1. Определить класс train, содержащий поля: название пункта назначения, номер поезда, время отправки. Ввести данные в массив, упорядочить элементы по номерам поездов. Добавить вывод информации о поезде, номер которого введен пользователем. Добавить сортировку по пункту назначения.

//MAIN  
public class Ex {  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.in);  
 Scanner scL = new Scanner(System.in);  
 TrainGroup tg = new TrainGroup();  
 int n = sc.nextInt();  
 for(int j = 0;j<n;j++){  
 String name = scL.nextLine();  
 String num = scL.nextLine();  
 int time = sc.nextInt();  
 Train t = new Train(name,num,time);  
 tg.add(t);  
 }  
 tg.printU();  
 tg.print();   
 System.out.println("Information train on number?");  
 String m = scL.nextLine();  
 tg.find(m);   
 tg.sortByName();  
 tg.print();  
 }  
}  
//CLASS TRAIN  
class Train{  
 private String name, num;  
 private int time;  
 Train(String name, String num, int time){  
 this.name = name;  
 this.num = num;  
 this.time = time;   
 }  
 String getName() {return name;}  
 String getNum() {return num;}  
 double getTime() {return time;}   
 void setName(String name) {this.name = name;}  
 void setNum(String num) {this.num = num;}  
 void setTime(int time) {this.time = time;}   
 public String toString()  
 return "Name: " + name + "\n Number: " + num + "\n Time: " + time;  
 public static Comparator<Train> NameComparator = new Comparator<Train>(){  
 public int compare(Train t1, Train t2){  
 return t1.getName().compareTo(t2.getName());  
 }  
 };  
 public static Comparator<Train> NumComparator = new Comparator<Train>(){  
 public int compare(Train t1, Train t2){  
 return t1.getNum().compareTo(t2.getNum());  
 }  
 };  
}  
//ARRAYLIST TRAINGROUP  
class TrainGroup {  
 ArrayList<Train>t;  
 TrainGroup()  
 t = new ArrayList<Train>();  
 void add(Train s)  
 t.add(s);  
 void print(){  
 for(int i = 0;i<t.size();i++)  
 System.out.println(t.get(i).toString());  
 }  
 void find(String m){  
 boolean f = false;  
 for(int i = 0;i<t.size();i++){  
 if(m.equalsIgnoreCase(t.get(i).getNum())){  
 f = true;  
 System.out.println(t.get(i).toString());  
 }  
 }  
 if(!f)  
 System.out.println("Train not find");  
 }  
 void sortByName()  
 t.sort(Train.NameComparator);  
 void printU()  
 t.sort(Train.NumComparator);

1. Создать абстрактный класс Фигура с функцией – площадь и периметр. Реализовать интерфейс в классах Круг, Прямоугольный треугольник, Трапеция со своими функциями площади и периметра.

//MAIN  
public class Ex1 {  
 public static void main(String[] args) {  
 Figure [] f;  
 Scanner sc = new Scanner(System.in);  
 int k = sc.nextInt();  
 f = new Figure[k];  
 for(int i = 0;i<k;i++){  
 System.out.println("1 - krug, 2 - pr treug, 3 - trapez");  
 int p = sc.nextInt();  
 if(p==1){  
 double r = sc.nextDouble();  
 Krug krug = new Krug(r);  
 f[i] = krug;  
 }  
 if(p==2){  
 double a = sc.nextDouble();  
 double b = sc.nextDouble();  
 double c = sc.nextDouble();  
 Treug treug = new Treug(a,b,c);  
 f[i]=treug;  
 }  
 if(p==3){  
 double a = sc.nextDouble();  
 double b = sc.nextDouble();  
 double c = sc.nextDouble();  
 double d = sc.nextDouble();  
 double h = sc.nextInt();  
 Trapez trapez = new Trapez(a,b,c,d,h);  
 f[i]=trapez;  
 }  
 }  
 for(int i = 0;i<k;i++){  
 f[i].print();  
 System.out.println("P = " + f[i].P()+" "+"S= "+f[i].S());  
 }   
 }   
}  
//INTERFACE  
abstract class Figure {  
 abstract double P();  
 abstract double S();  
 abstract void print();  
}  
//KRUG  
class Krug extends Figure{  
 double r;  
 Krug(double r)  
 this.r=r;  
 @Override double P()  
 return 2\*Math.PI\*r;  
 @Override double S()  
 return Math.PI\*r\*r;  
 @Override void print()  
 System.out.println("Krug = "+r);  
}  
//TREUG  
class Treug extends Figure{  
 double a;  
 double b;  
 double c;  
 Treug(double a, double b, double c){  
 this.a=a;  
 this.b=b;  
 this.c=c;  
 }  
 @Override double P()  
 return a+b+c;  
 @Override double S()  
 return (a\*b)/2;  
 @Override void print()  
 System.out.println("Treug = "+a+" "+b+" "+c);  
}  
//TRAPEZ  
class Trapez extends Figure{  
 double a;  
 double b;  
 double c;  
 double d;  
 double h;  
 Trapez(double a, double b, double c, double d, double h){  
 this.a=a;  
 this.b=b;  
 this.c=c;  
 this.d=d;  
 this.h=h;  
 }  
 @Override double P()  
 return a+b+c+d;  
 @Override double S()  
 return ((a+b)/2)\*h;  
 @Override void print()  
 System.out.println("Trapez = "+a+" "+b+" "+c+" "+d);  
}

