Heap sorting

package com.company;  
class HeapSort {  
 public static void main(String[] args) {  
 int arr[] = {8 ,6 ,4, 9,23,7};  
 HeapSort hs = new HeapSort();  
 hs.sort(arr);  
 hs.printarr(arr);  
 }  
  
  
 void sort(int[] arr) {  
 int leng = arr.length;  
 for (int i = leng / 2 - 1; i >= 0; i--)  
 {  
 heapify(arr, leng, i);  
 }  
 for (int i =leng-1;i>= 0; i--)  
 {  
 int temp = arr[0];  
 arr[0] = arr[i];  
 arr[i] = temp;  
 heapify(arr,i,0);  
 }  
  
 }  
  
 void heapify(int[] arr, int n, int i) {  
 int largest = i;  
 int l = 2 \* i + 1;  
 int r = 2 \* i + 2;  
 if (l < n && arr[l] > arr[largest]) {  
 largest = l;  
  
 }  
 if (r < n && arr[r] > arr[largest]) {  
 largest = r;  
 }  
 if (largest != i) {  
 int temp;  
 temp = arr[i];  
 arr[i] = arr[largest];  
 arr[largest] = temp;  
 heapify(arr,n, largest);  
  
  
 }  
  
 }  
  
 void printarr(int[] arr) {  
 for (int i = 0; i < arr.length; i++) {  
 System.*out*.println(arr[i]);  
 }  
  
  
 }  
}

A screenshot of a cell phone

Description automatically generated

Transpose matrix

package com.company;  
import java.util.\*;  
class Main {  
  
 public static void main(String[] args) {  
 Scanner sc=new Scanner(System.*in*);  
 System.*out*.println("Enter value");  
 int n=sc.nextInt();  
 int m=sc.nextInt();  
  
 int a[][]=new int[n][m];  
  
 System.*out*.println(" 2D array values");  
 for(int i=0;i<n;i++)  
 {  
 for (int j = 0; j< m; j++)  
 {  
 a[i][j] = sc.nextInt();  
 }  
 }  
  
 for (int i = 0; i < n; i++)  
 {  
 for (int j = 0; j < m; j++)3   
 System.*out*.print(a[i][j] + " ");  
 System.*out*.print("\n");  
 }  
 System.*out*.println("Orginal matix");  
  
  
System.*out*.println("Transpose of matix");  
 for ( int i = 0; i < n; i++)  
 {  
 for (int j = 0; j < m; j++)  
 System.*out*.print(a[j][i]+ " ");  
 System.*out*.print("\n");  
 }  
 }  
}

MergeSort {

class MergeSort {  
  
 void merge(int arr[], int l, int m, int r)  
 {  
   
 int n1 = m - l + 1;  
 int n2 = r - m;  
s  
 int L[] = new int [n1];  
 int R[] = new int [n2];  
  
  
 for (int i=0; i<n1; ++i)  
 L[i] = arr[l + i];  
 for (int j=0; j<n2; ++j)  
 R[j] = arr[m + 1+ j];  
  
  
  
 int i = 0, j = 0;  
  
   
 int k = l;  
 while (i < n1 && j < n2)  
 {  
 if (L[i] <= R[j])  
 {  
 arr[k] = L[i];  
 i++;  
 }  
 else  
 {  
 arr[k] = R[j];  
 j++;  
 }  
 k++;  
 }  
  
   
 while (i < n1)  
 {  
 arr[k] = L[i];  
 i++;  
 k++;  
 }  
  
  
 while (j < n2)  
 {  
 arr[k] = R[j];  
 j++;  
 k++;  
 }  
 }  
  
  
 void sort(int arr[], int l, int r)  
 {  
 if (l < r)  
 {  
 int m = (l+r)/2;  
  
  
 sort(arr, l, m);  
 sort(arr , m+1, r);  
  
   
 merge(arr, l, m, r);  
 }  
 }  
  
  
 static void printArray(int arr[])  
 {  
 int n = arr.length;  
 for (int i=0; i<n; ++i)  
 System.*out*.print(arr[i] + " ");  
 System.*out*.println();  
 }  
  
 public static void main(String args[])  
 {

Scanner sc=new Scanner (System.*in*);  
System.*out*.println("enter size");  
int n=sc.nextInt();  
System.*out*.println("Given Array");  
int arr[] = new int[n];  
for(int i=0;i<n;i++)  
{  
 arr[i]=sc.nextInt();  
  
}  
for(int i=0;i<n;i++)  
{  
 System.*out*.println(arr[i]);  
}

System.*out*.println("Given Array");  
 *printArray*(arr);  
  
 MergeSort ob = new MergeSort();  
 ob.sort(arr, 0, arr.length-1);  
  
