

Cyber Security Department

Faculty of computer and data science

Alexandria University

Final DATA STRUCTURES PROJECT

Phone Book Management Application

Team members:

Abdelrahman Usama Raslan: 20221460102

Rewan Salah Shiha: 20221447143

Aml Ibrahim Mohamed: 20221443988

Suhaila Adel Ali: 20221376543

20 22



Phone Book Management Application

Description:

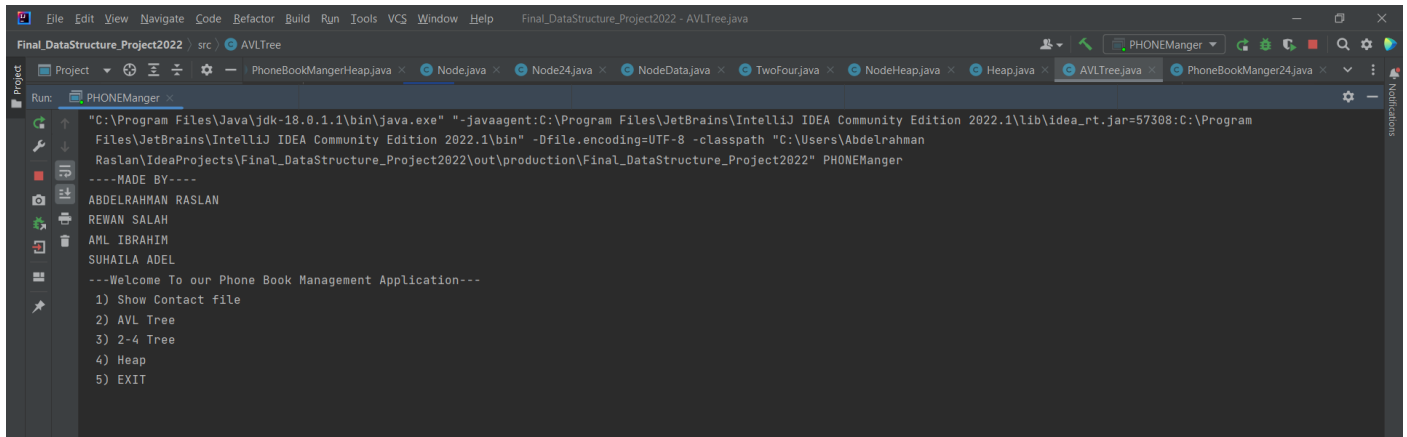
- Phone Book Management Application which encompasses searching, inserting and deleting operations.
- The application using the following types of binary search trees: AVL trees, 2-4 trees and heap
- Sorting on all trees by inorder traversal
- Use csv file or manually input to start previous functions
- The user chooses between the three methods of trees (AVL, 2-4, Heap)
- The user also chooses the method of search, insertion and deletion between the first name, the second name and the full name
- The program accepts files containing 4 to 5 columns of data with 1 to 10,000 or more rows
- Application with simple start menu for each tree

Classes of java App:

1. PHONEManger
2. PhoneBookMangerAVL
3. PhoneBookManger24
4. PhoneBookMangerHeap
5. Node
6. Node24
7. NodeData
8. AVLTree
9. TwoFour
10. NodeHeap
11. Heap

Samples of program Output:

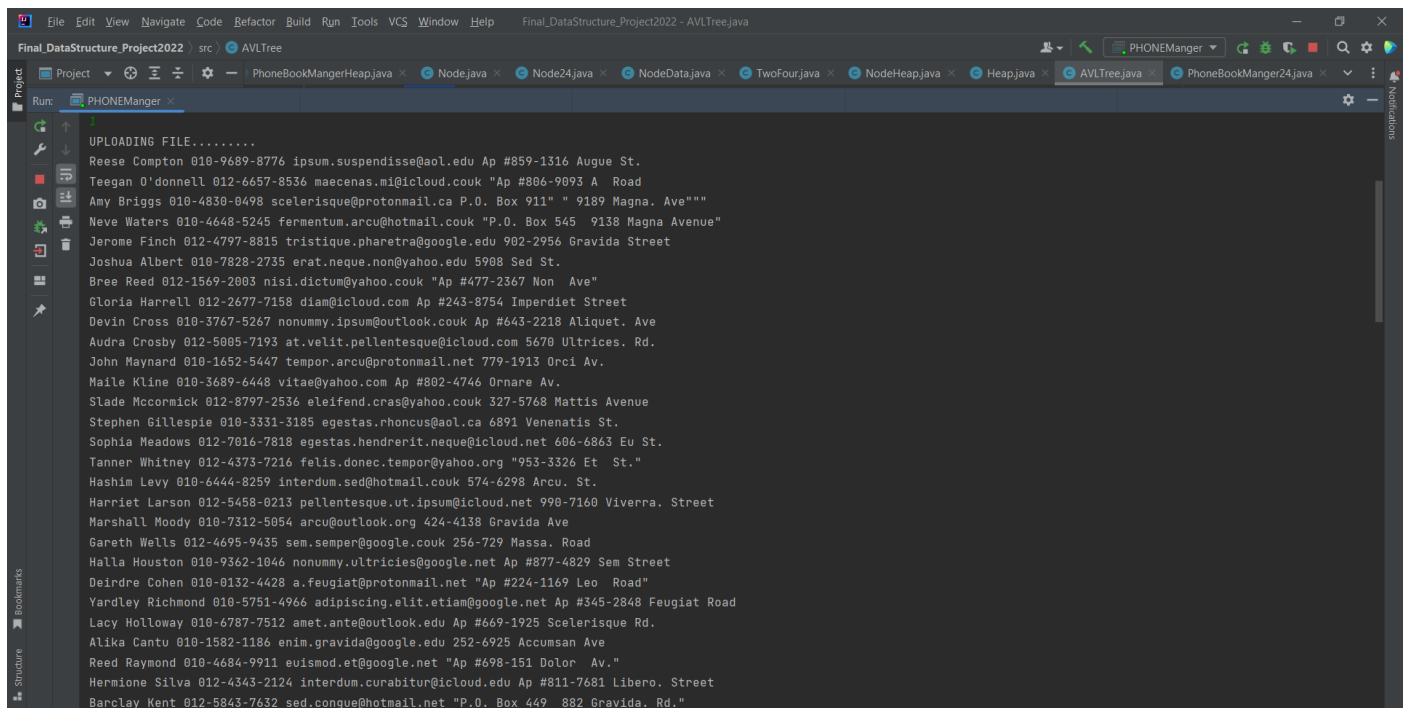
1. Start menu



```

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Final_DataStructure_Project2022 - AVLTree.java
Final_DataStructure_Project2022 | src | AVLTree
Run: PHONEManger
"C:\Program Files\Java\jdk-18.0.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.1\lib\idea_rt.jar=57308:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.1\bin" -Dfile.encoding=UTF-8 -classpath "C:\Users\Abdelrahman Raslan\IdeaProjects\Final_DataStructure_Project2022\out\production\Final_DataStructure_Project2022" PHONEManger
----MADE BY----
ABDELRAHMAN RASLAN
REWAN SALAH
AML IBRAHIM
SUHAILA ADEL
---Welcome To our Phone Book Management Application---
1) Show Contact file
2) AVL Tree
3) 2-4 Tree
4) Heap
5) EXIT
  
