**Bubble sort (sắp xếp nổi bọt)**

public class BubbleSortExample {

static void bubbleSort(int[] arr) {

int n = arr.length;

int temp = 0;

for(int i=0; i < n; i++){

for(int j=1; j < (n-i); j++){

if(arr[j-1] > arr[j]){

//swap elements

temp = arr[j-1];

arr[j-1] = arr[j];

arr[j] = temp;

}

}

}

}

public static void main(String[] args) {

int arr[] ={3,60,35,2,45,320,5};

System.out.println("Array Before Bubble Sort");

for(int i=0; i < arr.length; i++){

System.out.print(arr[i] + " ");

}

System.out.println();

bubbleSort(arr);//sorting array elements using bubble sort

System.out.println("Array After Bubble Sort");

for(int i=0; i < arr.length; i++){

System.out.print(arr[i] + " ");

}

}

}

**Selection sort (sắp xếp chọn)**

**(Cách 1: )**

public class SelectionSortExample {

public static void selectionSort(int[] arr){

for (int i = 0; i < arr.length - 1; i++)

{

int index = i;

for (int j = i + 1; j < arr.length; j++){

if (arr[j] < arr[index]){

index = j;//searching for lowest index

}

}

int smallerNumber = arr[index];

arr[index] = arr[i];

arr[i] = smallerNumber;

}

}

public static void main(String a[]){

int[] arr1 = {9,14,3,2,43,11,58,22};

System.out.println("Before Selection Sort");

for(int i:arr1){

System.out.print(i+" ");

}

System.out.println();

selectionSort(arr1);//sorting array using selection sort

System.out.println("After Selection Sort");

for(int i:arr1){

System.out.print(i+" ");

}

}

}

**( Cách 2: )**

import java.util.Scanner;

public class SelectionSortExample2

{

public static void main(String args[])

{

int size, i, j, temp;

int arr[] = new int[50];

Scanner scan = new Scanner(System.in);

System.out.print("Enter Array Size : ");

size = scan.nextInt();

System.out.print("Enter Array Elements : ");

for(i=0; i<size; i++)

{

arr[i] = scan.nextInt();

}

System.out.print("Sorting Array using Selection Sort Technique..\n");

for(i=0; i<size; i++)

{

for(j=i+1; j<size; j++)

{

if(arr[i] > arr[j])

{

temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

}

System.out.print("Now the Array after Sorting is :\n");

for(i=0; i<size; i++)

{

System.out.print(arr[i]+ " ");

}

}

}

**Insertion sort (sắp xếp chèn)**

public class InsertionSortExample {

public static void insertionSort(int array[]) {

int n = array.length;

for (int j = 1; j < n; j++) {

int key = array[j];

int i = j-1;

while ( (i > -1) && ( array [i] > key ) ) {

array [i+1] = array [i];

i--;

}

array[i+1] = key;

}

}

public static void main(String a[]){

int[] arr1 = {9,14,3,2,43,11,58,22};

System.out.println("Before Insertion Sort");

for(int i:arr1){

System.out.print(i+" ");

}

System.out.println();

insertionSort(arr1);//sorting array using insertion sort

System.out.println("After Insertion Sort");

for(int i:arr1){

System.out.print(i+" ");

}

}

}

**Quick sort (cải tiến từ phương pháp đổi chỗ)**

**(Cách 1: )**

// Java program for implementation of QuickSort

class QuickSort

{

int partition(int arr[], int low, int high)

{

int pivot = arr[high];

int i = (low-1); // index of smaller element

for (int j=low; j<high; j++)

{

// If current element is smaller than or

// equal to pivot

if (arr[j] <= pivot)

{

i++;

// swap arr[i] and arr[j]

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

// swap arr[i+1] and arr[high] (or pivot)

int temp = arr[i+1];

arr[i+1] = arr[high];

arr[high] = temp;

return i+1;

}

void sort(int arr[], int low, int high)

{

if (low < high)

{

/\* pi is partitioning index, arr[pi] is

now at right place \*/

int pi = partition(arr, low, high);

// Recursively sort elements before

// partition and after partition

sort(arr, low, pi-1);

sort(arr, pi+1, high);

}

}

/\* A utility function to print array of size n \*/

static void printArray(int arr[])

{

int n = arr.length;

for (int i=0; i<n; ++i)

System.out.print(arr[i]+" ");

System.out.println();

}

// Driver program

public static void main(String args[])

{

int arr[] = {10, 7, 8, 9, 1, 5};

int n = arr.length;

QuickSort ob = new QuickSort();

ob.sort(arr, 0, n-1);

System.out.println("sorted array");

printArray(arr);

}

}

**(Cách 2:)**

package quickSort;

import java.util.Scanner;

class arrList {

private int a[];

private int n;

public arrList(int size) {

a = new int[size];

n = 0;

}

public void add(int x) {

a[n++] = x;

}

public void set(int n, int i) {

a[i] = n;

}

public int get(int i) {

return a[i];

}

public int nOfItem() {

return n;

}

public void quickSort() {

sort(0, n - 1);

}

private void sort(int l, int r) { // khai bao 2 bien left and right

int i, j;

int x, t;

i = l;

j = r;

x = a[(l + r) / 2];

do {

while (a[i] < x)

i++;

while (a[j] > x)

j--;

if (i <= j) {

t = a[i];

a[i] = a[j];

a[j] = t;

i++;

j--;

}

} while (i <= j); // left <= right

if (l < j)

sort(l, j);

if (i < r)

sort(i, r);

}

}

public class quicksort {

static Scanner sc = new Scanner(System.in);

static void nhap(arrList r) {

int x;

System.out.print("(0 de exit) moi nhap phan tu: ");

do {

x = sc.nextInt();

if (x != 0) {

r.add(x);

}

} while (x != 0);

}

static void xuat(arrList r) {

for (int i = 0; i < r.nOfItem(); i++) {

System.out.print(r.get(i) + " ");

}

System.out.println();

}

public static void main(String[] args) {

arrList r = new arrList(1000);

int chon;

do {

System.out.print("0.thoat");

System.out.print("\n1.nhap");

System.out.print("\n2.xuat");

System.out.print("\n3. sap xep kieu quickSort");

System.out.print("\n----------------------");

System.out.print("\n chon:");

chon = sc.nextInt();

switch (chon) {

case 1:

nhap(r);

break;

case 2:

xuat(r);

break;

case 3:

r.quickSort();

break;

default:

break;

}

} while (chon != 0);