 System.*out*.println("\nSorted array");  
 *printArray*(arr);  
 }  
}

enter size

7

Given Array

4 7 8 9 8 6 3

4

7

8

9

8

6

3

Given Array

4 7 8 9 8 6 3

Sorted array

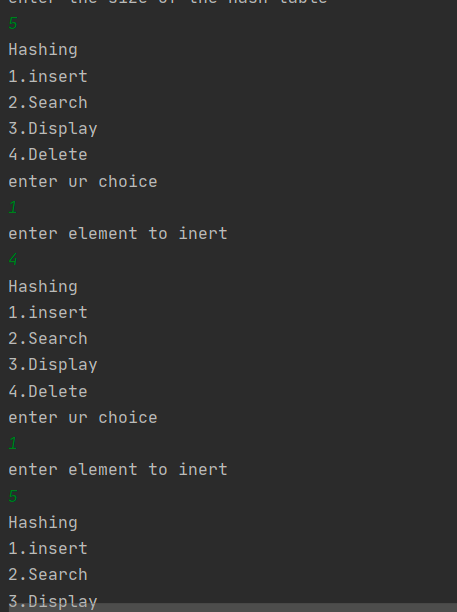
3 4 6 7 8 8 9

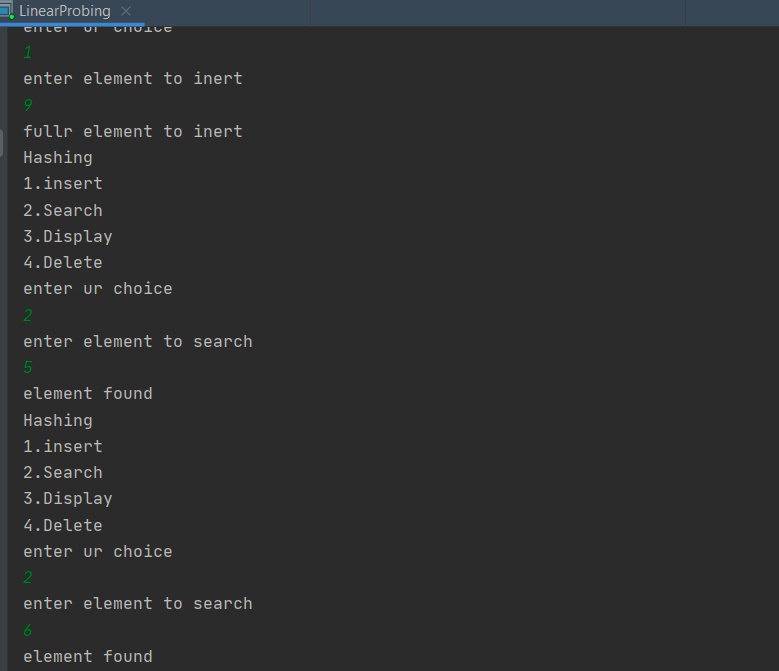
Process finished with exit code 0

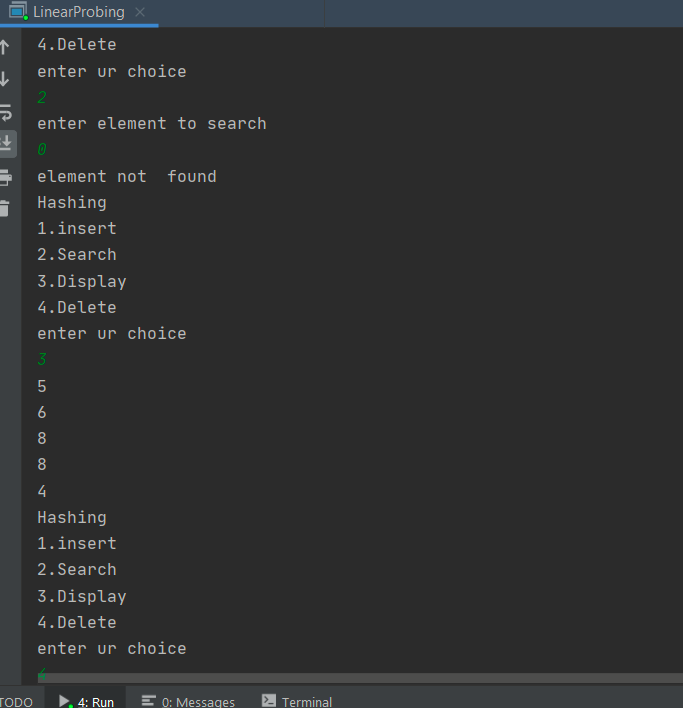
class InsertionSort {  
  
 void sort(int arr[])  
 {  
 int n = arr.length;  
 for (int i = 1; i < n; ++i) {  
 int key = arr[i];  
 int j = i - 1;  
  
  
 while (j >= 0 && arr[j] > key) {  
 arr[j + 1] = arr[j];  
 j = j - 1;  
 }  
 arr[j + 1] = key;  
 }  
 }  
  
 static void printArray(int arr[])  
 {  
 int n = arr.length;  
 for (int i = 0; i < n; ++i)  
 System.*out*.print(arr[i] + " ");  
  
 System.*out*.println();  
 }  
  
  
 public static void main(String args[])  
 {  
 int arr[]={6,7,4,2,1};  
 InsertionSort ob = new InsertionSort();  
 ob.sort(arr);  
  