```

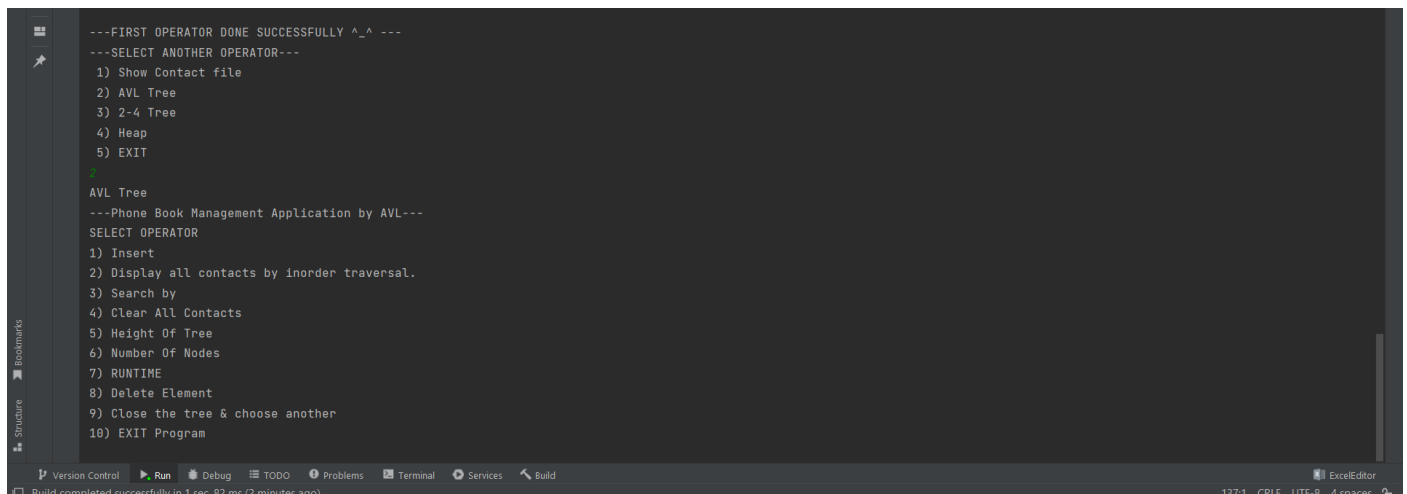
2. Show Contact file (A simple sample of the data that appears)



```

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Final_DataStructure_Project2022 - AVLTree.java
Final_DataStructure_Project2022 | src | AVLTree
Run: PHONEManger
UPLOADING FILE.....
Reese Compton 010-9689-8776 ipsum.suspendisse@aol.edu Ap #859-1316 Augue St.
Teegan O'donnell 012-6657-8536 maecenas.mi@icloud.couk "Ap #806-9093 A Road
Amy Briggs 010-4830-0498 scelerisque@protonmail.ca P.O. Box 911" " 9189 Magna. Ave""
Neve Waters 010-4648-5245 fermentum.arcu@hotmail.couk "P.O. Box 545 9138 Magna Avenue"
Jerome Finch 012-4797-8815 tristique.pharetra@google.edu 902-2956 Gravida Street
Joshua Albert 010-7828-2735 erat.neque.non@yahoo.edu 5908 Sed St.
Bree Reed 012-1569-2803 nisi.dictum@yahoo.couk "Ap #477-2367 Non Ave"
Gloria Harrell 012-2677-7158 diam@icloud.com Ap #243-8754 Imperdiet Street
Devlin Cross 010-3767-5267 nonummy.ipsum@outlook.couk Ap #643-2218 Aliquet. Ave
Audra Crosby 012-5005-7193 at.velit.pellentesque@icloud.com 5670 Ultrices. Rd.
John Maynard 010-1652-5447 tempor.arcu@protonmail.net 779-1913 Orci Av.
Maile Kline 010-3689-6448 vitae@yahoo.com Ap #802-4746 Ornare Av.
Slade McCormick 012-8797-2536 eleifend.cras@yahoo.couk 327-5768 Mattis Avenue
Stephen Gillespie 010-3331-3185 egestas.rhoncus@aol.ca 6891 Venenatis St.
Sophia Meadows 012-7016-7818 egestas.hendrerit.neque@icloud.net 606-6863 Eu St.
Tanner Whitney 012-4373-7216 felis.donec.tempor@yahoo.org "953-3326 Et St."
Hashim Levy 010-6444-8259 interdum.sed@hotmail.couk 574-6298 Arcu. St.
Harriet Larson 012-5458-0213 pellentesque.ut.ipsum@icloud.net 990-7160 Viverra. Street
Marshall Moody 010-7312-5854 arcu@outlook.org 424-4138 Gravida Ave
Gareth Wells 012-4695-9435 sem.semper@google.couk 256-729 Massa. Road
Halla Houston 010-9362-1046 nonummy.ultricies@google.net Ap #877-4829 Sem Street
Deirdre Cohen 010-0132-4428 a.feugiat@protonmail.net "Ap #224-1169 Leo Road"
Yardley Richmond 010-5751-4966 adipiscing.elit.etiam@google.net Ap #345-2848 Feugiat Road
Lacy Holloway 010-6787-7512 amet.ante@outlook.edu Ap #669-1925 Scelerisque Rd.
Alika Cantu 010-1582-1186 enim.gravida@google.edu 252-6925 Accumsan Ave
Reed Raymond 010-4684-9911 euismod.et@google.net "Ap #698-151 Dolor Av."
Hermione Silva 012-4343-2124 interdum.curabitur@icloud.edu Ap #811-7681 Libero. Street
Barclay Kent 012-5843-7632 sed.congue@hotmail.net "P.O. Box 449 882 Gravida. Rd."
  