}

}

**Heap sort (sơ đồ cây)**

// Java program for implementation of Heap Sort

public class HeapSort

{

public void sort(int arr[])

{

int n = arr.length;

// Build heap (rearrange array)

for (int i = n / 2 - 1; i >= 0; i--)

heapify(arr, n, i);

// One by one extract an element from heap

for (int i=n-1; i>=0; i--)

{

// Move current root to end

int temp = arr[0];

arr[0] = arr[i];

arr[i] = temp;

// call max heapify on the reduced heap

heapify(arr, i, 0);

}

}

// To heapify a subtree rooted with node i which is

// an index in arr[]. n is size of heap

void heapify(int arr[], int n, int i)

{

int largest = i; // Initialize largest as root

int l = 2\*i + 1; // left = 2\*i + 1

int r = 2\*i + 2; // right = 2\*i + 2

// If left child is larger than root

if (l < n && arr[l] > arr[largest])

largest = l;

// If right child is larger than largest so far

if (r < n && arr[r] > arr[largest])

largest = r;

// If largest is not root

if (largest != i)

{

int swap = arr[i];

arr[i] = arr[largest];

arr[largest] = swap;

// Recursively heapify the affected sub-tree

heapify(arr, n, largest);

}

}

/\* A utility function to print array of size n \*/

static void printArray(int arr[])

{

int n = arr.length;

for (int i=0; i<n; ++i)

System.out.print(arr[i]+" ");

System.out.println();

}

// Driver program

public static void main(String args[])

{

int arr[] = {12, 11, 13, 5, 6, 7};

int n = arr.length;

HeapSort ob = new HeapSort();

ob.sort(arr);

System.out.println("Sorted array is");

printArray(arr);

}

}

**ShakerSort (sàng qua sàng lại)**

public void shakersort() {

int j, l, r, k;

int temp;

l = k = 0;

r = n - 1;

do {

for (j = r; j > l; j--) {

if (a[j] < a[j - 1]) {

temp = a[j];

a[j] = a[j - 1];

a[j - 1] = temp;

k = j;

}

}

l = k;

for (j = l; j < r; j++) {

if (a[j] > a[j + 1]) {

temp = a[j];

a[j] = a[j + 1];

a[j + 1] = temp;

k = j;

}

}

r = k;

} while (l < r);

}

**Bài tập tự làm về bubbleSort**

package BubbleSort;

import java.util.Scanner;

class arrList {

private int[] a;

int n;

public arrList(int size) {

a = new int[size];

n = 0;

}

public void add(int x) {

a[n++] = x;

}

public void set(int x, int i) {

a[i] = x;

}

public int get(int i) {

return a[i];

}

// độ dài mảng n

public int nOfItem() {

return n;

}

public void bubblesort() {

for (int i = nOfItem() - 1; i >= 1; i--) {

boolean check = true;

for (int j = 0; j < i; j++) {

if (a[j] > a[j + 1]) {

int temp;

temp = a[j];

a[j] = a[j + 1];

a[j + 1] = temp;

check = false;

}

}

if (check)

break;

}

}

}

public class Run {

static Scanner sc = new Scanner(System.in);

// nhap mang sap xep

static void nhapMang(arrList a) {

int x;

do {

x = sc.nextInt();

if(x>0) {

a.add(x);

}

}while(x>0);

}

// xuat mang sap xep

static void xuatMang(arrList a) {

for(int i=0;i< a.nOfItem();i++) {

System.out.print(a.get(i) + " ");

}

System.out.println();

}

public static void main(String[] args) {

arrList a = new arrList(1000);

int chon=0;

do {

System.out.print("1. nhap");

System.out.print("\n2. xuat");

System.out.print("\n3. sap xep");

System.out.print("\nChon: ");

chon = sc.nextInt();

switch (chon) {

case 1: {

System.out.print("nhap day so ( 0 ket thuc ): ");

nhapMang(a);

break;

}

case 2:

xuatMang(a);

break;

case 3:

a.buddlesort();;

break;

default:

throw new IllegalArgumentException("Unexpected value: " + chon);

}

}while(chon>0);

}

}

**Bài tập tự làm về insertSort**

package InsertionSort;

import java.util.ArrayList;

import java.util.Scanner;

class ArrList {

private int[] a;

private int n;

public ArrList(int size) {

a = new int[size];

n = 0;

}

// x la so nhap vao

public void add(int x) {

a[n++] = x;

}

public void set(int x, int i) {

a[i] = x;

}

public int get(int i) {

return a[i];

}

// phan tu n toi da

public int nOfItem() {

return n;

}

public void insertionSort() {

int i, j, x;

for (i = 1; i < n; i++) {

x = a[i];

j = i - 1;

while (j >= 0 && x < a[j]) {

a[j + 1] = a[j];

j--;

}

a[j + 1] = x;

}

}

}

public class Run {

static Scanner sc = new Scanner(System.in);

// nhap phan tu

static void nhap(ArrList a) {

int x;

do {

x = sc.nextInt();

if (x > 0)

a.add(x);

} while (x > 0);

}

// xuat phan tu

static void xuat(ArrList a) {

for (int i = 0; i < a.nOfItem(); i++) {

System.out.print(a.get(i) + " ");

}

System.out.println();

}

public static void main(String[] args) {

ArrList ar = new ArrList(1000);

int chon;

do {

System.out.print("1. nhap");

System.out.print("\n2. xuat");

System.out.print("\n3. sap xep");

System.out.print("\nChon: ");

chon = sc.nextInt();

switch (chon) {

case 1: {

System.out.print("nhap day so ( 0 ket thuc ): ");

nhap(ar);

break;

}

case 2:

xuat(ar);

break;

case 3:

ar.insertionSort();

break;

default:

throw new IllegalArgumentException("Unexpected value: " + chon);

}

} while (chon > 0);

}

}

**Bài tập về quản lý sinh viên**

package QuanLiSinhVien;

import java.util.\*;

class sinhVien {

private String masv;

private String hoten;

private float diem;

public sinhVien(String masv, String hoten, float diem) {

this.masv = masv;

this.hoten = hoten;

this.diem = diem;

}

public sinhVien() {

}

public String getMasv() {

return masv;

}

public void setMasv(String masv) {

this.masv = masv;

}

public String getHoten() {

return hoten;

}

public void setHoten(String hoten) {

this.hoten = hoten;

}

public float getDiem() {

return diem;

}

public void setDiem(float diem) {

this.diem = diem;