 *printArray*(arr);  
 }

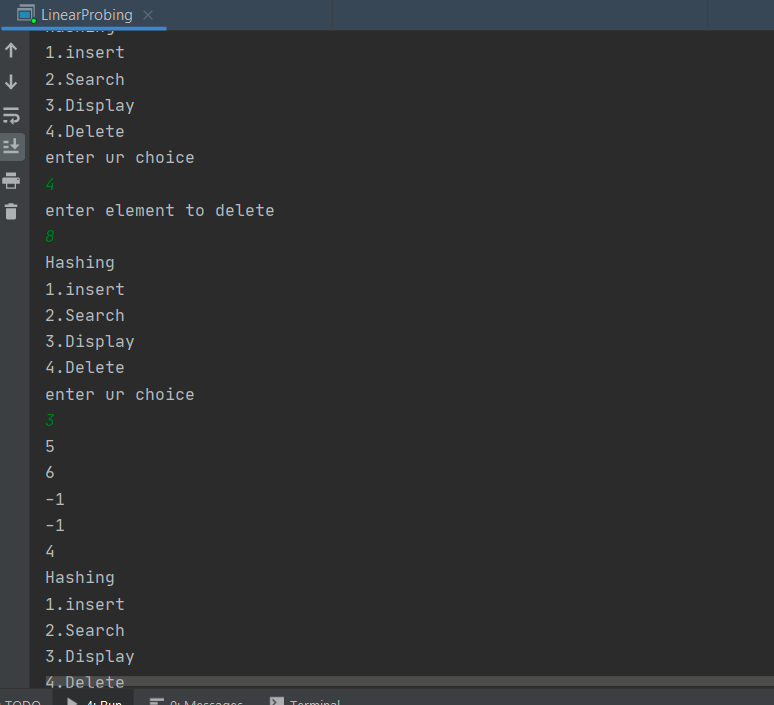
**HASHING TECHNIQUES**

**Class LinearProbing {**  
 **static int *a*[], *n*, *c* = 0;**  
  
 **public static void main(String args[]) {**  
 **int ch;**  
 **System.*out*.println("enter the size of the hash table");**  
 **Scanner sc = new Scanner(System.*in*);**  
 ***n* = sc.nextInt();**  
 ***a* = new int[*n*];**  
 **for (int i = 0; i < *n*; i++) {**  
 ***a*[i] = -1;**  
 **}**  
 **do {**  
 **System.*out*.println("Hashing");**  
 **System.*out*.println("1.insert");**  
 **System.*out*.println("2.Search");**  
 **System.*out*.println("3.Display");**  
 **System.*out*.println("4.Delete");**  
 **System.*out*.println("enter ur choice");**  
 **ch = sc.nextInt();**  
 **switch (ch) {**  
 **case 1:**  
 ***insert*();**  
 ***c*++;**  
  
 **break;**  
 **case 2:**  
 ***Search*();**  
 **break;**  
 **case 3:**  
 ***display*();**  
 **break;**  
 **case 4:**  
 ***delete*();**  
 **break;**  
  
 **case 5:**  
 **System.*exit*(0);**  
 **}**  
 **} while (ch != 5);**  
 **}**  
  
 **public static void insert() {**  
 **Scanner sc = new Scanner(System.*in*);**  
 **int index;int i;**  
 **System.*out*.println("enter element to inert");**  
 **int x = sc.nextInt();**  
 **int key = x % *n*;**  
 **for (i = 0; i < *n*; i++)**  
 **{**  
 **index = (key + i) % *n*;**  
 **if (*a*[index] == -1)**  
 **{**  
 ***a*[index] = x;**  
 **break;**  
 **}**  
 **}**  
 **if (i >= *n*) {**  
 **System.*out*.println("fullr element to inert");**  
  
  
 **}**  
  
 **}**  
  
 **public static void Search() {**  
 **int key, index;**  
 **int f=1;**  
 **Scanner sc = new Scanner(System.*in*);**  
 **System.*out*.println("enter element to search");**  
 **int x = sc.nextInt();**  
 **key = x % *n*;**  
 **for (int i = 0; i < *n*; i++) {**  
 **index = (key + i) % *n*;**  
 **if (*a*[index] == x) {**  
 **f = 0;**  
  
 **}**  
 **}**  
 **if (f == 0) {**  
 **System.*out*.println("element found");**  
 **} else**  
 **System.*out*.println("element not found");**  
 **}**  
  
 **public static void delete()**  
 **{**  
 **int key, f, index;**  
 **Scanner sc = new Scanner(System.*in*);**  
 **System.*out*.println("enter element to delete");**  
 **int x = sc.nextInt();**  
 **key = x % *n*;**  
 **for (int i = 0; i < *n*; i++) {**  
 **index = (key + i) % *n*;**  
 **if (*a*[index] == x) {**  
 ***a*[index] = -1;**  
 **}**  
 **}**  
 **}**  
 **public static void display ()**  
 **{**  
 **for (int i = 0; i < *n*; i++)**  
 **{**  
 **System.*out*.println(*a*[i]);**  
 **}**  
 **}**  
  
