DS Program Manual

1. Populating The Data Structure:

a) Enter number 1

```
Welcome to the Budget Management System!
---------------------------------------
Menu:
1. Generate Expenses
2. Add Expense
3. Delete Expense
4. Update Expense
5. Sort expenses
6. Search Expenses
7. Print Expenses
8. Clear Data Structures
9. Exit
---------------------------------------
Choose an option: 1
```

b) enter the amount of data to generate:

```
Welcome to the Budget Management System!
---------------------------------------
Menu:
1. Generate Expenses
2. Add Expense
3. Delete Expense
4. Update Expense
5. Sort expenses
6. Search Expenses
7. Print Expenses
8. Clear Data Structures
9. Exit
---------------------------------------
Choose an option: 1
Enter the number of expenses to generate: 1000
```

c) performance metric printed

```
Arraylist Performance Metrics:
|-- Memory Usage: 2,672.00 bytes
|-- Time Taken: 33,100.00 Nanoseconds

LinkedList Performance Metrics:
|-- Memory Usage: 2,672.00 bytes
|-- Time Taken: 77,100.00 Nanoseconds

No expense found at index 0
HashMap:  Performance Metrics:
|-- Memory Usage: 2,672.00 bytes
|-- Time Taken: 484,000.00 Nanoseconds

Stack Performance Metrics:
|-- Memory Usage: 11,664.00 bytes
|-- Time Taken: 893,700.00 Nanoseconds
```

```
AVL Tree Performance Metrics:
|-- Memory Usage: 2,648.00 bytes
|-- Time Taken: 118,800.00 Nanoseconds
```

2. Addition of Data:

a) Enter number 2

```
----------------------------------------
Menu:
1. Generate Expenses
2. Add Expense
3. Delete Expense
4. Update Expense
5. Sort expenses
6. Search Expenses
7. Print Expenses
8. Clear Data Structures
9. Exit
----------------------------------------
Choose an option: 2
```

b) choose which data structure

```
Menu:
1. Generate Expenses
2. Add Expense
3. Delete Expense
4. Update Expense
5. Sort expenses
6. Search Expenses
7. Print Expenses
8. Clear Data Structures
9. Exit
----------------------------------------
Choose an option: 2
Choose a data structure to add an expense:
1. arrayList, linkedList, hashMap, Stack (Addition by index)
2. AVL Tree (Addition by Value)
3. Back to main menu
```

c) Choose the index or amount to add

```
Choose an option: 2
Choose a data structure to add an expense:
1. arrayList, linkedList, hashMap, Stack (Addition by index)
2. AVL Tree (Addition by Value)
3. Back to main menu
1
Enter the index to add the expense: 0
```

```
Choose an option: 2
Choose a data structure to add an expense:
1. arrayList, linkedList, hashMap, Stack (Addition by index)
2. AVL Tree (Addition by Value)
3. Back to main menu
2
Enter the amount to add to the AVL Tree: 10
```

d) performance metric printed:

```
Arraylist Performance Metrics:
|-- Memory Usage: 2,672.00 bytes
|-- Time Taken: 33,100.00 Nanoseconds

LinkedList Performance Metrics:
|-- Memory Usage: 2,672.00 bytes
|-- Time Taken: 77,100.00 Nanoseconds

No expense found at index 0
HashMap:  Performance Metrics:
|-- Memory Usage: 2,672.00 bytes
|-- Time Taken: 484,000.00 Nanoseconds

Stack Performance Metrics:
|-- Memory Usage: 11,664.00 bytes
|-- Time Taken: 893,700.00 Nanoseconds
```

```
AVL Tree Performance Metrics:
|-- Memory Usage: 2,648.00 bytes
|-- Time Taken: 118,800.00 Nanoseconds
```

3.Deleting Data

a) enter number 3

```
Menu:
1. Generate Expenses
2. Add Expense
3. Delete Expense
4. Update Expense
5. Sort expenses
6. Search Expenses
7. Print Expenses
8. Clear Data Structures
9. Exit
----------------------------------------
Choose an option: 3
```

b) Choose a data structure(1 or 2)

```
Menu:
1. Generate Expenses
2. Add Expense
3. Delete Expense
4. Update Expense
5. Sort expenses
6. Search Expenses
7. Print Expenses
8. Clear Data Structures
9. Exit
----------------------------------------
Choose an option: 3
Choose a data structure to delete an expense from:
1. arrayList, linkedList, hashMap, Stack (deletion by index)
2. AVL Tree (deletion by Value)
3. Back to main menu
```

c) enter the index or amount to delete

```
Choose a data structure to delete an expense from:
1. arrayList, linkedList, hashMap, Stack (deletion by index)
2. AVL Tree (deletion by Value)
3. Back to main menu
1
Enter the index to delete in the data structures: 0
```

```
Choose a data structure to delete an expense from:
1. arrayList, linkedList, hashMap, Stack (deletion by index)
2. AVL Tree (deletion by Value)
3. Back to main menu
2
Enter the amount to delete in AVl tree: 1
```