}

}

class arrayList {

private sinhVien[] a;

private int n;

public arrayList(int size) {

a = new sinhVien[size];

n = 0;

}

public void add(sinhVien x) {

a[n++] = x;

}

public void set(sinhVien x, int i) {

a[i] = x;

}

public sinhVien get(int i) {

return a[i];

}

public int nOfItem() {

return n;

}

public void insertionSort() {

int i, j;

sinhVien x;

for (i = 1; i < n; i++) {

x = a[i];

j = i - 1;

while (j >= 0 && x.getMasv().compareTo(a[j].getMasv()) < 0) {

a[j + 1] = a[j];

j--;

}

a[j + 1] = x;

}

}

}

public class SV {

static Scanner sc = new Scanner(System.in);

static void nhap(arrayList a) {

String mssv, ten;

int diem;

sinhVien x;

do {

sc.nextLine();

System.out.print("nhap ma sv: ");

mssv = sc.nextLine();

if (mssv.length() > 0) {

System.out.println("ho va ten: ");

ten = sc.nextLine();

System.out.println("diem: ");

diem = sc.nextInt();

x = new sinhVien(mssv, ten, diem);

a.add(x);

}

} while (mssv.length() > 0);

}

static sinhVien tim(arrayList a, String mssv) {

int i = 0;

while (i < a.nOfItem() && mssv.compareTo(a.get(i).getMasv()) != 0)

i++;

if (i < a.nOfItem())

return a.get(i);

return null;

}

static void xuat(arrayList a) {

for (int i = 0; i < a.nOfItem(); i++) {

sinhVien x = a.get(i);

System.out.println("------------------------------");

System.out.println("|ma sv: " + x.getMasv() + "|");

System.out.println("|ten sv: " + x.getHoten() + "|");

System.out.println("|diem: " + x.getDiem() + "|");

System.out.println("------------------------------");

}

}

public static void main(String[] args) {

arrayList a = new arrayList(1000);

int chon;

String mssv;

do {

System.out.println("0. ket thuc");

System.out.println("1. nhap thong tin");

System.out.println("2. xuat thong tin");

System.out.println("3. sap xep insertionSort");

System.out.println("4. tim theo mssv");

System.out.print("Nhap: ");

chon = sc.nextInt();

switch (chon) {

case 1:

nhap(a);

break;

case 2:

xuat(a);

break;

case 3:

a.insertionSort();

break;

case 4:

sc.nextLine();

System.out.print("mssv cần tìm: ");

mssv = sc.nextLine();

sinhVien x = tim(a, mssv);

System.out.println("sv can tim la: " + x.getMasv());

System.out.println("ten can tim la: " + x.getHoten());

System.out.println("diem can tim la:" + x.getDiem());

// tim(a, mssv);

break;

default:

throw new IllegalArgumentException("Unexpected value: " + chon);

}

} while (chon != 0);

}

}

**(Quản Lý Sinh Viên)**

package sapXepSinhVien;

import java.util.\*;

class arrList {

private sinhVien a[];

private int n;

public arrList(int size) {

a = new sinhVien[size];

n = 0;

}

public void add(sinhVien x) {

a[n++] = x;

}

public void set(sinhVien n, int i) {

a[i] = n;

}

public sinhVien get(int i) {

return a[i];

}

public int nOfItem() {

return n;

}

public void quickSort() {

sort(0, n - 1);

}

private void sort(int l, int r) { // khai bao 2 bien left and right

int i, j;

sinhVien x, t;

i = l;

j = r;

x = a[(l + r) / 2];

do {

while (a[i].getMasv().compareTo(x.getMasv()) < 0)

i++;

while (a[j].getMasv().compareTo(x.getMasv()) > 0)

j--;

if (i <= j) {

t = a[i];

a[i] = a[j];

a[j] = t;

i++;

j--;

}

} while (i <= j); // left <= right

if (l < j)

sort(l, j);

if (i < r)

sort(i, r);

}

}

class sinhVien {

private String masv;

private String hoten;

private float diem;

public sinhVien(String masv, String hoten, float diem) {

this.masv = masv;

this.hoten = hoten;

this.diem = diem;

}

public sinhVien() {

}

public String getMasv() {

return masv;

}

public void setMasv(String masv) {

this.masv = masv;

}

public String getHoten() {

return hoten;

}

public void setHoten(String hoten) {

this.hoten = hoten;

}

public float getDiem() {

return diem;

}

public void setDiem(float diem) {

this.diem = diem;

}

public void nhapsv() {

Scanner sc = new Scanner(System.in);

System.out.print("nhap ma sv:");

do {

this.masv = sc.nextLine();

System.out.print("nhap ten sv:");

this.hoten = sc.next();

System.out.print("nhap diem:");

this.diem = sc.nextFloat();

} while (diem > 10 || diem < 0);

}

public void xuatsv() {

System.out.println("ma sv:" + masv);

System.out.println("ten sv:" + hoten);

System.out.println("diem:" + diem);

}

}

public class sapxepsinhvien {

static Scanner sc = new Scanner(System.in);

static void nhap(arrList r) {

int x;

do {

sinhVien sv = new sinhVien();

sv.nhapsv();

r.add(sv);

System.out.print("bat ki de tiep tuc, 0 de dung");

x = sc.nextInt();

} while (x != 0);

}

static void xuat(arrList r) {

for (int i = 0; i < r.nOfItem(); i++) {

r.get(i).xuatsv();

}

System.out.println();

}

public static void main(String[] args) {

arrList arr = new arrList(1000);

int chon;

do {

System.out.print("0.thoat");

System.out.print("\n1.nhap");

System.out.print("\n2.xuat");

System.out.print("\n3.sap xep theo quicksort");

System.out.print("\n chon:");

chon = sc.nextInt();

switch (chon) {

case 1:

nhap(arr);

break;

case 2:

xuat(arr);

break;

case 3:

arr.quickSort();

break;

default:

break;

}

} while (chon != 0);

}

}

**Bài tập tổng hợp (Sort)**

package TongHopSort;

import java.util.\*;

class Arrlist {

private int[] a;

private int n;

public Arrlist(int size) {

a = new int[size];

n = 0;

}

public void add(int x) {

a[n++] = x;

}

public void set(int x, int i) {

a[i] = x;

}

public int get(int i) {

return a[i];

}

public int nOfItem() {

return n;

}

// insertion sort

public void insertionSort() {

int i, j, x;

for (i = 1; i < n; i++) {

x = a[i];

j = i - 1;

while (j >= 0 && x < a[j]) {

a[j + 1] = a[j];

j--;