1. Определить параметризированный класс вектор с методами добавления элемента, замены порядка элементов на обратный, поиск элемента под заданным индексом, поиск максимального элемента, вывод до заданного индекса. Создать экземпляры этого класса для параметра типа Double, String.

//MAIN  
public class Ex2 {  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.in);  
 System.out.println("Size vector double: ");  
 int n = sc.nextInt();  
 Double[]v=new Double[n];  
 for(int i=0;i<v.length;i++){  
 v[i]=sc.nextDouble();  
 }  
 Vec<Double> vd = new Vec<>(v);  
 try{  
 System.out.println("Element pod indexom 2:");  
 System.out.println(vd.elemByInd(2));  
 System.out.println("Vivod vectora:");  
 vd.print();  
 System.out.println("Chast vectora do 2 elementa:");  
 vd.chast(2);  
 System.out.println("Vivid obratnii:");  
 vd.obr();  
 System.out.println("max = " + vd.max());  
 }  
 catch(ArrayIndexOutOfBoundsException e){  
 System.out.println(e.getMessage());  
 }  
 System.out.println("Size vector string:");  
 int m = sc.nextInt();  
 String[]v1=new String[m];  
 for(int i=0;i<v1.length;i++){  
 v1[i]=sc.nextLine();  
 }  
 Vec<String> vs = new Vec<>(v1);  
 try{  
 System.out.println("Element pod indexom 2:");  
 System.out.println(vs.elemByInd(2));  
 System.out.println("Vivod vectora:");  
 vs.print();  
 System.out.println("Chast vectora do 2 elementa:");  
 vs.chast(2);  
 System.out.println("Vivid obratnii:");  
 vs.obr();  
 System.out.println("max = " + vs.max());  
 }  
 catch(ArrayIndexOutOfBoundsException e){  
 System.out.println(e.getMessage());  
 }  
 }   
}  
//CLASS VEC  
class Vec<T extends Comparable<T>>{  
 T[]a;  
 Vec(T[]a){  
 this.a = a.clone();  
 }  
 void print(){  
 for(int i = 0;i<a.length;i++)  
 System.out.print(a[i]+" ");  
 System.out.println();  
 }  
 T elemByInd(int i) throws ArrayIndexOutOfBoundsException{  
 if(i<0 || i>=a.length)  
 throw new ArrayIndexOutOfBoundsException("Vihod za granitsi indexa");  
 return a[i];  
 }  
 void obr(){  
 for(int i=0;i<a.length/2;i++){  
 T t=a[i];  
 a[i]=a[a.length-i-1];  
 a[a.length-i-1]=t;   
 }  
 for(int i =0;i<a.length;i++)  
 System.out.print(a[i]+" ");  
 System.out.println();  
 }  
 void chast(int p) throws ArrayIndexOutOfBoundsException{  
 if(p<0 || p>= a.length)  
 throw new ArrayIndexOutOfBoundsException("Vihod za granitsi indexa");  
 for(int i = 0;i<p;i++)  
 System.out.print(a[i]+" ");  
 System.out.println();  
 }  
 T max(){  
 T res = a[0];  
 for(int i = 0;i<a.length;i++){  
 if(a[i].compareTo(res)>0)  
 res=a[i];  
 }  
 return res;  
 }  
}

1. Определить класс параллелепипед. Предусмотреть методы создания объектов, перемещения, изменения размеров, вычисления высоты, нахождения центра тяжести.