```

3. AVL menu



```

---FIRST OPERATOR DONE SUCCESSFULLY ^_ ^_ ---
---SELECT ANOTHER OPERATOR---
1) Show Contact file
2) AVL Tree
3) 2-4 Tree
4) Heap
5) EXIT

AVL Tree
---Phone Book Management Application by AVL---
SELECT OPERATOR
1) Insert
2) Display all contacts by inorder traversal.
3) Search by
4) Clear All Contacts
5) Height Of Tree
6) Number Of Nodes
7) RUNTIME
8) Delete Element
9) Close the tree & choose another
10) EXIT Program
  
```

Version Control Run Debug TODO Problems Terminal Services Build

Build completed successfully in 1 sec, 82 ms (2 minutes ago)

ExcelEditor 137:1 CRLF UTF-8 4 spaces

4. AVL Insertion option & sort by option (we test now on first name sorting)

```

Choose index to sort by it (0= first name,1 = last name,2= full name)

Choose option
1) .csv file
2) Manually typing
3) CANCEL
  
```

5. Csv file insertion on AVL Tree (A sample of the data that appears line by line)

```

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Final_DataStructure_Project2022 - AVLTree.java
Final_DataStructure_Project2022 src AVLTree
Run: PHONEManager
3) CANCEL
---Phone Contacts---
Line 1: Teegan,O'donnell,012-6657-8536,maecenas.m@icloud.couk,"Ap #806-9093 A, Road
Line 2: Amy,Briggs,010-4830-0498,scelerisque@protonmail.ca,P.O. Box 911"," 9189 Magna. Ave""
Line 3: Weve,Waters,010-4648-5245,fermentum.arcu@hotmail.couk,"P.O. Box 545, 9138 Magna Avenue",
Line 4: Jerome,Finch,012-4797-8815,tristique.pharetra@google.edu,902-2956 Gravida Street,
Line 5: Joshua,Albert,010-7828-2735,erat.neque.non@yahoo.edu,5908 Sed St.,
Line 6: Bree,Reed,012-1569-2003,nisi.dictum@yahoo.couk,"Ap #477-2367 Non, Ave",
Line 7: Gloria,Harrell,012-2677-7158,diam@icloud.com,Ap #243-8754 Imperdiet Street,
Line 8: Devin,Cross,010-3767-5267,nonummy.ipsu@outlook.couk,Ap #643-2218 Aliquet. Ave,
Line 9: Audra,Crosby,012-5005-7193,at.velit.pellentesque@icloud.com,5670 Ultrices. Rd.,
Line 10: John,Maynard,010-1652-5447,tempor.arcu@protonmail.net,779-1913 Orci Av.,
Line 11: Maile,Kline,010-3689-6448,vitae@yahoo.com,Ap #802-4746 Ornare Av.,
Line 12: Slade,Mccormick,012-8797-2536,eleifend.cras@yahoo.couk,327-5768 Mattis Avenue,
Line 13: Stephen,Gillespie,010-3331-3185,egestas.rhoncus@aol.ca,6891 Venenatis St.,
Line 14: Sophia,Meadows,012-7016-7818,egestas.hendrerit.neque@icloud.net,606-6863 Eu St.,
Line 15: Tanner,Whitney,012-4373-7216,felis.donec.tempor@yahoo.org,"953-3326 Et, St.",
Line 16: Hashim,Levy,010-6444-8259,interdum.sed@hotmail.couk,574-6298 Arcu. St.,
Line 17: Harriet,Larson,012-5458-0213,pellentesque.ut.ipsu@icloud.net,990-7160 Viverra. Street,
Line 18: Marshall,Moody,010-7312-5054,arcu@outlook.org,424-4138 Gravida Ave,
Line 19: Gareth,Wells,012-4695-9435,sem.semper@google.couk,256-729 Massa. Road,
Line 20: Halla,Houston,010-9362-1046,nonummy.ultrices@google.net,Ap #877-4829 Sem Street,
Line 21: Deirdre,Cohen,010-0132-4428,a.feugiat@protonmail.net,"Ap #224-1169 Leo, Road",
Line 22: Yardley,Richmond,010-5751-4966,adipiscing.elit.etiam@google.net,Ap #345-2848 Feugiat Road,