}

a[j + 1] = x;

}

}

// selection sort

public void selectionSort() {

for (int i = 0; i < nOfItem() - 1; i++) {

int index = i;

for (int j = i + 1; j < nOfItem(); j++) {

if (a[j] < a[index]) {

index = j;

}

}

int smallerNumber = a[index];

a[index] = a[i];

a[i] = smallerNumber;

}

}

// Bubble Sort

public void bubblesort() {

for (int i = nOfItem() - 1; i >= 1; i--) {

boolean check = true;

for (int j = 0; j < i; j++) {

if (a[j] > a[j + 1]) {

int temp;

temp = a[j];

a[j] = a[j + 1];

a[j + 1] = temp;

check = false;

}

}

if (check)

break;

}

}

// Shaker Sort

public void shakersort() {

int j, l, r, k;

int temp;

l = k = 0;

r = n - 1;

do {

for (j = r; j > l; j--) {

if (a[j] < a[j - 1]) {

temp = a[j];

a[j] = a[j - 1];

a[j - 1] = temp;

k = j;

}

}

l = k;

for (j = l; j < r; j++) {

if (a[j] > a[j + 1]) {

temp = a[j];

a[j] = a[j + 1];

a[j + 1] = temp;

k = j;

}

}

r = k;

} while (l < r);

}

// Heap Sort

void heapify(int arr[], int n, int i) {

int largest = i; // Initialize largest as root

int l = 2 \* i + 1; // left = 2\*i + 1

int r = 2 \* i + 2; // right = 2\*i + 2

// If left child is larger than root

if (l < n && arr[l] > arr[largest])

largest = l;

// If right child is larger than largest so far

if (r < n && arr[r] > arr[largest])

largest = r;

// If largest is not root

if (largest != i) {

int swap = arr[i];

arr[i] = arr[largest];

arr[largest] = swap;

// Recursively heapify the affected sub-tree

heapify(arr, n, largest);

}

}

public void Heapsort() {

int n = nOfItem();

for (int i = n / 2 - 1; i >= 0; i--)

heapify(a, n, i);

// One by one extract an element from heap

for (int i = n - 1; i >= 0; i--) {

// Move current root to end

int temp = a[0];

a[0] = a[i];

a[i] = temp;

// call max heapify on the reduced heap

heapify(a, i, 0);

}

}

// Quicksort

public void quickSort() {

sort(0, n - 1);

}

private void sort(int l, int r) { // khai bao 2 bien left and right

int i, j;

int x, t;

i = l;

j = r;

x = a[(l + r) / 2];

do {

while (a[i] < x)

i++;

while (a[j] > x)

j--;

if (i <= j) {

t = a[i];

a[i] = a[j];

a[j] = t;

i++;

j--;

}

} while (i <= j); // left <= right

if (l < j)

sort(l, j);

if (i < r)

sort(i, r);

}

}

public class btTongHopSort {

static Scanner sc = new Scanner(System.in);

// nhap phan tu

static void nhap(Arrlist a) {

int x;

do {

x = sc.nextInt();

if (x > 0)

a.add(x);

} while (x > 0);

}

// xuat phan tu

static void xuat(Arrlist a) {

for (int i = 0; i < a.nOfItem(); i++) {

System.out.print(a.get(i) + " ");

}

System.out.println();

}

public static void main(String[] args) {

Arrlist ar = new Arrlist(1000);

int chon;

do {

System.out.print("1. Tạo mảng ngẫu nhiên: (chưa làm) ");

System.out.print("\n2. Nhập mảng: ");

System.out.print("\n3. Xuất mảng: ");

System.out.print("\n4. Insertion Sort: ");

System.out.print("\n5. Selection Sort: ");

System.out.print("\n6. Bubble Sort: ");

System.out.print("\n7. Shaker Sort: ");

System.out.print("\n8. Interchange Sort: (chưa làm) ");

System.out.print("\n9. Shell Sort: (chưa làm)");

System.out.print("\n10. Heap Sort: ");

System.out.print("\n11. Quick Sort (tăng)");

System.out.print("\n12. Quick Sort (Giảm)");

System.out.print("\n13. Liner Search:");

System.out.print("\n14. Binary Search:");

System.out.print("\nChon: ");

chon = sc.nextInt();

switch (chon) {

case 1:

break;

case 2: {

System.out.print("nhap day so ( 0 ket thuc ): ");

nhap(ar);

break;

}

case 3:

xuat(ar);

break;

case 4:

ar.insertionSort();

break;

case 5:

ar.selectionSort();

break;

case 6:

ar.bubblesort();

break;

case 7:

ar.shakersort();

break;

case 10:

ar.Heapsort();

break;

case 11:

ar.quickSort();

break;

default:

throw new IllegalArgumentException("Unexpected value: " + chon);