//MAIN  
public class Main {  
 public static void main(String[] args) {  
 Scanner sc=new Scanner(System.in);  
 int x=sc.nextInt();  
 int y=sc.nextInt();  
 int z=sc.nextInt();  
 int xsize=sc.nextInt();  
 int ysize=sc.nextInt();  
 int zsize=sc.nextInt();  
 paral p=new paral(x,y,z,xsize,ysize,zsize);  
 int xp=sc.nextInt();  
 int yp=sc.nextInt();  
 int zp=sc.nextInt();  
 p.peremesh(xp,yp,zp);  
 int xs=sc.nextInt();  
 int ys=sc.nextInt();  
 int zs=sc.nextInt();  
 p.size(xs,ys,zs);  
 p.height();  
 p.centr();  
 }  
}  
//CLASS PARAL

public class paral {

int x,y,z;

int xsize,ysize,zsize;

paral(int x,int y,int z,int xsize,int ysize,int zsize){

this.x=x;

this.y=y;

this.z=z;

this.xsize=xsize;

this.ysize=ysize;

this.zsize=zsize;

}

public int getX() {

return x;

}

public void setX(int x) {

this.x = x;

}

public int getY() {

return y;

}  
 public void setY(int y) {

this.y = y;

}

public int getXsize() {

return xsize;

}

public void setXsize(int xsize) {

this.xsize = xsize;

}

public int getYsize() {

return ysize;

}

public void setYsize(int ysize) {

this.ysize = ysize;

}

public int getZ() {

return z;

}

public void setZ(int z) {

this.z = z;

}

public int getZsize() {

return zsize;

}

public void setZsize(int zsize) {

this.zsize = zsize;

}

public void peremesh(int xp, int yp,int zp){

x+=xp;

y+=yp;

z+=zp;

System.out.println("New position= "+x+" "+y+" "+z);

}

public void size(int xs,int ys,int zs){

ysize+=ys;

xsize+=xs;

zsize+=zs;

System.out.println("New size= "+xsize+" "+ysize+" "+zsize);

}

public void height(){

System.out.println("Height= "+xsize\*ysize\*zsize/(xsize\*ysize));

}

public void centr(){

System.out.println("Centr= "+x+xsize/2+" "+y+ysize/2+" "+z+zsize/2);

}

}

1. Создать абстрактный класс справочник с методами вывода на экран и поиска по имени. Создать производные классы Персона, Друг, Организация. Создать массив из n элементов, вывести на экран и организовать поиск по имени.

//MAIN

public class Main {

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

Scanner scl=new Scanner(System.in);

int n=sc.nextInt();

Spravochnik[] s=new Spravochnik[n];

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

int mode=sc.nextInt();

if(mode==1){

String name = scl.nextLine();

String adress = scl.nextLine();

int tel = sc.nextInt();

int date = sc.nextInt();

Drug d = new Drug(name, adress, tel, date);

s[i] = d;

}

if(mode==2){

String name1 = scl.nextLine();

String adress1 = scl.nextLine();

int tel1 = sc.nextInt();

Persona p = new Persona(name1, adress1, tel1);

s[i] = p;

}

if(mode==3){

String name2 = scl.nextLine();

String adress2 = scl.nextLine();

String nazv = scl.nextLine();

int tel2 = sc.nextInt();

int faks = sc.nextInt();

Org o = new Org(name2, adress2, nazv, tel2, faks);

s[i] = o;

}

}

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

s[i].print();

String fname=scl.nextLine();

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

s[i].find(fname);

}

}

//CLASS DRUG

class Drug extends Spravochnik{

String name,adress;

int tel,date;

Drug(String name,String adress,int tel,int date){

this.name=name;

this.adress=adress;

this.date=date;

this.tel=tel;

}

public String getName()

return name;

public void setName(String name)

this.name = name;

public String toString()

return name+" "+adress+" "+tel+" "+date+"\n";

@Override void print()

System.out.println(toString());

@Override void find(String n){

if(n.equalsIgnoreCase(name))

print();