d) performance metric is printed

```
Arraylist Performance Metrics:
|-- Memory Usage: 2,672.00 bytes
|-- Time Taken: 33,100.00 Nanoseconds

LinkedList Performance Metrics:
|-- Memory Usage: 2,672.00 bytes
|-- Time Taken: 77,100.00 Nanoseconds

No expense found at index 0
HashMap:  Performance Metrics:
|-- Memory Usage: 2,672.00 bytes
|-- Time Taken: 484,000.00 Nanoseconds

Stack Performance Metrics:
|-- Memory Usage: 11,664.00 bytes
|-- Time Taken: 893,700.00 Nanoseconds
```

```
AVL Tree Performance Metrics:
|-- Memory Usage: 2,648.00 bytes
|-- Time Taken: 118,800.00 Nanoseconds
```

4. Searching Data

a) enter number 6

```
----------------------------------------
Menu:
1. Generate Expenses
2. Add Expense
3. Delete Expense
4. Update Expense
5. Sort expenses
6. Search Expenses
7. Print Expenses
8. Clear Data Structures
9. Exit
----------------------------------------
Choose an option: 6
```

b) choose which data structure to search form

```
----------------------------------------
Menu:
1. Generate Expenses
2. Add Expense
3. Delete Expense
4. Update Expense
5. Sort expenses
6. Search Expenses
7. Print Expenses
8. Clear Data Structures
9. Exit
----------------------------------------
Choose an option: 6
Choose a data structure to search an expense from:
1. arrayList, linkedList, hashMap, stack (search based on index)
2. AVL Tree (search based on Value)
3. Back to main menu
1
```

c) enter index or amount to search

```
Choose a data structure to search an expense from:
1. arrayList, linkedList, hashMap, stack (search based on index)
2. AVL Tree (search based on Value)
3. Back to main menu
1
Enter the index to search the expense: 10
```

```
Choose a data structure to update an expense from:
1. arrayList, linkedList, hashMap, stack (update based on index)
2. AVL Tree (update based on Value)
3. Back to main menu
2
(e.g. if yoou want to update a node with amount 30, you input 30 here)
Enter the amount to update: 10
```

d) Performance metric is printed

```
Arraylist Performance Metrics:
|-- Memory Usage: 2,672.00 bytes
|-- Time Taken: 33,100.00 Nanoseconds

LinkedList Performance Metrics:
|-- Memory Usage: 2,672.00 bytes
|-- Time Taken: 77,100.00 Nanoseconds

No expense found at index 0
HashMap:  Performance Metrics:
|-- Memory Usage: 2,672.00 bytes
|-- Time Taken: 484,000.00 Nanoseconds

Stack Performance Metrics:
|-- Memory Usage: 11,664.00 bytes
|-- Time Taken: 893,700.00 Nanoseconds
```

```
AVL Tree Performance Metrics:
|-- Memory Usage: 2,648.00 bytes
|-- Time Taken: 118,800.00 Nanoseconds
```

5. Updating Data:

a) enter number 4

```
Menu:
1. Generate Expenses
2. Add Expense
3. Delete Expense
4. Update Expense
5. Sort expenses
6. Search Expenses
7. Print Expenses
8. Clear Data Structures
9. Exit
----------------------------------------
Choose an option: 4
```

b) Choose which data structure to update (1 or 2)

```
Menu:
1. Generate Expenses
2. Add Expense
3. Delete Expense
4. Update Expense
5. Sort expenses
6. Search Expenses
7. Print Expenses
8. Clear Data Structures
9. Exit
----------------------------------------
Choose an option: 4
Choose a data structure to update an expense from:
1. arrayList, linkedList, hashMap, stack (update based on index)
2. AVL Tree (update based on Value)
3. Back to main menu
```

c) enter index or amount to update

```
Choose a data structure to update an expense from:
1. arrayList, linkedList, hashMap, stack (update based on index)
2. AVL Tree (update based on Value)
3. Back to main menu
1
Enter the index to update the expense: 0
```

```
Choose a data structure to update an expense from:
1. arrayList, linkedList, hashMap, stack (update based on index)
2. AVL Tree (update based on Value)
3. Back to main menu
2
```

d) Performance metric is printed:

```
Arraylist Performance Metrics:
|-- Memory Usage: 2,672.00 bytes
|-- Time Taken: 33,100.00 Nanoseconds

LinkedList Performance Metrics:
|-- Memory Usage: 2,672.00 bytes
|-- Time Taken: 77,100.00 Nanoseconds

No expense found at index 0
HashMap:  Performance Metrics:
|-- Memory Usage: 2,672.00 bytes
|-- Time Taken: 484,000.00 Nanoseconds

Stack Performance Metrics:
|-- Memory Usage: 11,664.00 bytes
|-- Time Taken: 893,700.00 Nanoseconds
```

```
AVL Tree Performance Metrics:
|-- Memory Usage: 2,648.00 bytes
|-- Time Taken: 118,800.00 Nanoseconds
```