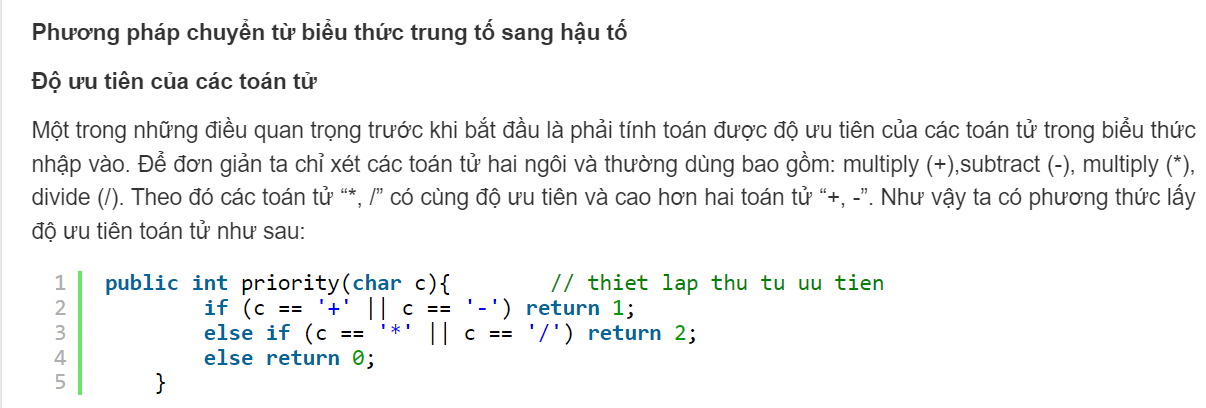
}

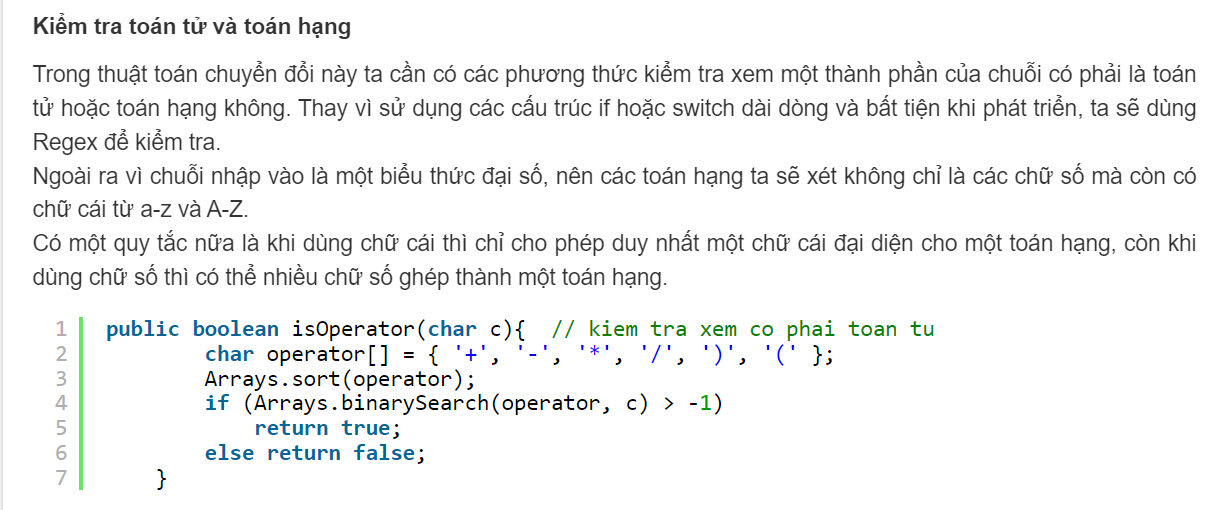
} while (chon > 0);

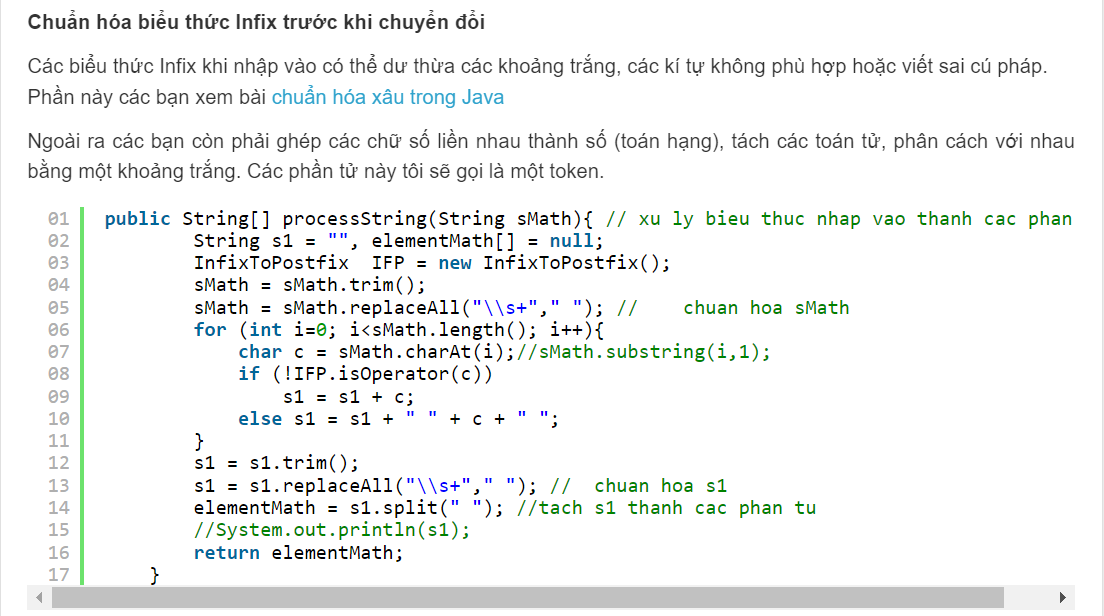
}

}

**Ngăn xếp (Stack) trung tố -> hậu tố**







**Stack đảo chuỗi**

package Stackdaochuoi;

import java.util.\*;

class stackA<T> {

private T[] a;

private int top;

public stackA(int s) {

a = (T[]) new Object[s];

top = -1;

}

public boolean isEmpty() {

return top == -1;

}

public void push(T x) {

a[++top] = x;

}

public T pop() {

return a[top--];

}

}

public class daoChuoi {

public static void main(String[] args) {

String str;

Scanner sc = new Scanner(System.in);

System.out.print("Nhap chuoi:");

;

str = sc.nextLine();

stackA<String> s = new stackA(50);

StringTokenizer stn = new StringTokenizer(str, " ", false);

while (stn.hasMoreTokens())

s.push(stn.nextToken());

System.out.print("Chuoi dao: ");

while (!s.isEmpty()) {

System.out.print(s.pop() + " ");

}

System.out.println();

}

}

**Stack trung tố , hậu tố (tính toán)**

package StackTrungHauTo;

import java.util.\*;

class stackA<T> {

private T[] a;

private int top;

public stackA(int size) {

a = (T[]) new Object[size];

top = -1;

}

// check rỗng

public boolean isEmpty() {

return top == -1;

}

// đẩy vào stack

public void push(T x) {

a[++top] = x;

}

// lấy ra stack

public T pop() {

return a[top--];

}

}

public class trungHauTo {

// phân loại độ ưu tiên

static int doUT(char pt) {

switch (pt) {

case '+':

case '-':

return 1;

case '\*':

case '/':

return 2;

}

return 0;

}

// tính toán

static int tinhToan(Integer b, Integer a, char pt) {

switch (pt) {

case '+':

return a + b;

case '-':

return a - b;

case '\*':

return a \* b;

case '/':

return a / b;

}

return 0;

}

public static void main(String[] args) {

String str, tmp = "";

int i, k, n;

boolean error, stop;

String[] trungTo = new String[50];

String[] hauTo = new String[50];

Scanner sc = new Scanner(System.in);

System.out.print("Nhap bieu thuc: ");

str = sc.nextLine();

stackA<String> s = new stackA(50);