}

public static Comparator<Drug>NameComporator=new Comparator<Drug>(){

public int compare(Drug o1, Drug o2)

return o1.getName().compareTo(o2.getName());

};

}

//CLASS PERSONA

class Persona extends Spravochnik{

String name,adress;

int tel;

Persona(String name,String adress,int tel){

this.name=name;

this.adress=adress;

this.tel=tel;

}

public String getName()

return name;

public void setName(String name)

this.name = name;

public String toString()

return name+" "+adress+" "+tel+"\n";

@Override void print()

System.out.println(toString());

@Override void find(String n){

if(n.equalsIgnoreCase(name))

print();

}

public static Comparator<Persona> NameComporator=new Comparator<Persona>(){

public int compare(Persona o1, Persona o2)

return o1.getName().compareTo(o2.getName());

};

}

//CLASS ORG

class Org extends Spravochnik{

String name,adress,nazv;

int tel,faks;

Org(String name,String adress,String nazv,int tel,int faks){

this.name=name;

this.adress=adress;

this.faks=faks;

this.tel=tel;

this.nazv=nazv;

}

public String getName()

return name;

public void setName(String name)

this.name = name;

public String toString()

return name+" "+adress+" "+tel+" "+faks+" "+nazv+"\n";

@Override void print()

System.out.println(toString());

@Override void find(String n){

if(n.equalsIgnoreCase(nazv))

print();

}

public static Comparator<Org> NameComporator=new Comparator<Org>(){

public int compare(Org o1, Org o2)

return o1.getName().compareTo(o2.getName());

};

}

//ABSTRACT  
abstract class Spravochnik {

abstract void print();

abstract void find(String name);

}

1. Создать класс работник и производные классы почасовая оплата, Штатный работник и Процентная ставка. Определить в каждом классе функцию начисления зарплаты.

//MAIN

public class Main {

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

Scanner scl=new Scanner(System.in);

String name=scl.nextLine();

int age=sc.nextInt();

Rab rab=new Rab(name,age);

System.out.println(rab.toString());

int stoim=sc.nextInt();

int chas=sc.nextInt();

Chas ch=new Chas(name,age,stoim,chas);

System.out.println(ch.toString());

int zarp=sc.nextInt();

Shtat sh=new Shtat(name,age,zarp);

System.out.println(sh.toString());

Double zarpl=sc.nextDouble();

int proc=sc.nextInt();

Procent pro=new Procent(name,age,zarpl,proc);

System.out.println(pro.toString());

}

}  
//CLASS CHAS  
public class Chas extends Rab{

int stoim,chas;

Chas(String name,int year,int stoim,int chas){

super(name,year);

this.stoim=stoim;

this.chas=chas;

}

public int getStoim()

return stoim;

public void setStoim(int stoim)

this.stoim = stoim;

public int getChas()

return chas;

public void setChas(int chas)

this.chas = chas;

public String toString()

return super.toString()+chas\*stoim+"\n";

}

//CLASS PROCENT  
public class Procent extends Rab{

int procent;

double zarp;

Procent(String name,int age,double zarp,int procent)

{

super(name,age);

this.zarp=zarp;

this.procent=procent;

}

public Double getZarp()

return zarp;

public void setZarp(Double zarp)

this.zarp = zarp;

public int getProcent()

return procent;

public void setProcent(int procent)

this.procent = procent;

public String toString()

return super.toString()+zarp\*procent/100;

}

//CLASS SHTAT  
public class Shtat extends Rab{

int zarp;

Shtat(String name,int age,int zarp){

super(name, age);

this.zarp=zarp;

}

public int getZarp()

return zarp;

public void setZarp(int zarp)

this.zarp = zarp;

public String toString()

return super.toString()+zarp+"\n";

}

//CLASS RAB  
public class Rab {

String name;

int age;

Rab(String name,int age){

this.name=name;

this.age=age;

}

public String getName(){return name;}

public void setName(String name){this.name=name;}

public int getAge(){return age;}

public void setAge(int age){this.age=age;}

public String toString()

return name+" "+age+"\n";

}

1. Определить класс «Авто», экземпляры которого имеют поля – марка, мощность, объем двигателя. Определить функции назначения и изменения полей и функцию вывода на экран. Определить производный класс – такси, экземпляры которого имеют поле – тариф. Определить функцию вывода на экран. Определить функцию переназначения тарифа и функцию расчета стоимости поездки.