 **}**







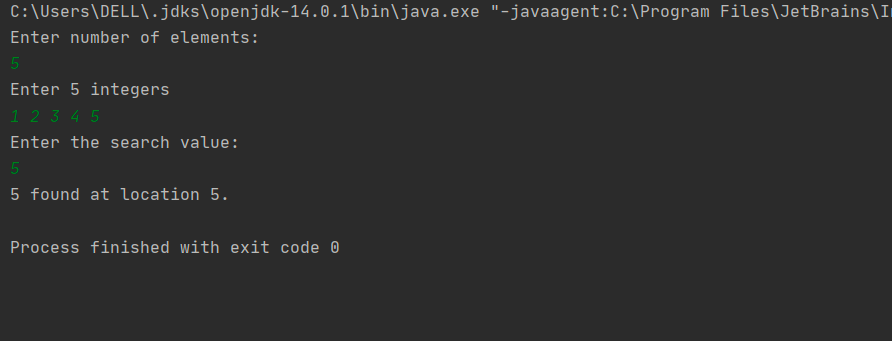


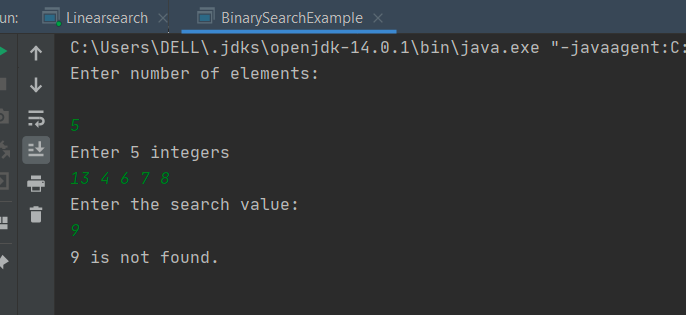
class Double probing {  
 static int *a*[], *n*, *c* = 0;  
  
 public static void main(String args[]) {  
 int ch;  
 System.*out*.println("enter the size of the hash table");  
 Scanner sc = new Scanner(System.*in*);  
 *n* = sc.nextInt();  
 *a* = new int[*n*];  
 for (int i = 0; i < *n*; i++) {  
 *a*[i] = -1;  
 }  
 do {  
 System.*out*.println("Hashing");  
 System.*out*.println("1.insert");  
 System.*out*.println("2.Search");  
 System.*out*.println("3.Display");  
 System.*out*.println("4.Delete");  
 System.*out*.println("enter ur choice");  
 ch = sc.nextInt();  
 switch (ch) {  
 case 1:  
 *insert*();  
 *c*++;  
  
 break;  
 case 2:  
 *Search*();  
 break;  
 case 3:  
 *display*();  
 break;  
 case 4:  
 *delete*();  
 break;  
  
 case 5:  
 System.*exit*(0);  
 }  
 } while (ch != 5);  
 }  
  
 public static void insert() {  
 Scanner sc = new Scanner(System.*in*);  
 int index;  
 int i;  
 System.*out*.println("enter element to inert");  
 int x = sc.nextInt();  
 int key = x % *n*;  
 for (i = 0; i < *n*; i++) {  
 index = (key + i \* i) % *n*;  
 if (*a*[index] == -1) {  
 *a*[index] = x;  
 break;  
 }  
 }  
 if (i >= *n*) {  
 System.*out*.println("fullr element to inert");  
  
  
 }  
  
 }  
  
 public static void Search() {  
 int key, index;  
 int f = 1;  
 Scanner sc = new Scanner(System.*in*);  
 System.*out*.println("enter element to search");  
 int x = sc.nextInt();  
 key = x % *n*;  
 for (int i = 0; i < *n*; i++) {  
 index = (key + i \* i) % *n*;  
 if (*a*[index] == x) {  
 f = 0;  
  
 }  
 }  
 if (f == 0) {  
 System.*out*.println("element found");  
 } else  
 System.*out*.println("element not found");  
 }  
  
 public static void delete() {  
 int key, f, index;  
 Scanner sc = new Scanner(System.*in*);  
 System.*out*.println("enter element to delete");  
 int x = sc.nextInt();  
 key = x % *n*;  
 for (int i = 0; i < *n*; i++) {  
 index = (key + i \* i) % *n*;  
 if (*a*[index] == x) {  
 *a*[index] = -1;  
 }  
 }  
 }  
  
 public static void display() {  
 for (int i = 0; i < *n*; i++) {  
 System.*out*.println(*a*[i]);  
 }  
 }  
}

**same input as linear probing….**

mport java.util.Scanner;  
 class Linearsearch  
{  
 public static void main(String[]args)  
 {  
 Scanner sc=new Scanner(System.*in*);  
 System.*out*.println("enter size of array");  
 int n=sc.nextInt();  
 int i;  
 System.*out*.println("enter array");  
 int []a=new int[n];  
 for( i=0;i<n;i++)  
 {  
 a[i]=sc.nextInt();  
 }  
  
 System.*out*.println("enter element to search");  
 int x=sc.nextInt();  
 for(i=0;i<n;i++)  
 {  
 if (a[i] == x) {  
 System.*out*.println("element foundat position" + (i + 1));  
 break;  
 }  
 }  
 if(n==i)  
 {  
 System.*out*.println("element not found");  
 }  
 }  
 }

import java.util.Scanner;  
  
  
class **BinarySearchExample**  
{  
 public static void main(String args[])  
  
 {  
  
 int counter, num, item, array[], first, last, middle;  
  
  
 Scanner input = new Scanner(System.*in*);  
  
 System.*out*.println("Enter number of elements:");  
  
 num = input.nextInt();  
  
  
  
 array = new int[num];  
  
 System.*out*.println("Enter " + num + " integers");  
  
  
 for (counter = 0; counter < num; counter++)  
  
 array[counter] = input.nextInt();  
  
 System.*out*.println("Enter the search value:");  
  
 item = input.nextInt();  
  
 first = 0;  
  
 last = num - 1;  
  
 middle = (first + last)/2;  
  
 while( first <= last )  
  