Line 23: Lacy,Holloway,010-6787-7512,amet.ante@outlook.edu,Ap #669-1925 Scelerisque Rd.,
Line 24: Alike,Cantu,010-1582-1186,enim.gravida@google.edu,252-6925 Accumsan Ave,
Line 25: Reed,Ramond,010-4684-9911,euismod.et@google.net,"Ap #698-151 Dolor, Av.",
Line 26: Hermione,Silva,012-4343-2124,interdum.curabitur@icloud.edu,Ap #811-7681 Libero. Street,
  
```

6. Manually insertion on AVL Tree

```

Choose option
1) .csv file
2) Manually typing
3) CANCEL
Enter Node Manually :
Abdelrahman,Raslan,010-000-000,aras@alexu.eg,Alex
Inserting Done
  
```

7. Display all contacts by inorder traversal on AVL Tree (A sample of the data by first name sorting – you can use last name or full name in first step)

```

Run: PHONEManager
---Contact InOrderTraversal by First Name---
Abdelrahman raslan 010-000-000 aras@alexu.eg alex
abigail cash 010-4455-3583 molestie.sed@outlook.org 5854 vel st.
aidan strickland 012-8087-4207 aliquam@google.ca 795-3162 netus st.
akeem rodriguez 012-9431-8188 lorem.auctor@protonmail.ca "173-6466 non
alan holder 010-4312-1471 nunc.ut@aol.org 452-1570 quisque av.
alexander delgado 010-8843-8269 arcu.sed.eu@aol.org "p.o. box 874
alika cantu 010-1582-1186 enim.gravida@google.edu 252-6925 accumsan ave
alyssa bullock 010-5310-9702 amet@icloud.org ap #668-2438 tincidunt rd.
amelia farley 012-6785-6539 magnis.dis@google.edu ap #378-1888 cursus rd.
amy briggs 010-4830-0498 scelerisque@protonmail.ca p.o. box 911"
angela saunders 010-7510-1696 nulla.tempor.augue@protonmail.ca ap #144-3575 hendrerit. st.
astra petersen 012-7430-9888 diam.sed.diam@hotmail.com ap #440-820 facillisis street
audra crosby 012-5005-7193 at.velit.pellentesque@icloud.com 5670 ultrices. rd.
azalia deLeon 012-5868-6827 per@aol.org 7995 proin av.
barclay kent 012-5843-7632 sed.congue@hotmail.net "p.o. box 449
beck herrera 012-1234-7296 faucibus.orci@google.edu 4756 dictum. st.
bianca barker 012-5889-4455 mollis.dui.in@google.couk 7451 magnis st.
blaze franklin 012-1297-1615 augue.malesuada@protonmail.ca 541-9087 venenatis st.
bree reed 012-1569-2003 nisi.dictum@yahoo.couk "ap #477-2367 non
chandler olson 012-5813-2838 dui.risus@aol.org 800-2369 et av.
chiquita valenzuela 010-8781-3536 scelerisque.mollis@protonmail.edu "p.o. box 971
colleen huff 010-3477-8028 non.bibendum@yahoo.couk ap #836-4539 sed avenue
davis malone 010-1813-1912 massa.vestibulum@aol.org 907-6881 magnis st.
delcote cohen 010-0132-4428 a.feugiat@protonmail.net "ap #224-1169 leo
derek kent 010-3303-3079 vel.venenatis@protonmail.com "4833 amet
devin cross 010-3767-5267 nonummy.ipsu@outlook.couk ap #643-2218 aliquet. ave
drake joyce 010-6211-1787 nunc@icloud.net ap #981-6498 iaculis avenue
duncan rush 012-1335-5814 convallis.est@icloud.couk 980-3666 consequat av.
  
```

8. Searching on AVL Tree (by first name, you can use last name or full name in first step)

```

Enter the First Name you want to search by : yardley
Does this Contact contain "yardley" ?
yardley richmond 810-5751-4966 adipiscing.elit.etiam@google.net ap #345-2848 feugiat road .