//MAIN

public class Ex3 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

Scanner scL = new Scanner(System.in);

String marka = scL.nextLine();

int mosh = sc.nextInt();

double vol = sc.nextDouble();

Avto a = new Avto(marka, mosh, vol);

System.out.println(a.toString());

String tarif = scL.nextLine();

int pricekm = sc.nextInt();

Taxi t = new Taxi(marka, mosh, vol, tarif, pricekm);

double km1 = sc.nextInt();

System.out.println("Stoimost poezdki = " + t.stoim(km1));

System.out.println(t.toString());

}

}

//AVTO

class Avto {

String marka;

int mosh;

double vol;

Avto(String marka, int mosh, double vol){

this.marka = marka;

this.mosh = mosh;

this.vol = vol;

}

public String toString()

return "Mark: " + marka + "\n Moshnost: " + mosh + "\n V dvigat: " + vol;

String getMarka() { return marka; }

int getMosh() { return mosh; }

double getVol() { return vol; }

String setMarka() { return marka; }

int setMosh() { return mosh; }

double setVol() { return vol; }

}

//TAXI  
class Taxi extends Avto{

private String tarif;

private int pricekm;

Taxi(String marka, int mosh, double vol, String tarif, int pricekm){

super(marka, mosh, vol);

this.tarif = tarif;

this.pricekm = pricekm;

}

double stoim(double km) {return km\*pricekm;}

String getTarif() {return tarif;}

int getPricekm() {return pricekm;}

String setTarif() {return tarif;}

int setPricekm() {return pricekm;}

public String toString()

return super.toString() + "\n Tarif: " + tarif + "\n PriceKm: " + pricekm;

}

1. Создать класс точка, предусмотреть в нем несколько конструкторов, методы определения координатной четверти, которой принадлежит точка, перемещения точки, метод проверяющий являются ли две точки симметричными относительно начала координат, метод, вычисляющий расстояние между двумя точками.

//MAIN

public class Ex4 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter p1 = ");

int a1 = sc.nextInt();

int b1 = sc.nextInt();

Point p1 = new Point(a1,b1);

System.out.println("Koordinat chetv: "+p1.koordinat(a1, b1));

int x = sc.nextInt();

int y = sc.nextInt();

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

p1.per(p1, x, y);

System.out.println("Enter p2: ");

a1 = sc.nextInt();

b1 = sc.nextInt();

Point p2 = new Point(a1,b1);

System.out.println("Symmetric: " + p1.sym(p2, p1));

System.out.println("Rast: " + p1.dist(p2));

}

}

//POINT

public class Point {

int x,y;

Point(int x, int y){

this.x = x;

this.y = y;

}

int getX() {return x;}

int getY() {return y;}

void setX(int x) {this.x = x;}

void setY(int y) {this.y = y;}

int koordinat(int x, int y){

if(x>0 && y>0)

return 1;

else if(x<0 && y>0)

return 2;

else if(x<0 && y<0)

return 3;

else if(x>0 && y<0)

return 4;

else return 0;

}

void per(Point p, int x, int y){

int newX = p.getX() + x;

int newY = p.getY() + y;

System.out.println(newX + " " + newY);

}

boolean sym(Point p, Point p1)

return p.x == -p1.x && p.y == -p1.y;

double dist(int x, int y){

int dx = this.x - x;

int dy = this.y - y;

return Math.sqrt(dx\*dx+dy\*dy);

}

double dist(Point p)

return dist(p.getX(), p.getY());

}

1. ПАРАМЕТРИЗИРОВАННАЯ МАТРИЦА

//MAIN  
public class KR\_Djenerics {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Size mass[][]: ");

int n = sc.nextInt();

Integer[][]m1= new Integer[n][n];

for(int i=0;i<m1.length;i++){

for(int j = 0;j<m1.length;j++){

m1[i][j]=sc.nextInt();

}

}

Matrix<Integer> mi = new Matrix<>(m1);

try{

System.out.println("Vivod massiva:");

mi.print();

System.out.println("Elem pod indexom [1][2] = " + mi.elemByInd(1,2));

System.out.println("Stroka s max elem:" /\*+ Arrays.toString(mi.maxEl())\*/);

mi.maxEl();

System.out.println();