6. Sorting Data

A) enter number 5 to sort all of the

```
Menu:
1. Generate Expenses
2. Add Expense
3. Delete Expense
4. Update Expense
5. Sort expenses
6. Search Expenses
7. Print Expenses
8. Clear Data Structures
9. Exit
----------------------------------------
Choose an option: 5
```

B) performance metric printed

```
Expenses sorted by amount.
Arraylist Performance Metrics:
|-- Memory Usage: 476,608.00 bytes
|-- Time Taken: 2,155,700.00 Nanoseconds

HashMap:  Performance Metrics:
|-- Memory Usage: 12,640.00 bytes
|-- Time Taken: 2,463,700.00 Nanoseconds

LinkedList Performance Metrics:
|-- Memory Usage: 534,368.00 bytes
|-- Time Taken: 3,547,100.00 Nanoseconds

Any expense objects added to the AVL Tree are automatically sorted based on their amount, from least to most.
Expenses sorted by amount.
Stack Performance Metrics:
|-- Memory Usage: 477,720.00 bytes
|-- Time Taken: 2,925,100.00 Nanoseconds
```

7. Printing data

a) enter number 7

```
---------------------------------------
Menu:
1. Generate Expenses
2. Add Expense
3. Delete Expense
4. Update Expense
5. Sort expenses
6. Search Expenses
7. Print Expenses
8. Clear Data Structures
9. Exit
---------------------------------------
Choose an option: 7
```

b) choose a data structure to be printed

```
Choose a data Struture that is to be printed:
(Max amount printed is 50)
1. ArrayList
2. LinkedList
3. Hashmap
4. AVL tree
5. Stack
6. Return to Main Menu
4
```

C) all content up to 50 entries will be printed

```
Viewing expenses in AVL Tree (max 50 entries):
Amount: 1.0, Year: 2024, Month: 12, Day: 13, Description: random expense , frequency: Daily
Amount: 1.0, Year: 2024, Month: 12, Day: 13, Description: random expense , frequency: Daily
Amount: 1.0, Year: 2024, Month: 12, Day: 13, Description: random expense , frequency: Daily
Amount: 1.0, Year: 2024, Month: 12, Day: 13, Description: random expense , frequency: Daily
Amount: 3.0, Year: 2024, Month: 11, Day: 28, Description: random expense , frequency: Daily
Amount: 3.0, Year: 2024, Month: 11, Day: 28, Description: random expense , frequency: Daily
Amount: 4.0, Year: 2024, Month: 5, Day: 13, Description: random expense , frequency: Daily
Amount: 4.0, Year: 2024, Month: 5, Day: 13, Description: random expense , frequency: Daily
Amount: 4.0, Year: 2024, Month: 5, Day: 13, Description: random expense , frequency: Daily
Amount: 4.0, Year: 2024, Month: 5, Day: 13, Description: random expense , frequency: Daily
Amount: 4.0, Year: 2024, Month: 5, Day: 13, Description: random expense , frequency: Daily
Amount: 5.0, Year: 2024, Month: 11, Day: 2, Description: random expense , frequency: Daily
Amount: 5.0, Year: 2024, Month: 11, Day: 2, Description: random expense , frequency: Daily
Amount: 5.0, Year: 2024, Month: 11, Day: 2, Description: random expense , frequency: Daily
Amount: 5.0, Year: 2024, Month: 11, Day: 2, Description: random expense , frequency: Daily
```

8) Clearing Data Structures

a) enter number 8

```
----------------------------------------
Menu:
1. Generate Expenses
2. Add Expense
3. Delete Expense
4. Update Expense
5. Sort expenses
6. Search Expenses
7. Print Expenses
8. Clear Data Structures
9. Exit
----------------------------------------
Choose an option: 8
```

b) all data from all of the data structures would be cleared

```
clearing the Data Structures
All arraylist content has been deleted
All Hashmpap content has been deleted
All content of Stack has been deeletd
All linkedlist content has been deleted
All content inside of the Avl tree has been deleted
----------------------------------------
```

9. Exiting the program

a) enter the number 9

```
Menu:
1. Generate Expenses
2. Add Expense
3. Delete Expense
4. Update Expense
5. Sort expenses
6. Search Expenses
7. Print Expenses
8. Clear Data Structures
9. Exit
----------------------------------------
Choose an option: 9
```

b) program would close

```
Exiting the program. Goodbye!
```