StringTokenizer stn = new StringTokenizer(str, "+-\*/()", true);

n = 0;

while (stn.hasMoreTokens())

// tăng giá trị arr trung tố sau từng lần lặp

trungTo[n++] = stn.nextToken();

k = i = 0;

error = false;

while (i < n && !error) {

switch (trungTo[i].charAt(0)) {

case '(':

s.push("(");

break;

case ')':

do {

if (s.isEmpty())

error = true;

else {

tmp = s.pop();

if (tmp.charAt(0) != '(')

hauTo[k++] = tmp;

}

} while (tmp.charAt(0) != '(' && !error);

break;

case '+':

case '-':

case '\*':

case '/':

stop = false;

while (!s.isEmpty() && !stop) {

tmp = s.pop();

if (doUT(trungTo[i].charAt(0)) < doUT(tmp.charAt(0)))

hauTo[k++] = tmp;

else {

s.push(tmp);

stop = true;

}

}

s.push(trungTo[i]);

break;

default:

hauTo[k++] = trungTo[i];

break;

}

i++;

}

while (!s.isEmpty() && !error) {

tmp = s.pop();

if (tmp.charAt(0) == '(')

error = true;

else

hauTo[k++] = tmp;

}

if (!error) {

for (i = 0; i < k; i++)

System.out.print(hauTo[i] + " | ");

} else

System.out.println("loi!!");

stackA<Integer> ss = new stackA(50);

i = 0;

tmp = "+-\*/";

while (i < k && !error) {

if (tmp.contains(hauTo[i]))

ss.push(tinhToan(ss.pop(), ss.pop(), hauTo[i].charAt(0)));

else

ss.push(Integer.parseInt(hauTo[i]));

i++;

}

if (!error) {

int x = ss.pop();

if (ss.isEmpty())

System.out.println("\nKet qua: " + x);

else

System.out.println("\nLoi!!");

}

}

}

**Bài tập Stack (Giai thừa)**

package stackGiaithua;

import java.util.\*;

class stackA<T> {

private T[] a;

private int top;

public stackA(int s) {

a = (T[]) new Object[s];

top = -1;

}

public boolean isEmpty() {

return top == -1;

}

public void push(T x) {

a[++top] = x;

}

public T pop() {

return a[top--];

}

}

public class giaiThua {

static int tinhToan(Integer a, Integer b) {

return a \* b;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int kq = 0;

int k = 0, i = 1;

System.out.println("nhap n giai thua: ");

int n = sc.nextInt();

stackA<Integer> st = new stackA(50);

while (n > 0) {

st.push(n);

n--;

k++;

}

while (i < k) {

st.push(tinhToan(st.pop(), st.pop()));

i++;

}

kq = st.pop();

if (st.isEmpty())

System.out.println(kq);

}

}

**Bài tập Queue(Cách 1)**

package hangDoi\_Queue;

import java.util.\*;

//Bổ sung một phần tử vào cuối danh sách (rear)

//Loại bỏ một phần tử ở đầu danh sách (front)

class queue {

private int[] queArray;

private int maxSize, front, rear, nItems;

public queue(int s) {

maxSize = s;

queArray = new int[maxSize];

front = 0;

rear = -1;

nItems = 0;

}

// đẩy vào

public void insert(int x) {

// phần tử cuối danh sách thì cho rear = -1

if (rear == maxSize - 1)

rear = -1;

queArray[++rear] = x;

nItems++;

}

// xóa

public int remove() {

int temp = queArray[front++];

if (front == maxSize)

front = 0;

nItems--;

return temp;

}

public int peekFront() {

return queArray[front];

}

// kiểm tra rỗng hay không

public boolean isEmpty() {

return (nItems == 0);

}

// kiểm tra tối đa

public boolean isFull() {

return (nItems == maxSize);

}

// kiểm tra kích thước

public int size() {

return nItems;

}

}

public class baiTapQueue {

// đếm số hàng đơn vị

static int socs(int n) {

int d = 0;

while (n > 0) {

d++;

n = n / 10;

}

return d;

}

// thuật toán

static void sort(int[] a, int n) {

int i, s, d, t, k, x, lt;

queue[] q = new queue[20];

for (i = 0; i < 20; i++)

q[i] = new queue(100);

x = a[0];

for (i = 0; i < n; i++) {

if (x < a[i])

x = a[i];

q[a[i] % 10].insert(a[i]);

}

lt = 1;

s = 0;

d = 10;

for (k = socs(x) - 1; k > 0; k--) {

lt = lt \* 10;

for (i = s; i < s + 10; i++)

while (!q[i].isEmpty()) {

x = q[i].remove();

q[d + x / lt % 10].insert(x);

}

t = s;

s = d;

d = t;

}

n = 0;

for (i = s; i < s + 10; i++)

while (!q[i].isEmpty())

a[n++] = q[i].remove();

}

static void ngauNhien(int[] a, int n) {

Random rd = new Random();

for (int i = 0; i < n; i++)

a[i] = rd.nextInt(1000000);

}

static void in(int[] a, int n) {

for (int i = 0; i < n; i++) {

System.out.printf("%8d", a[i]);

if (i % 10 == 9)

System.out.println();

}

}

public static void main(String[] args) {

int[] a = new int[100];

int n = 100;

ngauNhien(a, n);

System.out.println("Dãy ngẫu nhiên: ");

in(a, n);

sort(a, n);

System.out.println("Dãy đã sắp thứ tự: ");

in(a, n);

}

}

**Bài tập Queue (Cách 2)**

package hangDoi\_QueueC2;

import java.util.\*;

// Bổ sung một phần tử vào cuối danh sách (rear)

// Loại bỏ một phần tử ở đầu danh sách (front)

class queue {

private int[] queArray;

private int maxSize, front, rear, nItems;

public queue(int s) {

maxSize = s;

queArray = new int[maxSize];

front = 0;

rear = -1;

nItems = 0;

}

// đẩy vào

public void insert(int x) {

// phần tử cuối danh sách thì cho rear = -1

if (rear == maxSize - 1)

rear = -1;

queArray[++rear] = x;

nItems++;

}

// xóa

public int remove() {

int temp = queArray[front++];

if (front == maxSize)

front = 0;

nItems--;

return temp;

}

public int peekFront() {

return queArray[front];

}

// kiểm tra rỗng hay không

public boolean isEmpty() {

return (nItems == 0);

}

// kiểm tra tối đa

public boolean isFull() {

return (nItems == maxSize);

