Lab 5 External Documentation   
CPSC 1150 – W02

**Program** Find a person’s middle name

**File Name:** FindMiddleName.java

**Purpose:** Find the middle name from full name.

**Input:** String – full name

**Output:** String – middle name

**Algorithm:**

BEGIN

1. Input fullname.
2. Set startIndexMiddleName as first space index + 1 of fullName.
3. Set endIndexMiddleNAme as last space index of fullName.
4. Set middleName as substring of fullName from startIndexMiddleName and endIndexMiddleName
5. Print middleName.

END.

**Summary:** The middle name is from first to last index of space in full name.

**Sample input and output:**

Enter your name: Cuong Ngoc Hoang Dinh

Your middle name is Ngoc Hoang

Enter your name: Jorge Mario Bergoglio

Your middle name is Mario

**Program** Conversions between kilograms and pounds

**File Name:** KilogramsAndPounds.java

**Purpose:** Print the table which convert kg to lb and lb to kg

**Input:** No input

**Output:** Table convert kg to lb and lb to kg

**Algorithm:**

BEGIN

1. Print the header and line.
2. Init row as 0;
3. While row is less than 10:
   1. Print 1+row\*2 for kg and use convertKgtoLb(1+row\*2) to print the lb; print 20+row\*5 for lb and convertLbtoKg(20+row\*5) to print the kg.
   2. Add row to 1

END.

float convertKgtoLb(int kilograms) BEGIN

1. Return kilograms \* 2.2f

END.

loat convertLbtoKg(int pounds) BEGIN

1. Return pounds / 2.2f

END.

**Summary:** For each row from 0 to 9, the main kg or lb is base+row\*step. For example, the first row of kilogram is 1+0\*2=1, or the last row of lb is 20+9\*5=65.

**Sample input and output:**

kilograms pounds | pounds kilograms

---------------------|---------------------

1 2.200 | 20 9.091

3 6.600 | 25 11.364

5 11.000 | 30 13.636

7 15.400 | 35 15.909

9 19.800 | 40 18.182

11 24.200 | 45 20.455

13 28.600 | 50 22.727

15 33.000 | 55 25.000

17 37.400 | 60 27.273

19 41.800 | 65 29.545

**Program** Display Pyramid

**File Name:** DisplayPyramid.java

**Purpose:** Input the height of the pyramid, output the pyramid.

**Input:** Integer – height of pyramid (from 1-15)

**Output:** Pyramid

**Algorithm:**

BEGIN

1. Input height.
2. For each level of the pyramid from 1 to height:
   1. Create the new string line.
   2. For each number from height to 1 (left-hand side of the pyramid), line=addNumber(line, number, level)
   3. For each number from 2 to height (right-hand side of the pyramid), line=addNumber(line, number, level)
   4. Print the line.

END.

String addNumber(String line, int number, int level) BEGIN

1. If the number is greater than level, then add spaces to line.
2. Else if the number is greater or equal 10, then add number to line; else add a space then add number to line.
3. Add a space between numbers.
4. Return line

END.

**Summary:** Go through each level of the height from 1 to height, create an empty string first. For the left-hand of the line, go through number from height to 1, if number is greater then level then add space, else add number. For the right-hand side of the line from 2 to height, do the same as left-hand side.

**Sample input and output:**

Enter height of the pyramid (1-15): 12

1

2 1 2

3 2 1 2 3

4 3 2 1 2 3 4

5 4 3 2 1 2 3 4 5

6 5 4 3 2 1 2 3 4 5 6

7 6 5 4 3 2 1 2 3 4 5 6 7

8 7 6 5 4 3 2 1 2 3 4 5 6 7 8

9 8 7 6 5 4 3 2 1 2 3 4 5 6 7 8 9

10 9 8 7 6 5 4 3 2 1 2 3 4 5 6 7 8 9 10

11 10 9 8 7 6 5 4 3 2 1 2 3 4 5 6 7 8 9 10 11

12 11 10 9 8 7 6 5 4 3 2 1 2 3 4 5 6 7 8 9 10 11 12

Enter height of the pyramid (1-15): 5

1

2 1 2

3 2 1 2 3

4 3 2 1 2 3 4

5 4 3 2 1 2 3 4 5

**Program** Simulate the movement of an elevator

**File Name:** Elevator.java

**Purpose:** Simulate the movement of an elevator

**Input:** Integer – the floor you want to go

**Output:** Simulate the movement of an elevator

**Algorithm:**

BEGIN

1. Set the currentFloor=1.
2. Print the elevator at currentFloor.
3. While true
   1. Input the targetFloor.
   2. Check if the targetFloor is valid, if not then print “Invalid floor” and redo the loop.
   3. If the targetFloor==0 then exit program.
   4. If the targetFloor > currentFloor, then make a loop position=targetFloor until position==currentFloor, elevator(position, true)
   5. Else make a loop position= currentFloor until position== targetFloor, elevator(position, false)
   6. Set the currentFloor = targetFloor.

END.

elevator(int floor) BEGIN

1. Print the floor with out moving sign.

END.

elevator(int floor, boolean up) BEGIN

1. If up then print the floor with up sign.
2. Else print the floor with down sign.

END.

**Summary:** Start at floor 1, ask for target floor. If target floor is 0 then exit program. If target floor is greater then current floor then loop from target to current floor (to move up) and print the elevator, else loop from current floor to target floor(to move down) and print the elevator.

**Sample input and output:**

o---o

| 1 |

o---o

You are on floor 1

Enter floor (1-9): 2

o---o /\

| 2 | / \

o---o / \

o---o /\

| 1 | / \

o---o / \

You are on floor 2

Enter floor (1-9): 4

o---o /\

| 4 | / \

o---o / \

o---o /\

| 3 | / \

o---o / \

o---o /\

| 2 | / \

o---o / \

You are on floor 4

Enter floor (1-9): 1

o---o \ /

| 4 | \ /

o---o \/

o---o \ /

| 3 | \ /

o---o \/

o---o \ /

| 2 | \ /

o---o \/

o---o \ /

| 1 | \ /

o---o \/

You are on floor 1