```

9. Height of AVL Tree

```

Height Of tree : 8

```

10. Number of Nodes on AVL Tree (by Full name & delete null and repeater lines)

```

Number of Nodes : 99

```

NO DE deleted (null)

11. Delete and check by search method

```

Enter Full Name You Want to delete
Sage Payne
Contact Deleted Successfully !

```

```

Enter the Full Name you want to search by Sage Payne
Does this Contact contain "sage" ?
No there is not such a Contact like "sage".

```

12. Clear all nodes and check by number of node number

```

1) Insert
2) Display all contacts by inorder traversal.
3) Search by
4) Clear All Contacts
5) Height Of Tree
6) Number Of Nodes
7) RUNTIME
8) Delete Element
9) Close the tree & choose another
10) EXIT Program

```

All Contacts removed from the Tree.

SELECT NEXT OPERATOR:

```

1) Insert
2) Display all contacts by inorder traversal.
3) Search by
4) Clear All Contacts
5) Height Of Tree
6) Number Of Nodes
7) RUNTIME
8) Delete Element
9) Close the tree & choose another
10) EXIT Program

```

Number of Nodes : 0

13. Exit program of AVL (you can only exit tree & go to another tree by type "9" not "10")

```

10) EXIT Program
Thanks!! Best Regards, ARAS

```

14. 2-4 menu

```
2-4 Tree
---Phone Book Management Application by 2-4 Tree---
SELECT OPERATOR
1) Insert
2) Display all contacts by inorder traversal & Show Height .
3) Search by
4) RUNTIME
5) Delete Element
6) Close the tree & choose another
7) EXIT Programme
```

15. AVL Insertion option & sort by option (we test now on last name sorting)

```
Choose index to sort by it (0= first name,1 = last name,2= full name)
Choose option
1).csv file
2) Manually typing
3) CANCEL
```

16. Csv file insertion on 2-4 Tree

```
Choose option
1).csv file
2) Manually typing
3) CANCEL

INSERTING.....

Inserting Done
```

17. Manually Input on 2-4 Tree

```
2) Manually typing
3) CANCEL
---Enter Manually---
Enter First Name :
Name
Enter Last Name :
Name
Enter Email Address :
Name@gmail.com
Enter Address :
Name
Enter Phone Number :
012-345-6789
SELECT NEXT OPERATOR:
1) Insert
```

18. Display Last name inorder traversal & Show Height

```

Tree:
level=0 child=0 McCormick
Height of tree is : 0
level=1 child=0 Compton Finch
Height of tree is : 1
level=2 child=0 Briggs
Height of tree is : 2
level=3 child=0 Barker
Height of tree is : 3
level=4 child=0 Albert Ayers
Height of tree is : 4
level=4 child=1 Bass Bates Bray
Height of tree is : 4
level=3 child=1 Cantu Christensen
Height of tree is : 3
level=4 child=0 Brock Bullock Cantu
Height of tree is : 4
level=4 child=1 Cash Chandler
Height of tree is : 4
level=4 child=2 Cohen
Height of tree is : 4
level=2 child=1 Curry
Height of tree is : 2
level=3 child=0 Crosby
Height of tree is : 3
level=4 child=0 Cotton
Height of tree is : 4
level=4 child=1 Cross
Height of tree is : 4

```

```

Height of tree is : 4
level=4 child=2 Rush Salah
Height of tree is : 4
level=3 child=1 Shepard
Height of tree is : 3
level=4 child=0 Saunders Shelton
Height of tree is : 4
level=4 child=1 Shepard Sherman Sherman
Height of tree is : 4
level=2 child=2 Valenzuela
Height of tree is : 2
level=3 child=0 Stephens Summers
Height of tree is : 3
level=4 child=0 Skinner Slater
Height of tree is : 4
level=4 child=1 Strickland
Height of tree is : 4
level=4 child=2 Talley Thornton
Height of tree is : 4
level=3 child=1 Waters Wells
Height of tree is : 3
level=4 child=0 Walker
Height of tree is : 4
level=4 child=1 Welch
Height of tree is : 4
level=4 child=2 West Whitney Yates
Height of tree is : 4

```

Final Height

19. Search by last name on 2-4 Tree (can also search by first name and full name)

```

Enter Last Name to find:
Whitney
NodeData{FirstName=Tanner, LastName=Whitney, Address="953-3326 Et, EmailAddress=felis.donec.tempor@yahoo.org, Phonenumber=012-4373-7216}
Found Whitney

```

20. Delete node on 2-4 Tree

```

Enter Last Name to delete:
Whitney
NodeData{FirstName=Tanner, LastName=Whitney, Address="953-3326 Et, EmailAddress=felis.donec.tempor@yahoo.org, Phonenumber=012-4373-7216}
DeletedWhitney

```

21. Close the tree 2-4 Tree & choose another (can exit program, type "6")

```

6) Close the tree & choose another
7) EXIT Programme

GOOD BYE

---FIRST OPERATOR DONE SUCCESSFULLY ^_^ ---
---SELECT ANOTHER OPERATOR---
1) Show Contact file
2) AVL Tree
3) 2-4 Tree
4) Heap
5) EXIT

