

```

public class Test {

    // metodi di supporto
    public static void stampa(String []a){
        // prerequisito: a!=null
        for (int i=0;i<a.length;i++)
            System.out.println(a[i]);
        //un newline per separare
        System.out.println();
    }

    public static void scambia(String [] a, int i, int j) {
        // prerequisito: a!=null
        // i e j sono indici validi
        String tmp = a[i];
        a[i]=a[j];
        a[j]=tmp;
    }

    public static String [] copia(String [] a) {
        // prerequisito: a!=null
        String [] Copia = new String [a.length];
        for (int i=0; i<a.length; i++)
            if (a[i]!=null)
                Copia[i]= new String(a[i]);
        return(Copia);
    }

    // un valore <> null è max, faccio un ord. crescente
    public static void compatta_null_testa(String [] a) {
        // prerequisito: a!=null
        for (int i=0;i<a.length;i++) {
            for (int j=0;j<a.length-i-1;j++)
                if(a[j]!=null && a[j+1]==null)
                    scambia(a,j,j+1);
        }
    }

    // un valore <> null è min, faccio un ord. decrescente
    public static void compatta_null_coda(String [] a) {
        // prerequisito: a!=null
        for (int i=0;i<a.length;i++) {
            for (int j=0;j<a.length-i-1;j++)
                if(a[j]==null && a[j+1]!=null)
                    scambia(a,j,j+1);
        }
    }

    // versione per un array non compatto
    public static int conta_elementi (String []a) {
        // prerequisito: a!=null
        int conta = 0;
    }
}

```

```

        for(int i=0;i<a.length;i++)
            if (a[i]!=null)
                conta++;
        return (conta);
    }

    public static int conta_elementi_comp (String []a) {
        // prerequisito: a!=null
        int i = 0;
        while (i<a.length && a[i]!= null)
            i++;
        return (i);
    }

    public static boolean inserisci (String []a, String s){
        // prerequisito: a!=null
        boolean inserito = true;
        int n = conta_elementi(a);
        if (n < a.length){
            a[n]=s;
        }
        else inserito = false;
        return (inserito);
    }

    public static boolean elimina (String []a, int pos){
        // prerequisito: a!=null
        boolean eliminato = true;
        if (pos < a.length){
            a[pos]=null;
        }
        else eliminato = false;
        return (eliminato);
    }

    //selezione decrescente in a[0] ci va il max
    // si suppone che il vettore sia compatto
    public static void ordina_sel_dec(String []a){
        // prerequisito: a!=null
        int i,j;
        // posso usare anche conta_elementi_comp
        // che è meglio
        int n=conta_elementi(a);
        for(i=0;i<n-1;i++){
            int max=a[i].length();
            int pos_max=i;

            for(j=i+1;j<n;j++){
                if(a[j].length(>max){
                    pos_max=j;
                    max=a[j].length();
                }
            }
            scambia(a,i,pos_max);
        }
    }

```

```

}

// attenzione agli shortcut
// faccio un doppio ordinamento
// devo ciclare su tutto l'array
public static void ordina_dec_compatta(String []a){
    // prerequisito: a!=null
    int i,j;
    for(i=0;i<a.length;i++){
        for(j=0;j<a.length-i-1;j++){
            if(a[j]==null||a[j+1]!=null&&a[j].length()<a[j+1].length())
                scambia(a,j,j+1);
        }
        // stampa di debug
        // stampa(a);
    }
}
}

```

```

public static void main(String[] args) {
    // TODO Auto-generated method stub
    String []a = new String [] {null, "pippo", "pluto", null};
    System.out.println("** test compattazione null in testa **");
    System.out.println("vettore prima della compattazione");
    stampa(a);
    compatta_null_testa(a);
    System.out.println("vettore dopo la compattazione");
    stampa(a);

    System.out.println("** test compattazione null in coda **");
    a = new String [] {null, null, "pluto", null};
    System.out.println("vettore prima della compattazione");
    stampa(a);
    System.out.println("vettore dopo la compattazione");
    compatta_null_coda(a);
    stampa(a);

    System.out.println("test conta_elementi_comp");
    System.out.println(counta_elementi_comp(a));
    System.out.println();

    System.out.println("** test inserimento elemento");
    a = new String [] {null, "pippo", "pluto", null};
    System.out.println("vettore prima dell'inserimento");
    compatta_null_coda(a);
    stampa(a);
    inserisci(a, "topolino");
    System.out.println("vettore dopo l'inserimento");
    stampa(a);

    System.out.println("** test elimina elemento");
    System.out.println("vettore prima dell'eliminazione");
    stampa(a);
    elimina(a,1);
    System.out.println("vettore dopo l'eliminazione");
    stampa(a);
}

```

```
System.out.println("** test ordinamento con ins/elim ");
a = new String [] {null, "pippo", "plutone", null};
System.out.println("vettore iniziale");
stampa(a);
compatta_null_coda(a);
System.out.println("vettore dopo la compattazione");
stampa(a);
ordina_sel_dec(a);
System.out.println("vettore ordinato");
stampa(a);
System.out.println("inserimento topolino");
inserisci(a, "topolino");
stampa(a);

System.out.println("elimina elemento pos 1");
elimina(a,1);
stampa(a);

// si fa una copia per testare due soluzioni
String [] b = copia(a);

System.out.println("compatta e ordina in due metodi");
compatta_null_coda(a);
ordina_sel_dec(a);
stampa(a);

System.out.println("compatta e ordina in un metodo");

System.out.println("vettore copia");
stampa(b);
ordina_dec_compatta(b);
System.out.println("vettore ordinato");
stampa(b);
}
}
```