 {  
  
 if ( array[middle] < item )  
  
 first = middle + 1;  
  
 else if ( array[middle] == item )  
  
 {  
  
 System.*out*.println(item + " found at location " + (middle + 1) + ".");  
  
 break;  
  
 }  
  
 else  
  
 {  
  
 last = middle - 1;  
  
 }  
  
 middle = (first + last)/2;  
  
 }  
  
 if ( first > last )  
  
 System.*out*.println(item + " is not found.\n");  
  
 }  
  
 }





**Data binary search example:**

import java.util.\*;  
class node{  
 node left, right;  
 int data;  
}  
class Binarytree{  
 static Scanner *sc*=new Scanner(System.*in*);  
  
 static node create()  
 {  
 node root=new node();  
 System.*out*.println("Reference for the root " + root);  
 System.*out*.println("enter current node data:");  
 root.data=*sc*.nextInt();  
 System.*out*.println("is there left child to "+root.data);  
 char ch;  
 ch=*sc*.next().charAt(0);  
 if(ch=='y'||ch=='Y') {  
 root.left=*create*();  
 System.*out*.println("Reference for the left child " + root.left);}  
 else  
 root.left=null;  
 System.*out*.println("is there right child to "+root.data);  
 ch=*sc*.next().charAt(0);  
 if(ch=='y'||ch=='Y') {  
 root.right=*create*();  
 System.*out*.println("Reference for the left child " + root.right); }  
 else  
 root.right=null;  
  
 return root;  
 }  
 static void inorder(node root)  
 {  
 if(root!=null)  
 {  
 *inorder*(root.left);  
 System.*out*.print(root.data+" ");  
 *inorder*(root.right); }}  
 static void preorder(node root)  
 {  
 if(root!=null)  
 {  
 System.*out*.print(root.data+" ");  
 *preorder*(root.left);  
 *preorder*(root.right);  
  
 }  
 }  
 static void postorder(node root)  
 {  
 if(root!=null)  
 {  
 *postorder*(root.left);  
 *postorder*(root.right);  
 System.*out*.print(root.data+" ");  
  
  
 }  
 }  
 public static void main(String args[])  
 {  
 node root;  
 root=*create*();  
 *inorder*(root);  
 System.*out*.println();  
 *preorder*(root);  
 System.*out*.println();  
 *postorder*(root);  
  
 }  
}

enter current node data:

7

is there left child to 7

y

Reference for the root node@5197848c

enter current node data:

5

is there left child to 5

y

Reference for the root node@17f052a3

enter current node data:

9

is there left child to 9

n

is there right child to 9

n

Reference for the left child node@17f052a3

is there right child to 5

7

Reference for the left child node@5197848c

is there right child to 7

12

Reference for the left child node@1d56ce6a

is there right child to 7

n

9 5 7 7

5 7 97

9 5 7 7 n

Process finished with exit code 0

Sum of all nodes

public int calculateSum(Node temp){  
 int sum, sumLeft, sumRight;  
 sum = sumRight = sumLeft = 0;  
  
 if(*root* == null) {  
 System.*out*.println("Tree is empty");  
 return 0;  
 }  
 else {  
  
 if(temp.left != null)  
 sumLeft = calculateSum(temp.left);  
  
  
 if(temp.right != null)  
 sumRight = calculateSum(temp.right);  
  
 sum = temp.key+ sumLeft + sumRight;  
 return sum;  
 }  
}

do u want to continue

n

Sum of all nodes of binary tree: 55

Smallest element   
int minvalue(Node node) {  
 Node current = node;  
  
 while (current.left != null) {  
 current = current.left;  
 }  
 return (current.key);  
}

enter current node data:

10

do u want to continue

y

enter current node data:

7

do u want to continue

y

enter current node data:

8

do u want to continue

y

enter current node data:

9

do u want to continue

y

enter current node data:

5

do u want to continue

y

enter current node data:

3

do u want to continue

y

enter current node data:

6

do u want to continue

n

Minimum value of BST is 3

Largest element in bst

int minvalue(Node root) {  
 Node current = root;  
  
 while (current.right != null) {  
 current = current.right;  
 }  
 return (current.key);  
}

enter current node data:

4

do u want to continue

7

Maximum value of BST is 10

Count number of leaf nodes:

static void Countleafs(node root) {  
 int count = 0;  
  
 if (root != null) {  
 if (root.right == null && root.left == null) {  
 count++;  
 System.*out*.println(count);  
  
 } else {  
 if (root.left != null) {  
 *Countleafs*(root.left);  
  
 }  
 if (root.right != null) {  
 *Countleafs*(root.right);  
  
 }  
 }  
  
 }

mport java.util.\*;  
class BinarySearchtree  
{  
 class Node  
 {  
 int key;  
 Node left,right;  
 public Node(int data)  
 {  
 key =data;  
 left=right=null;  
 }  
 }  
 Node root;  
 BinarySearchtree()  
 {  
 root=null;  
 }  
 void insert(int key) {  
 root = insertRec(root, key);  
 }  
  
 /\* A recursive function to insert a new key in BST \*/  
 Node insertRec(Node root, int key) {  
  