```

22. Heap menu

```

heap
---Phone Book Management Application by Heap---
SELECT OPERATOR
1) Insert by csv File
2) Display all contacts by inorder traversal.
3) Search by
4) Remove Root
5) Height Of Tree
6) Number Of Nodes
7) RUNTIME
8) Delete Element
9) Display Heap
10) Close the tree & choose another
11) EXIT Programme

```

23. Csv file insertion & Display by inorder traversal on Heap

(A sample of the data that appears line by line)

```

Final_DataStructure_Project2022 | src | PhoneBookManger24 | main_24test
Project | PhoneBookMangerHeap.java | Node.java | Node24.java | NodeData.java | TwoFour.java | NodeHeap.java | Heap.java | AVLTree.java | PhoneBookManger24.java
Run: PHONEManger

FirstName: Harriet LastName: Larso Email: pellentesque.ut.ipsu@icloud.net Address: 998-7168 Viverra, Street Phone: 012-5458-0213
FirstName: Indira LastName: Joyne Email: dui.gravida.praesent@yahoo.ca Address: 570-2001 Nibh St. Phone: 010-1433-6446
FirstName: Drake LastName: Joyc Email: nunc@icloud.net Address: Ap #901-6498 Iaculis Avenue Phone: 010-6211-1787
FirstName: Leilani LastName: Welc Email: rhoncus@hotmail.net Address: 449-9676 Elit. Ave Phone: 012-2304-5277
FirstName: Ezekiel LastName: Thornto Email: nec.enim@hotmail.edu Address: "812 Nostra Phone: 010-2682-5831
FirstName: Lucas LastName: Chandle Email: non.justo.proin@outlook.com Address: Ap #460-9233 Cras Ave Phone: 012-4358-1911
FirstName: Derek LastName: Ken Email: vel.venenatis@protonmail.com Address: "4833 Amet Phone: 010-3303-3079
FirstName: Mollie LastName: Cant Email: proin.vel@aol.net Address: 4294 Felis Av. Phone: 010-8385-1887
FirstName: Hashim LastName: Lev Email: interdum.sed@hotmail.couk Address: 574-6298 Arcu. St. Phone: 010-6444-8259
FirstName: Ila LastName: Ayer Email: aliquam@yahoo.edu Address: 6249 Posuere Rd. Phone: 010-9212-3779
FirstName: Devin LastName: Cros Email: nonummy.ipsu@outlook.couk Address: Ap #643-2218 Aliquet. Ave Phone: 010-3767-5267
FirstName: Hashim LastName: Palme Email: nibh.dolor@protonmail.com Address: 139-7493 At Road Phone: 010-6677-9185
FirstName: Felicia LastName: Daughert Email: eget.mollis@aol.couk Address: "P.O. Box 784 Phone: 012-8584-6248
FirstName: Michael LastName: Merril Email: purus.nullam@yahoo.com Address: Ap #929-5921 Integer Rd. Phone: 010-5472-8865
FirstName: Astra LastName: Peterse Email: diam.sed.diam@hotmail.com Address: Ap #440-820 Facilisis Street Phone: 012-7430-9888

```

24. Search on Heap

```

3
Enter the name you want to search by : Lucy Littl
Contact found:
FirstName: Lucy LastName: Littl Email: quis.diam@yahoo.ca Address: 977-8078 Lectus St. Phone: 012-1480-3565

```


25. Remove root on Heap

```
REMOVING ROOT.....
Deleted Contact:
  FirstName: Aidan   LastName: Stricklan   Email: alikuan@google.ca   Address: 795-3162 Netus St.   Phone: 012-8087-4207
```

26. Height of Heap

```
27) Exit Programme
Height: 6
```

27. Number of Nodes on Heap

```
27) Exit Programme
Number of nodes: 98
```

We deleted node and root $100 - 2 = 98$

28. Display Heap (sample of nodes)

PARENT NODE	LEFT CHILD NODE	RIGHT CHILD NODE
Akeem Rodrique	Alan Holde	Amelia Farle
Alan Holde	Astra Peterse	Alexander Delgad
Amelia Farle	Barclay Ken	Angela Saunder
Astra Peterse	Derek Ken	Azalia Deleo
Alexander Delgad	Amy Brigg	Alika Cant
Barclay Ken	Beck Herrerr	Bianca Barke
Angela Saunder	Blaze Frankli	Audra Crosb
Derek Ken	Drake Joyc	Devin Cros
Azalia Deleo	Felix Christense	Guy Goodma
Amy Brigg	Gloria Ferguso	Bree Ree
Alika Cant	Alyssa Bulloc	Derek Mcfarlan
Beck Herrerr	Hiram Skinne	Hilda Pec
Bianca Barke	Duncan Rus	Hermione Silv
Blaze Frankli	Dylan Walke	Chiquita Valenzuel
Audra Crosb	Harriet Mori	Colleen Huf
Drake Joyc	Harriet Larso	Ezekiel Thornto
Devin Cros	Hashim Lev	Felicia Daughert
Felix Christense	Marshall Mood	Gareth Well
Guy Goodma	Joshua Alber	Patrick Slate
Gloria Ferguso	Morgan Lower	Halla Housto
Bree Ree	John Maynar	Deirdre Cohe
Alyssa Bulloc	Jenna Salaza	Chandler Olso
Derek Mcfarlan	Kennedy Shelto	Kareem Floy
Hiram Skinne	Reagan Yate	Kibo Wes
Hilda Pec	Slade Mccormic	Nicole Pec
Duncan Rus	Stephen Gillespi	Gloria Harrel
Hermione Silv	Philip Rio	Magee Pool