}

// kiểm tra kích thước

public int size() {

return nItems;

}

}

public class baiTapQueue2 {

// đếm số hàng đơn vị

static int socs(int n) {

int d = 0;

while (n > 0) {

d++;

n = n / 10;

}

return d;

}

// thuật toán

static void sort(int[] a, int n) {

int i, k, x, lt;

queue[] q1 = new queue[10];

queue[] q2 = new queue[10];

queue[] s, d, t;

for (i = 0; i < 10; i++) {

q1[i] = new queue(n);

q2[i] = new queue(n);

}

x = a[0];

for (i = 0; i < n; i++) {

if (x < a[i])

x = a[i];

q1[a[i] % 10].insert(a[i]);

}

lt = 1;

s = q1;

d = q2;

for (k = socs(x) - 1; k > 0; k--) {

lt = lt \* 10;

for (i = 0; i < 10; i++)

while (!s[i].isEmpty()) {

x = s[i].remove();

d[x / lt % 10].insert(x);

}

t = s;

s = d;

d = t;

}

n = 0;

for (i = 0; i < 10; i++)

while (!s[i].isEmpty())

a[n++] = s[i].remove();

}

static void ngauNhien(int[] a, int n, int max) {

Random rd = new Random();

for (int i = 0; i < n; i++)

a[i] = rd.nextInt(max);

}

static void in(int[] a, int n) {

for (int i = 0; i < n; i++) {

System.out.printf("%8d", a[i]);

if (i % 10 == 9)

System.out.println();

}

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Nhập số phần tử n và max: ");

int n = sc.nextInt();

int max = sc.nextInt();

int[] a = new int[n];

ngauNhien(a, n, max);

System.out.println("Dãy ngẫu nhiên: ");

in(a, n);

sort(a, n);

System.out.println("Dãy đã sắp xếp thứ tự: ");

in(a, n);

}

}

**Bài tập danh sách liên kết đơn**

package danhSachLienKet;

import java.util.\*;

class sinhVien {

private String masv;

private String ho, ten;

private float diem;

public sinhVien(String masv, String ho, String ten, float diem) {

this.masv = masv;

this.ho = ho;

this.ten = ten;

this.diem = diem;

}

public sinhVien() {

}

public String getMasv() {

return masv;

}

public void setMasv(String masv) {

this.masv = masv;

}

public String getHo() {

return ho;

}

public void setHo(String ho) {

this.ho = ho;

}

public String getTen() {

return ten;

}

public void setTen(String ten) {

this.ten = ten;

}

public float getDiem() {

return diem;

}

public void setDiem(float diem) {

this.diem = diem;

}

}

class Link {

public sinhVien dData;

public Link next;

public Link(sinhVien d) {

dData = d;

}

}

class LinkedList {

private Link first, pos, last;

private int count;

public LinkedList() {

first = last = pos = null;

count = 0;

}

public boolean isEmpty() {

return first == null;

}

private void insert(sinhVien dd, Link p) {

Link newLink = new Link(dd);

// chèn đầu danh sách

if (p == null) {

newLink.next = first;

first = newLink;

} else

// chèn trong danh sách

{

newLink.next = p.next;

p.next = newLink;

}

// chèn cuối danh sách

if (newLink.next == null) {

last = newLink;

count++;

}

}

// các hàm chèn phần tử

public void insertFirst(sinhVien dd) {

insert(dd, null);

}

public void insertPos(sinhVien dd) {

insert(dd, pos);

}

public void insertLast(sinhVien dd) {

insert(dd, last);

}

private void delete(Link p) {

Link t;

// xóa phần tử đầu danh sách

if (p == null) {

t = first;

first = t.next;

// xóa phần tử giữa danh sách

} else {

t = p.next;

p.next = t.next;

}

// xóa phần tử cuối danh sách

if (t.next == null)

last = p;

count--;

}

public void deleteFirst() {

delete(null);

}

public void deletePos() {

delete(pos);

}

// tìm kiếm trên danh sách ngẫu nhiên

public boolean searchList(sinhVien dd) {

Link c;

c = first;

pos = null;

while (c != null && c.dData.getMasv().compareTo(dd.getMasv()) != 0) {

pos = c;

c = c.next;

}

return c != null;

}

// tìm kiếm trên danh sách có thứ tự

public boolean searchOrderList(sinhVien dd) {

Link c;

c = first;

pos = null;

while (c != null && c.dData.getMasv().compareTo(dd.getMasv()) < 0) {

pos = c;

c = c.next;

}

if (c != null && c.dData.getMasv().compareTo(dd.getMasv()) > 0)

return false;

return c != null;

}

// Khởi đầu từ đầu danh sách

public void starList() {

pos = first;

}

// Chuyển sang phần tử kế tiếp

public void nextLink() {

if (pos == null)

pos = first;

pos = pos.next;

}

// Kiểm tra hết danh sách

public boolean endList() {

return pos == null;

}

// Lấy dữ liệu 1 phần tử

public sinhVien getData() {

return pos.dData;

}

// Số phần tử trong danh sách

public int nItem() {

return count;

}

}

public class lienKetDon {

static Scanner sc = new Scanner(System.in);

static void nhap(LinkedList l) {

String ms, ho, ten;

int d;

sinhVien x;

do {

sc.nextLine();

System.out.print("ma so sv: ");

ms = sc.nextLine();

if (ms.length() > 0) {

System.out.print("ho: ");

ho = sc.nextLine();

System.out.print("ten: ");

ten = sc.nextLine();

System.out.print("Diem: ");

d = sc.nextInt();

x = new sinhVien(ms, ho, ten, d);

l.insertLast(x);

}

} while (ms.length() > 0);

}

static sinhVien tim(LinkedList l, String ms) {

l.starList();

while (!l.endList() && ms.compareTo(l.getData().getMasv()) != 0)

l.nextLink();

if (!l.endList())

return l.getData();

return null;

}

static void xuat(LinkedList l) {

l.starList();

while (!l.endList()) {

sinhVien x = l.getData();

System.out.print(x.getMasv() + " " + x.getHo() + " " + x.getTen() + " " + x.getDiem());

l.nextLink();

}

}

static LinkedList saptt(LinkedList l) {

LinkedList l1 = new LinkedList();

sinhVien x;

while (!l.isEmpty()) {

l.starList();

x = l.getData();

l1.searchOrderList(x);

l1.insertPos(x);

l.deleteFirst();

}

return l1;

}

public static void main(String[] args) {

LinkedList l = new LinkedList();

int chon;

do {

System.out.println("1. nhap");