System.out.println("Smena strok: ");

mi.obr();

}

catch(ArrayIndexOutOfBoundsException e){

System.out.println(e.getMessage());

}

}

}  
//CLASS

class Matrix<T extends Comparable<T>>{

T[][]a;

Matrix(T[][]a){

this.a = a.clone();

}

void print(){

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

for(int j = 0;j<a.length;j++)

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

System.out.println();

}

}

T elemByInd(int i, int j) throws ArrayIndexOutOfBoundsException{

if(i<0 || i>=a.length)

if(j<0 || j>=a.length)

throw new ArrayIndexOutOfBoundsException("Vihod za granitsi indexa");

return a[i][j];

}

void maxEl(){

int maxInd = 0;

T maxEl = a[0][0];

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

for(int j = 0;j<a[0].length;j++){

if(a[i][j].compareTo(maxEl)>0){

maxEl = a[i][j];

maxInd = i;

}

}

}

for(int j = 0;j<a[0].length;j++){

System.out.print(a[maxInd][j]+ " ");

}

}

void obr(){

for(int i=0;i<a.length/2;i++){

T[] t=a[i];

a[i]=a[a.length-i-1];

a[a.length-i-1]=t;

}

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

for(int j = 0;j<a.length;j++){

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

}

System.out.println();

}

}

}

1. ИНТЕРФЕЙС

//MAIN  
public class Laba8 {

public static void main(String[] args) {

Body [] f;

Scanner sc = new Scanner(System.in);

int k = sc.nextInt();

f = new Body[k];

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

System.out.println("1 - shar, 2 - paralelogram");

int p = sc.nextInt();

if(p==1){

double r = sc.nextDouble();

Shar shar = new Shar(r);

f[i] = shar;

}

if(p==2){

double a = sc.nextDouble();

double b1 = sc.nextDouble();

double c = sc.nextDouble();

Paral paral = new Paral(a,b1,c);

f[i]=paral;

}

}

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

f[i].print();

System.out.println("S = " + f[i].S()+" "+"V= "+f[i].V());

}

}

}

//CLASS  
public class Paral implements Body{

double a,b,c;

Paral(double a, double b, double c){

this.a=a;

this.b=b;

this.c=c;

}

public double V()

return a\*b\*c;

public double S()

return 2\*(a\*b+b\*c+c\*a);

public void print()

System.out.println("Paralelogram = "+a+" "+b+" "+c);

}

//INTERFACE

public interface Body {

double V();

double S();

void print();

1. Реализовать класс студент с методами добавления, удаления студентов по фамилии, вывод данных на экран.

//MAIN  
public class Laba\_9 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

Scanner scL = new Scanner(System.in);

StudentGroup sg = new StudentGroup();

System.out.println("Enter count student: ");

int n = sc.nextInt();

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

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

String name = scL.nextLine();

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

String adress = scL.nextLine();

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

String facultet = scL.nextLine();

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

int kurs = sc.nextInt();

int []marks = new int[4];

System.out.println("Enter marks");

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

marks[i] = sc.nextInt();

Students s = new Students(name,adress,facultet,kurs,marks);

sg.add(s);

}

System.out.println("Who delate?");

String f = scL.nextLine();

sg.del(f);

sg.print();

}

}

//CLASS STUDENT

class Students {

String name, adress, facultet;

int kurs;

int []marks;

Students(String name, String adress, String facultet, int kurs, int []marks){

this.name = name;

this.adress = adress;

this.facultet = facultet;

this.kurs = kurs;

this.marks = marks;

}

String getName() {return name;}

String getAdress() {return adress;}

String getFacultet() {return facultet;}

int getKurs() {return kurs;}

int[] getMarks() {return marks;}

void setName(String name) {this.name = name;}

void setAdress(String adress) {this.adress = adress;}

void setFacultet(String facultet) {this.facultet = facultet;}

void setKurs(int kurs) {this.kurs = kurs;}

void setMarks(int []marks) {this.marks = marks;}

double sr(){

double s;

s = 0;

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

s += marks[i];

return s/marks.length;

}

public String toString(){

String s = " ";

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

s += String.valueOf(marks[i]);

s+=" ";

}

return "Name: " + name + "\n Adress: " + adress + "\n Facultet: " + facultet + "\n Kurs: " + kurs + "\n Marks" + s;