Trees Algorithms:

1) 2-4 Tree

i. Management of data

Setters & getters:

getChild → return childArray[childNum]

getParent → return parent

setNumberOfItems → numItems = theValue

getNumberOfItems → return numItems

setItem → itemArray[index] = theValue

return itemArray[index]

getItem → return itemArray[index]

getSiblings → if (numItems != 0)

for loop (i <= parent.numItem)

if (parent.childArray[i].itemArray[0] == null)

if (i != 0) x = p.childArray[i - 1]

ii. Connect Children:

- Put the child number in an array called child
- if (child != null) → connect this child to the parent

iii. disconnect Children:

- Put the child number in an array called child
- Make it equals null

iv. isLeaf:

- If (childArray[0] == null) → return true
- Else → return false

v. isFull:

- if (numItems == order - 1) → return true
- else → return false

vi. Insertion:

- 1) If(node == 4) → remove the middle value , split the remaining 3 nodes into pair of 2 nodes
 If(node == root) → the middle value becomes the new root 2-node, height is increased by one
 Else → push the middle value into the parent node
- 2) Search for the child whose interval has the value to be inserted
- 3) If(child == leaf) → just insert the value into a child
- 4) Else → descend into the child and repeat from step 1

vii. deletion:

- 1) If(node == leaf && key >= 2) → delete the node
- 2) If(leftChild == internalNode && key >= 2)
 Replace it with its predecessor then delete is recursively
 Else if(rightChild == internalNode && key >= 2)
 Replace it with its successor then delete is recursively
- 3) If [(leftChild || rightChild != internalNode) && (keyOfChildNode == 1) && (keyofSibling >= 2)]
 Move it from the parent into the child & move the element from the sibling into the parent

 if(keyOfChildNode == 1 && keyofSibling == 1)
 merge the child node with one of the siblings & move an element from the parent to the merged node

viii. File Management:

- 1) You choose the type of sorting :
 If (userChoice == 1) → Sort by firstName
 Else if (userChoice == 2) → Sort by lastName
 Else → sort by fullName
- 2) While(true)
 If(Node.isFull()) → split & getParent() the getNextChild()
 Else if (Node.isLeaf()) → break & stop the program
 Else → getNextChild

ix. Split:

- 1) Pick the middle element & move it to the parent
If(`root.hasParent() == false`) → create new node & move the middle element to the new root
- 2) Create two 2-Node & move the left element to one node & the right element to the other
- 3) If the original 4-Node were pointing to the same children would be now pointed by the created 2-Nodes

x. getNextChild:

- 1) For loop (`getNumberOfItems`){
If (`items < 0`) → return child
}
Return child

xi. Rotation:**1) Left**

- Get the smallest key in the root & move it to the 2-Node.
- Pick the highest key in the left sibling & move it to root.
- If the sibling has a subtree of interval > the highest key in the left sibling , subtree is also moved under the original 2-Nodes.

2) Right

- Get the highest key in the root & move it to the 2-Node.
- Pick the smallest key in the right sibling & move it to root.
 - If the sibling has a subtree of interval < the smallest key in the right sibling , subtree is also moved under the original 2-Nodes.

xii. Re-display tree:

- Display the node & get the highest level
For loop (`j < numItems + 1`) → `getChild(j)`
If(`nextNode != null`) → print (level+1)

xiii. In-order display:

- it is same as “Re-display tree” + sorting by inorder

xiv. Find the value:

- For loop($j < \text{numItems}$)
If ($\text{item } j == 0 \mid \mid \text{firstName} == 0 \mid \mid \text{lastName} == 0$) \rightarrow get the item then break

Else if [($\text{item } j < 0 \ \&\& \text{node} != \text{leaf}$) $\mid \mid$ ($\text{firstName} < 0 \ \&\& \text{node} != \text{leaf}$) $\mid \mid$ ($\text{lastName} < 0 \ \&\& \text{node} != \text{leaf}$)]
break

Else if [($\text{item } j > 0 \ \&\& \text{node} != \text{leaf}$) $\mid \mid$ ($\text{firstName} > 0 \ \&\& \text{node} != \text{leaf}$) $\mid \mid$ ($\text{lastName} > 0 \ \&\& \text{node} != \text{leaf}$)]
break

xv. Time complexity :

- Search in 2-3-4 Tree takes $O(\log n)$ in case that the tree is always perfectly balanced.
- Insertion takes $O(\log n)$ as all the splits are local transformations and we don't need multiple passes on the tree.
- Deletion takes $O(\log n)$ considering rotation/fusion/shrink are all $O(1)$ operations.

