complete. Lastly my other test would mostly be Implemented within the algorithm class Itself which would be to count the number of trucks used, this is simply printed at the very end of the method so that when we call the method we get an output of the trucks used.

Testing for correctness I would change the number of boxes I generated to a lower number such as 10 then simply use a notepad to see If the correct amount of trucks were being used.

### Testing Results

Truck Size: 60 length 60 width

Number of boxes: 50

Note: The length & Width of all boxes are randomly between 1 - 60

Test 1:

Next Fit Trucks: 25

First Fit Trucks: 13

Next Fit Duration: 77924 Nanoseconds

First Fit Duration: 218188 Nanoseconds

Test 2:

Next Fit Trucks: 26

First Fit Trucks: 12

Next Fit Duration: 106028 Nanoseconds

First Fit Duration: 313997 Nanoseconds

Test 3:

Next Fit Trucks: 30

First Fit Trucks: 14

Next Fit Duration: 83290 Nanoseconds

First Fit Duration: 209501 Nanoseconds

Test 4:

Next Fit Trucks: 30

First Fit Trucks: 18

Next Fit Duration: 78946 Nanoseconds

First Fit Duration: 224319 Nanoseconds

Test 5:

Next Fit Trucks: 28

First Fit Trucks: 17

Next Fit Duration: 111649 Nanoseconds

First Fit Duration: 197494 Nanoseconds

Test 6

Next Fit Trucks: 27

First Fit Trucks: 16

Next Fit Duration: 102707 Nanoseconds

First Fit Duration: 236838 Nanoseconds

### Testing Graphs

## Comments on tests

From the five tests, we did we can see that first fit requires less trucks then next fit, on average next fit required 27.6 trucks while next fit required 15. However, from the tests conducted we can see that first fit also requires more time to complete then next fit, with next fit taking an average of 93424 nanoseconds & first fit taking 233389 nanoseconds. This suggests that first fit takes slightly over twice as long to complete then next fit. As such there seems to be a trade-off between getting the best performance & using the least amount of trucks.