System.out.println("2. xuat");

System.out.println("3. tim kiem theo mssv");

System.out.println("4. sap xep thu tu theo phuong phap chen");

System.out.println("chon (0. ket thuc):");

chon = sc.nextInt();

switch (chon) {

case 1:

nhap(l);

break;

case 2:

xuat(l);

break;

case 3:

String ms;

sc.nextLine();

System.out.print("ma so sv:");

ms = sc.nextLine();

sinhVien x = tim(l, ms);

if (x == null)

System.out.println("khong tin thay sv");

else

System.out.println(x.getMasv() + " " + x.getHo() + " " + x.getTen() + " " + x.getDiem());

break;

case 4:

l = saptt(l);

break;

}

} while (chon > 0);

}

}

**Bài tập sơ đồ cây**

package cayNhiPhan;

import java.util.\*;

class Node {

public int dData;

public Node left, right;

public Node(int dd) {

dData = dd;

left = right = null;

}

}

class Tree {

static Scanner sc = new Scanner(System.in);

private Node root;

public Tree() {

root = null;

}

public boolean isEmpty() {

return root == null;

}

public int getData() {

return root.dData;

}

public void setData(int dd) {

root.dData = dd;

}

public Tree getLeft() {

Tree t = new Tree();

t.root = root.left;

return t;

}

public Tree getRight() {

Tree t = new Tree();

t.root = root.right;

return t;

}

// nhập cây số nguyên

public Node inputNode() {

int x = sc.nextInt();

Node r;

if (x == 0)

r = null;

else {

r = new Node(x);

r.left = inputNode();

r.right = inputNode();

}

return r;

}

public Tree inputTree() {

Tree t = new Tree();

t.root = inputNode();

return t;

}

}

public class nhiPhan {

public static int Sonut(Tree t) {

if (t.isEmpty())

return 0;

else

return 1 + Sonut(t.getLeft()) + Sonut(t.getRight());

}

public static int tong(Tree t) {

if (t.isEmpty())

return 0;

else

return t.getData() + tong(t.getLeft()) + tong(t.getRight());

}

public static int Sonutla(Tree t) {

if (t.isEmpty())

return 0;

if (t.getLeft().isEmpty() && t.getRight().isEmpty())

return 1;

return Sonutla(t.getLeft()) + Sonutla(t.getRight());

}

private static int max(int a, int b) {

return (a > b ? a : b);

}

private static int docao(Tree t) {

if (t.isEmpty())

return 0;

return 1 + max(docao(t.getLeft()), docao(t.getRight()));

}

public static void displayTree(Tree t, int k) {

if (!t.isEmpty()) {

displayTree(t.getLeft(), k + 1);

System.out.print("\n" + blank(4 \* k) + t.getData());

displayTree(t.getRight(), k + 1);

}

}

// In cây số nguyên

private static String blank(int n) {

String s = " ";

for (int i = 0; i < n; i++)

s = s + " ";

return s;

}

public static void displayTree(Tree t) {

displayTree(t, 1);

}

// Duyệt tiền tự

public static void preOrder(Tree t) {

if (!t.isEmpty()) {

System.out.print(t.getData() + " ");

preOrder(t.getLeft());

preOrder(t.getRight());

}

}

// Duyệt trung tự

public static void inOrder(Tree t) {

if (!t.isEmpty()) {

inOrder(t.getLeft());

System.out.print(t.getData() + " ");

inOrder(t.getRight());

}

}

// Duyệt hậu tự

public static void postOrder(Tree t) {

if (!t.isEmpty()) {

postOrder(t.getLeft());

postOrder(t.getRight());

System.out.print(t.getData() + " ");

}

}

// Đếm số nút trên mức chẵn/lẻ của cây

private static int Snmcl(Tree t, int m) {

if (!t.isEmpty())

return 0;

else

return m % 2 + Snmcl(t.getLeft(), m + 1) + Snmcl(t.getRight(), m + 1);

}

// Số nút mức lẻ

public static int Snml(Tree t) {

return Snmcl(t, 1);

}

// Số nút mức chẵn

public static int Snmc(Tree t) {

return Snmcl(t, 0);

}

// đếm số nút trên mức thứ k

public static int Snmk(Tree t, int k) {

if (t.isEmpty())

return 0;

if (k == 1)

return 1;

return Snmk(t.getLeft(), k - 1) + Snmk(t.getRight(), k - 1);

}

public static int soNutMotCon(Tree t) {

if (t.isEmpty())

return 0;

if ((t.getLeft().isEmpty() && !t.getRight().isEmpty()) || (!t.getLeft().isEmpty() && t.getRight().isEmpty()))

return 1;

return soNutMotCon(t.getLeft()) + soNutMotCon(t.getRight());

}

public static int soNutMotConChan(Tree t) {

if (t.isEmpty())

return 0;

if ((t.getLeft().isEmpty() && !t.getRight().isEmpty()) || (!t.getLeft().isEmpty() && t.getRight().isEmpty())) {

if (t.getData() % 2 == 0)

return 1 + soNutMotConChan(t.getLeft()) + soNutMotConChan(t.getRight());

}

return soNutMotConChan(t.getLeft()) + soNutMotConChan(t.getRight());

}

public static int soNutHaiCon(Tree t) {

if (t.isEmpty())

return 0;

if (!t.getLeft().isEmpty() && !t.getRight().isEmpty())

return 1 + soNutHaiCon(t.getLeft()) + soNutHaiCon(t.getRight());

return soNutHaiCon(t.getLeft()) + soNutHaiCon(t.getRight());

}

// 2 4 5 0 0 8 0 0 1 6 0 0 3 9 0 0 0

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

Tree t = new Tree();

// int a = 1, b = 5;

System.out.print("Nhập cây nhị phân: ");

t = t.inputTree();

displayTree(t, 4);

System.out.println();

System.out.println("Số nút trên cây: " + Sonut(t));

System.out.println("Tổng giá trị các nút: " + tong(t));

System.out.println("Số nút lá trên cây: " + Sonutla(t));

// System.out.println("Độ cao cây: " + max(a, b));

// System.out.println("Số nút lẻ: " + Snml(t));

// System.out.println("Số nút chẵn: " + Snmc(t));

System.out.println("Số nút 1 con: " + soNutMotCon(t));

System.out.println("Số nút 1 con chẵn: " + soNutMotConChan(t));

System.out.println("Số nút 2 con: " + soNutHaiCon(t));

}

}