2) Heap

i. getParentPosition(int p):

- return $p / 2$

ii. getLeftChildPosition (int p):

- return $2 * p$

iii. getRightChildPosition(int p):

- return $(2 * p) + 1$

iv. checkLeaf(int p):

- if $p \geq \text{size}/2$ & $p \leq \text{size}$
return true

v. swap(int f , int s):

- Node temp
- Temp=heap[f]
- Heap[f]=heap[s]
- Heap[s]=temp

vi. minHeapify(int p):

- If !checkLeaf(p)
- If heap[p].getName() > heap[getleft(p)].getName() or heap[p].getName() > heap[getRight(p)].getName()
- If heap[getleft(p)].getName() < heap[getRight(p)].getName()
- Swap(p , getleft(p))
- minHeapify(getleft(p))
- else
swap(p, getRight(P))
- minHeapify(getright(p))

vii. insertNode (Node d):

- If(size>=maxSize)
Return
- Heap[++size]=d
- c=size
- while heap[c].getName()<heap[getpostion(c)].getName()
- swap(c,getpostion(c))
- c=getposition(c)

viii. displayHeap():

- For (int k=1 ; k<=size/2 ;k++)
- If heap[2*k+1] ==null then
- Print heap[k].getName() and heap[2*k].getName()
- else
print heap[k].getName() and heap[2*k].getName() and heap[2*k+1].getName()

ix. designMinHeap():

- For (int p=size/2 ; p>=1 ; p--)
- minHeapify(P)

x. removeRoot():

- deletecontact(heap[1].getName())

xi. search(String v):

- for(int i=0 ; i<size ; i++)
- if heap[i].getName() contain (v)
- print heap[i]

xii. getContactIndex(String name):

- for(int i = 1; i <= size; i++)
- if heap[i].getName()==name
return i

xiii. deleteContact(String name):

- Index=getContactIndex(name)
- If index==-1 then
- Contact not found
- Swap(index , size)
- Node deleted = heap[size]
- heap[size]=null
- minHeapify(index);
- size--
- print deleted

xiv. inorderDisplay(int index):

- If heap[index]==null then
return
- inorderDisplay(2*index)
- print heap[index]
- inorderDisplay((2*index)+1)

xv. height():

- p=1 , h=0
- while heap[2*p]!=null & 2*p<=heap.length-1
- p*=2
- h++
- return h

3) AVL Tree

1. Create an object of AVL node and assign values to it (element, height, count, Node left, Node Right).

2. create a constructor that contains all the data of the file (first name, last name, phone number, email, address) and initialize (height = 1, node left, right = null, count = 1)

3. Create a class of AVL Tree then create a new node

Node root = null

4. Create constructor that initializes root = null

Methods:

i. height(Node node) : checks the height of the AVL tree

- If (node == null) → Return 0
- Else → return height

ii. getBalance(Node node) : manages the balance of the tree

- if (root = null) → return 0
- else → return (height of left – height of right)

iii. maxHeight(int leftHeight, int rightHeight) :

- returns the maximum height of the tree (right or left)

iv. Node rightRotate(Node grandParent)

- rotates the tree in the right side
- parent = grandParent.getLeft()
- child = parent.getRight()
- then make a set up for all the nodes (left, right, parents, grandParents, children) and the height of the tree

v. Node leftRotate(Node grandParent) :

- rotates the tree in the left side
- parent = grandParent.getRight()
- child = parent.getLeft()
- then make a set up for all the nodes (left, right, parents, grandParents, children) and the height of the tree

insertion in AVL tree

- 1) Search for the subtree where the new value should be added and compare these values
- 2) Then create a new node in this subtree
- 3) This new node is a leaf
- 4) Return parent and set the balance up through the rotation

deletion in AVL tree

1) Select the node you want to delete

2) If (node == leaf) → delete

Else if (child == 1) → replace the target node with the child then delete it

Else if (child == 2) → find the inorder successor node that has (child == 0) then replace the target node with the inorder successor node.

3) Rearrange the tree & set the balance up.

vi. CalculateHeight : returns the height of the root

vii. deleteTree : return root = null

viii. inOrder(Node node) :

```
if (node != null) → inOrder(node.getLeft())
    System.out.println(node.getLine())
    inOrder(node.getRight())
```

ix. search (String word) :

```
while loop (root != null)
    if (root < 0) → get the left root
    else if (root > 0) → get the right root
    else → return false
```

x. public returnNode(String word) : it is the same as the search + (if false → return null)

xi. private getTotalNumberOfNodes() : calculate the number of nodes

```
if (head == null) → return 0
else → length = 1, update length = getTotalNumberOf (left, right) then return length
```

xii. public getTotalNumberOfNodes() : uses the private method to prevent unauthorized access for the method

xiii. getReplacementRecursive :

```
if (node == null) → Return null
else → replace the node, right node and the target
then you must adjust and update the height and balance of the tree
```

xiv. updateHeight(Node node):

```
if (leftNode != null) → LeftHeight = node.getLeft().getHeight()
if (rightNode != null) → RightHeight = node.getRight().getHeight()
then set the max height [setHeight(max(iLeftHeight, iRightHeight) + 1)]
```

xv. max(int first, int second) :

```
return Math.max(first, second)
```


- xvi. clearAll (AVLTree tree) : checks if the tree is empty or not and clear the all tree
if (root == null) → the tree is already empty
else → delete the whole tree
- xvii. contains(String element, AVLTree tree): checks if the element is exists in the tree or not
if (search(element) == true) → print(element)
else → print("the element does not exist")
- xviii. main test(): it tests the all of previous methods ,
calculates the Run Time by using [start=System.currentTimeMillis()] ,
views the menu on the screen ,
takes the user's choice as an input then let them choose the way of sorting (by first name , last name , full name) ,
then input the file , read it and make an index for every line (row) ,
makes inorder traversal to the file of contacts
closes the program.