}

}

//CLASS  
class StudentGroup {

ArrayList<Students>a;

StudentGroup()

a = new ArrayList<Students>();

void add(Students s)

a.add(s);

void del(String name){

int p = -1;

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

if(name.equalsIgnoreCase(a.get(i).getName()))

p = i;

}

if(p>-1){

a.remove(p);

System.out.println("delate student:" + name);

}

else System.out.println("student no");

}

void print(){

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

System.out.println(a.get(i).toString());

}

}

}

1. LIST

//MAIN

public class Laba\_13 {

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

List<Integer>list=new List<>();

int n=sc.nextInt();

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

list.add(sc.nextInt());

list.print();

System.out.println();

System.out.println("Remove element: ");

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

list.remove(sc.nextInt());

list.print();

}

}

}

//CLASS LIST

public class List <Item>{

private int kol;

private ListItem<Item> first;

private ListItem<Item> last;

List(){

kol=0;

first=null;

last=null;

}

List(Item data){

first=last= new ListItem<Item>(data);

kol=1;

}

public void add(Item data){

ListItem<Item> li=new ListItem<Item>(data);

li.setnext(null);

if(kol>0){

li.setprev(last);

last.setnext(li);

}

else

first=li;

last=li;

kol++;

}

public ListItem<Item> getfirst()

return first;

public ListItem<Item> getlast()

return last;

public int getquan()

return kol;

public ListItem<Item>get(int i){

if(i>kol)

return null;

else{

ListItem<Item>li=first;

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

li=li.getnext();

return li;

}

}

void print(){

ListItem<Item>li=first;

while(li!=null){

System.out.print(li.getdata());

li=li.getnext();

}

}

public void remove(int i){

if(i<=kol){

ListItem<Item> li=get(i);

ListItem<Item> prev=li.getprev();

ListItem<Item> next=li.getnext();

if(prev!=null)

prev.setnext(next);

if(next!=null){

next.setprev(prev);

if(i==1)

first=next;

}

kol--;

}

}

}

//CLASS LISTITEM

public class ListItem <Item>{

private Item data;

private ListItem<Item>next;

private ListItem<Item>prev;

ListItem(Item d){

data=d;

next=null;

prev=null;

}

public Item getdata()

return data;

public void setdata(Item d)

data=d;

public ListItem<Item> getprev()

return prev;

public ListItem<Item> getnext()

return next;

public void setprev(ListItem<Item>li)

prev=li;

public void setnext(ListItem<Item>li)

next=li;

}

1. LINCKEDLIST

//MAIN

public class Laba\_14 {

public static void main(String[] args) {

MyList l = new MyList();

l.createlist();

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

l.print();

System.out.println();

System.out.println("List reverse: ");

l.print\_reverse();

l.del("111");

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

l.print();

System.out.println("Simmetryc: " + l.simm());

System.out.println("Delate first end: ");

l.delEnd("222");

l.print();

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

l.dubl();

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

l.dobavl();

}

}

//CLASS MYLIST

public class MyList {

LinkedList<String> list;

MyList()

list= new LinkedList<String>();

void createlist(){

String s;

Scanner sc = new Scanner(System.in);

s = sc.nextLine();

while(!s.equals("")){

list.addLast(s);

s = sc.nextLine();

}

}

void print(){

ListIterator<String> it = list.listIterator();

while(it.hasNext()){

System.out.println(it.next());

}

}

void print\_reverse(){

ListIterator<String> it = list.listIterator(list.size());

while(it.hasPrevious()){

System.out.println(it.previous());

}

}

void del(String s){

ListIterator<String> it = list.listIterator();

while(it.hasNext()){

if(it.next().equals(s))

it.remove();

}

}

boolean simm(){

ListIterator<String> it = list.listIterator();

ListIterator<String> reit = list.listIterator(list.size());

while(it.hasNext() && reit.hasPrevious()){

if(!it.next().equals(reit.previous()))

return false;

}

return true;

}

void delEnd(String f){ //???