 /\* If the tree is empty, return a new node \*/  
 if (root == null) {  
 root = new Node(key);  
 return root;  
 }  
 /\* Otherwise, recur down the tree \*/  
 if (key < root.key)  
 root.left=insertRec(root.left,key);  
 else if (key > root.key)  
 root.right = insertRec(root.right, key);  
 else if(key ==root.key)  
 System.*out*.println("Duplicate values are not allowed");  
  
 return root;  
 }  
  
 // This method mainly calls InorderRec()  
 void inorder() {  
 inorderRec(root);  
 }  
  
 // A utility function to do inorder traversal of BST  
 void inorderRec(Node root) {  
 if (root != null) {  
 inorderRec(root.left);  
 System.*out*.println(root.key);  
 inorderRec(root.right);  
 }  
 }  
 public static void main(String[] args) {  
 BinarySearchtree tree = new BinarySearchtree();  
 char ch;  
 Scanner sc=new Scanner(System.*in*);  
 do  
 {  
 System.*out*.println("enter current node data:");  
 tree.insert(sc.nextInt());  
 System.*out*.println(" do u want to continue");  
 ch=sc.next().charAt(0);  
 }while(ch=='y'||ch=='Y');  
  
 tree.inorder();  
 }  
}

C:\Users\DELL\.jdks\openjdk-14.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2020.1.1\lib\idea\_rt.jar=60951:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2020.1.1\bin" -Dfile.encoding=UTF-8 -classpath G:\javanew\out\production\javanew BinarySearchtree

enter current node data:

11

do u want to continue

y

enter current node data:

12

do u want to continue

y

enter current node data:

6

do u want to continue

8

6

11

12

Process finished with exit code 0

Queues

import java.util.\*;  
class Queus  
{  
 static int *f*,*r*,*ch*,*n*;  
 static int[] *Q*=new int[*n*];  
 public static void main(String args []){  
 *f*=-1;  
 *r*=-1;  
 Scanner sc=new Scanner(System.*in*);  
 System.*out*.println("enter size if queus");  
 int n=sc.nextInt();  
 do {  
 System.*out*.println("QUESUES implemenation");  
 System.*out*.println("1.enques");  
 System.*out*.println("2.deques");  
 System.*out*.println("3.Display");  
 int ch=sc.nextInt();  
 switch(ch)  
 {  
 case 1:  
 *enques*();  
 break;  
 case 2:  
 *deques*();  
 break;  
 case 3:  
 *Display*();  
 break;  
  
 }  
 }  
 while(*ch*!=3);  
 }  
 public static void enques()  
 {  
 if(*r*==*n*-1)  
  
 System.*out*.println("ques is full");  
  
 else if(*r*==-1&&*f*==-1)  
 {  
 *f*++;  
 *r*++;  
 Scanner sc=new Scanner(System.*in*);  
 int item=sc.nextInt();  
 *Q*[*r*]=item;  
 }  
 else  
 {  
 *r*++;  
 Scanner sc=new Scanner(System.*in*);  
 int item=sc.nextInt();  
 *Q*[*r*]=item;  
 }  
 }  
 public static void deques()  
 {  
 if(*r*==-1)  
  
 System.*out*.println("ques is empty");  
  
 else if(*r*==*f*)  
 {  
 *f*=*r*=-1;  
 }  
 else  
 {  
 System.*out*.println("enter element to delete" + *Q*[*r*]);  
 *f*++;  
 }  
  
  
 }  
 public static void Display()  
 {  
 for(int i=0;i<*r*;i++)  
 {  
 System.*out*.println(*Q*[i]);  
 }  
 }  
}

C:\Users\DELL\.jdks\openjdk-14.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2020.1.1\lib\idea\_rt.jar=61087:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2020.1.1\bin" -Dfile.encoding=UTF-8 -classpath G:\javanew\out\production\javanew Queus

QUESUES implemenation

1.enques

2.deques

3.Display

enter your choice

1

enter element to insert

12

QUESUES implemenation

1.enques

2.deques

3.Display

enter your choice

1

enter element to insert

13

QUESUES implemenation

1.enques

2.deques

3.Display

enter your choice

1

enter element to insert

14

QUESUES implemenation

1.enques

2.deques

3.Display

enter your choice

1

enter element to insert

15

QUESUES implemenation

1.enques

2.deques

3.Display

enter your choice

1

enter element to insert

16

QUESUES implemenation

1.enques

2.deques

3.Display

enter your choice

1

ques is full

QUESUES implemenation

1.enques

2.deques

3.Display

enter your choice

3

12

13

14

15

16

QUESUES implemenation

1.enques

2.deques

3.Display

enter your choice

2

enter element to delete16

QUESUES implemenation

1.enques

2.deques

3.Display

enter your choice

3

12

13

14

15

16

QUESUES implemenation

1.enques

2.deques

3.Display

enter your choice

Stacks

import java.util.\*;  
class Stack  
{  
 static int *top*,*ch*,*n*;  
 static int[] *stack*=new int[5];  
 public static void main(String args []){  
 *top*=-1;  
 Scanner sc=new Scanner(System.*in*);  
 int n;  
 n=5;  
 do {  
 System.*out*.println("Stack implemenation");  
 System.*out*.println("1.INSERTION(push)");  
 System.*out*.println("2.Deletion(pop)");  
 System.*out*.println("3.Display");  
 System.*out*.println("enter your choice");  
  
 int ch=sc.nextInt();  
 switch(ch)  
 {  
 case 1:  
 *push*();  
 break;  
 case 2:  
 *pop*();  
 break;  
 case 3:  
 *Display*();  
 break;  
 case 4:  
 System.*out*.println("exit");  
 break;  
  
  
 }  
 }  
 while(*ch*!=4);  
 }  
 static void push()  
 {  
 if(*top*==4)  
 System.*out*.println("stack is full");  
  
 else  
 {  
  
 *top*++;  
 Scanner sc=new Scanner(System.*in*);  
 System.*out*.println("enter element to insert");  
 int item=sc.nextInt();  
 *stack*[*top*]=item;  
 }  
  
 }  
 static void pop()  
 {  
 if(*top*==-1)  
  
 System.*out*.println("stack is empty");  
 else  
 {  
 System.*out*.println("enter element to delete" + *stack*[*top*]);  
 *top*-- ;  
 }  
  
  
 }  
 static void Display()  
 {  
 for(int i=0;i<=4;i++)  
 {  
 System.*out*.println(*stack*[i]);  
 }  
 }  
}

tack implemenation

1.INSERTION(push)

2.Deletion(pop)

3.Display

enter your choice

1

enter element to insert

12

Stack implemenation

1.INSERTION(push)

2.Deletion(pop)

3.Display

enter your choice

1

enter element to insert

13

Stack implemenation

1.INSERTION(push)

2.Deletion(pop)

3.Display

enter your choice

1

enter element to insert

14

Stack implemenation

1.INSERTION(push)

2.Deletion(pop)

3.Display

enter your choice

1

enter element to insert

15

Stack implemenation

1.INSERTION(push)

2.Deletion(pop)

3.Display

enter your choice

1

enter element to insert

16

Stack implemenation

1.INSERTION(push)

2.Deletion(pop)

3.Display

enter your choice

1

stack is full

Stack implemenation

1.INSERTION(push)

2.Deletion(pop)

3.Display

enter your choice

3

12

13

14

15

16

Stack implemenation

1.INSERTION(push)

2.Deletion(pop)

3.Display

enter your choice

2

enter element to delete16

Stack implemenation

1.INSERTION(push)

2.Deletion(pop)

3.Display

enter your choice

3

12

13

14

15

16

Stack implemenation

1.INSERTION(push)

2.Deletion(pop)

3.Display

Single linked creation and display

import java.util.Scanner;  
class Node{  
 int data;  
 Node next;  
}  
class Sll {  
 static Scanner *sc* = new Scanner(System.*in*);  
 static Node *head*, *tail* = null;  
  
 static void create() {  
 System.*out*.println("enter size of linked list ");  
 int n = *sc*.nextInt();  
 for (int i = 0; i < n; i++)  
 {  
 Node temp = new Node();  
 System.*out*.println("enter data");  
 int data = *sc*.nextInt();  
 temp.data = data;  
 temp.next = null;  
 if (*head* == null) {  
 *head* = temp;  
 *tail* = temp;  
 } else {  
 *tail*.next = temp;  
 *tail* = temp;  
 }  
 }  
 }  
  
 static void Display() {  
 if (*head* == null) {  
 System.*out*.println("list is empty");  
 } else {  
 Node temp;  
 temp = *head*;  
 while (temp != null) {  
 System.*out*.println(temp.data);  
 temp = temp.next;  
 }  
 System.*out*.println();  
 }  
 }  
  
 public static void main(String args[]) {  
  
 *create*();  
 *Display*();  
 }  
}

C:\Users\DELL\.jdks\openjdk-14.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrain

enter size of linked list

5

enter data

12

enter data

13

enter data

14

enter data

15

enter data

116

12

13

14

15

116

Process finished with exit code 0

INSERT AT BEGIN

static void insertatbeg()  
{  
 Node temp=new Node();  
 System.*out*.println("enter data");  
 int data=*sc*.nextInt();  
 temp.data=data;  
 temp.next=null;  
 if(*head*==null)  
 {  
 *head*=*tail*=temp;  
 }  
 else  
 {  
 temp.next=*head*;  
 *head*=temp;  
 }  
}

enter size of linked list

4

enter data

12

enter data

13

enter data

14

enter data

15

enter data

1

Display

1

12

13

14

15

deleteatbegin()  
public static void deleteatbegin()  
{  
 Node temp=*head*;  
 if(*head*!=null) {  
 *head* = *head*.next;  
 temp = null;  
 }  
 else if(*head*==*tail*)  
 {  
 *head*=*tail*=null;  
 }  
 else  
 System.*out*.println("list is empty");  
 }

enter data

11

enter data

1

enter data

12

enter data

13

Display

11

1

12

13

Deletion at begining

After linked list Deletion at beginning 11 is beem deleted here..

1

12

13.

insertatend()

public static void insertatend()  
{  
 Node temp=new Node();  
 System.*out*.println("enter data");  
 temp.data=*sc*.nextInt();  
 temp.next=null;  
 if(*head*==null)  
 {  
 *head*=*tail*=temp;  
 }  
 else  
 {  
 *tail*.next=temp;  
 *tail*=temp;  
 }

}

enter data

12

enter data

13

enter data

14

enter data

15

Display

12

13

14

15

After linked list insertion at ending

enter data here inserted …………………..