ListIterator<String> it = list.listIterator(list.size());

while(it.hasPrevious()){

if(it.previous().equals(f))

it.remove();

}

}

void dubl(){

ListIterator<String> it = list.listIterator();

while(it.hasNext()){

String temp = it.next();

it.add(temp);

}

System.out.println(list);

}

void dobavl(){

ListIterator<String> it = list.listIterator();

while(it.hasNext()){

System.out.println(it.next());

}

ListIterator<String> reit = list.listIterator(list.size());

while(reit.hasPrevious()){

System.out.println(reit.previous());

}

}

}

1. SET

//MAIN

public class Laba\_15 {

public static void main(String[] args) {

TreeSet <Integer> set = new TreeSet<Integer>();

MySet myset = new MySet(set);

myset.fillSet();

myset.print();

MySet simple = myset.simple();

System.out.println("New set: ");

simple.print();

TreeSet<Integer> s = new TreeSet<Integer>();

MySet ms = new MySet(s);

ms.fillSet();

MySet union = myset.union(ms);

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

union.print();

MySet notsimple = myset.notsimple();

System.out.println("New set: ");

notsimple.print();

MySet peresech = myset.peresech(ms);

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

peresech.print();

MySet razn = myset.razn(ms);

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

razn.print();

}

}

//CLASS MYSET

public class MySet {

TreeSet<Integer> set;

MySet(TreeSet<Integer>set)

this.set=set;

TreeSet<Integer> getSet()

return set;

void fillSet(){

Scanner sc = new Scanner(System.in);

System.out.println("Enter elements set:");

int x = sc.nextInt();

while(x!=0){

set.add(x);

x = sc.nextInt();

}

}

void print(){

Iterator it = set.iterator();

System.out.println("My set: ");

while(it.hasNext())

System.out.println(it.next()+" ");

System.out.println();

}

MySet simple(){

TreeSet<Integer> set1 = new TreeSet<Integer>(set);

Iterator it = set1.iterator();

while(it.hasNext()){

int k = 0;

Integer x = (Integer)it.next();

for(int i = 1;i<=x;i++)

if(x%i==0)

k++;

if(k==2)

it.remove();

}

return new MySet(set1);

}

MySet union(MySet a){

TreeSet<Integer> set1 = a.getSet();

TreeSet<Integer> res = new TreeSet<Integer>(set);

Iterator it = set1.iterator();

while(it.hasNext())

res.add((Integer)it.next());

return new MySet(res);

}

MySet notsimple(){

TreeSet<Integer> set1 = new TreeSet<Integer>(set);

Iterator it = set1.iterator();

while(it.hasNext()){

int k = 0;

Integer x = (Integer)it.next();

for(int i = 1;i<=x;i++)

if(x%i==0)

k++;

if(k>2)

it.remove();

}

return new MySet(set1);

}

MySet peresech(MySet a){

TreeSet<Integer> set1 = a.getSet();

TreeSet<Integer> res = new TreeSet<>();

Iterator it = set1.iterator();

while(it.hasNext()){

Integer x = (Integer)it.next();

if(set.contains(x))

res.add(x);

}

return new MySet(res);

}

MySet razn(MySet a){

TreeSet<Integer> set1 = a.getSet();

TreeSet<Integer> res = new TreeSet<Integer>(set);

for(Integer integer:set1)

if(set1!=res)

res.remove(integer);

return new MySet(res);

}

}

1. INTERFACE

//MAIN

public class Laba\_16 {

public static void main(String[] args) {

MyFrame frame = new MyFrame();

}

}

//MYFRAME

public class MyFrame extends JFrame{

MyFrame(){

setSize(850,300);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

MyPanel panel = new MyPanel();

add(panel);

setVisible(true);

}

}

//MYPANEL

class MyPanel extends JPanel{

JLabel l1,l2,l3;

JTextField f1,f2,f3;

JButton button;

MyPanel(){

l1 = new JLabel("1");

add(l1);

f1 = new JTextField();

f1.setColumns(10);

add(f1);

l2 = new JLabel("2");

add(l2);

f2 = new JTextField();

f2.setColumns(10);

add(f2);

l3 = new JLabel("Summa");

add(l3);

f3 = new JTextField();

f3.setColumns(10);

add(f3);

button = new JButton("Enter");

MyListener m = new MyListener();

button.addActionListener(m);

add(button);

}

class MyListener implements ActionListener {

public void actionPerformed(ActionEvent e) {

int s1 = Integer.parseInt(f1.getText());

int s2 = Integer.parseInt(f2.getText());

int s3 = s1+s2;

f3.setText(String.valueOf(s3));

}