16

12

13

14

15

16

Delete at end

public static void deleteatend() {  
 Node temp=*head*;  
 if(*head*==*tail*)  
 {  
 *head*=*tail*=null;  
 }  
 else if(*head*!=*tail*)  
 {  
 while(temp.next!=*tail*)  
 {  
 temp=temp.next;  
  
 }  
 *tail*=temp;  
 *tail*.next=null;  
 }  
 else  
 System.*out*.println("list is empty");  
}

enter size of linked list

4

enter data

1

enter data

2

enter data

3

enter data

4

Display

1

2

3

4

After linked list deletion at ending

1

2

3

……………………………………

DOUBLE LINKED LIST Creation and display

import java.util.Scanner;  
class Node{  
 int data;  
 Node next;  
 Node prev;  
}  
class Dll {  
 static Scanner *sc* = new Scanner(System.*in*);  
 static Node *head*, *tail* = null;  
  
 static void create() {  
 System.*out*.println("enter size of linked list ");  
 int n = *sc*.nextInt();  
 for (int i = 0; i < n; i++)  
 {  
 Node temp = new Node();  
 System.*out*.println("enter data");  
 int data = *sc*.nextInt();  
 temp.data = data;  
 temp.next = null;  
 temp.prev=null;  
 if (*head* == null) {  
 *head* = temp;  
 *tail* = temp;  
 } else {  
 *tail*.next = temp;  
 temp.prev=*tail*;  
 *tail*=temp;  
 }  
 }  
 }  
  
 static void Display() {  
 if (*head* == null) {  
 System.*out*.println("list is empty");  
 } else {  
 Node temp;  
 temp = *head*;  
 while (temp != null) {  
 System.*out*.println(temp.data);  
 temp = temp.next;  
 }  
 System.*out*.println();  
 }  
 }  
 insert at beg()  
 static void insertatbeg()

{  
 Node temp=new Node();  
 System.*out*.println("enter data");  
 int data=*sc*.nextInt();  
 temp.data=data;  
 temp.next=null;  
 temp.prev=null;  
 if(*head*==null)  
 {  
 *head*=*tail*=temp;  
 }  
 else  
 {  
 temp.next=*head*;  
 *head*.prev=temp;  
 *head*=temp;  
 }  
 }

enter size of linked list

4

enter data

11

enter data

12

enter data

13

enter data

14

Display dll

11

12

13

14

enter data

9

9

11

12

13

14

Insert at end

public static void insertatend()  
{  
 Node temp=new Node();  
 System.*out*.println("enter data");  
 temp.data=*sc*.nextInt();  
 temp.next=null;  
 if(*head*==null)  
 {  
 *head*=*tail*=temp;  
 }  
 else  
 {  
 *tail*.next=temp;  
 temp.prev=*tail*;  
 *tail*=temp;  
 }  
  
}

Display dll

12

13

14

15

enter data

8

Display dll after insertion at end

12

13

14

15

8

deleteatbegin()

public static void deleteatbegin()  
{  
 Node temp=*head*;  
 if(*head*!=null) {  
  
 *head* = *head*.next;  
 *head*.prev=null;  
 temp = null;  
 }  
 else if(*head*==*tail*)  
 {  
 *head*=*tail*=null;  
 }  
 else  
 System.*out*.println("list is empty");  
 }

delete at end()

public static void deleteatend() {  
 Node temp=*head*;  
 if(*head*==*tail*)  
 {  
 *head*=*tail*=null;  
 }  
 else if(*head*!=*tail*)  
 {  
 while(temp.next!=*tail*)  
 {  
 temp=temp.next;  
  
 }  
 *tail*=temp;  
 *tail*.next=null;  
 }  
 else  
 System.*out*.println("list is empty");  
}

Display dll

1

2

3

4

5

6

Display dll after deletion at enddd

1

2

3

4

5

CIRCULAR LINKED LIST IN JAVA

import java.util.Scanner;  
class Node{  
 int data;  
 Node next;  
}  
class circular{  
 static Scanner *sc* = new Scanner(System.*in*);  
 static Node *head*, *tail* = null;  
  
 static void create() {  
 System.*out*.println("enter size of linked list ");  
 int n = *sc*.nextInt();  
 for (int i = 0; i < n; i++)  
 {  
 Node temp = new Node();  
 System.*out*.println("enter data");  
 int data = *sc*.nextInt();  
 temp.data = data;  
 temp.next = null;  
 if (*head* == null) {  
 *head* = temp;  
 *tail* = temp;  
 temp.next=*head*;  
 } else {  
 *tail*.next = temp;  
 *tail* = temp;  
 *tail*.next=*head*;  
 }  
 }  
 }  
  
 static void Display() {  
 if (*head* == null) {  
 System.*out*.println("list is empty");  
 } else {  
 Node temp;  
 temp = *head*;  
 do{  
 //Prints each node by incrementing pointer.  
 System.*out*.print(" "+ temp.data);  
 temp = temp.next;  
 }while(temp!= *head*);  
 System.*out*.println();  
 }  
 }  
  
  
  
 public static void main(String args[]) {  
  
 *create*();  
 *Display*();  
 }  
}

enter size of linked list

5

enter data

1

enter data

2

enter data

3

enter data

4

enter data

5

1 2 3 4 5

Process finished with exit code 0