```
class Terna {
         int riga;
         int colonna;
         int valore;

        public Terna(int r, int c, int v){
                riga = r;
                colonna = c;
                valore = v;
        }

        public int getRiga(){
                return riga;
        }

        public int getColonna(){
                return colonna;
        }

        public int getValore(){
                return valore;
        }

        public String toString(){
                return (riga +" " + colonna +" " + valore);
        }
}//end class terna


class MatriceCompatta implements MatriceDiInteri {
        private Terna[] info;
        private int dim; // dimensione dell'array
        private int lastValue; //dimensione effettiva dell'array
        //indice dell'ultimo valore inserito

/* costruisce una matrice vuota */
        public MatriceCompatta (int dim, int rig, int col, int
dom) {
                int i;
                        this.info = new Terna[dim];
                        this.info[0] = new Terna(rig,col,dom);
                        for(i=1; i<dim; i++)
                                this.info[i] = new Terna(0,0,0);
                        this.dim = dim;
                        this.lastValue = 0;
        }

        public int numRighe(){
        // post: numero di righe della matrice
```

```
                return this.info[0].riga;
        }

        public int numColonne(){
        // post: numero di colonne della matrice
                return this.info[0].colonna;
        }

        public int getLastValue(){
        // post: dimensioni effettive della matrice
                return this.lastValue;
        }


         public boolean isEmpty() {
        //post: restituisce true sse la matrice ha dimensioni
nulle
                return (this.info[0].riga==0 &&
this.info[0].colonna==0);
        }


        public int accedi(int r, int c){
        //pre: r,c>=0
        //post: restituisce l'lemento in posizione r,c
                int i,res;

                i =1;
                while (info[i].riga<r && i<this.lastValue)
                        i++;
                //esco sulla riga r
                while (info[i].riga == r && info[i].colonna < c &&
i<this.lastValue)
                        i++;
                //esco sulla colonna r se esiste
                if (info[i].riga == r && info[i].colonna == c)
                        res = info[i].valore;
                else
                        res = info[0].valore;
                return res;
        }


        public void  memorizza(int r, int c, int n){
        //pre: r,c>=0
        //post: memorizza l'elemento n in posizione r,c
        // se c'e' spazio

        /* se si sostituisce un elemento diverso dal dominante con
uno dominante,
        si devono shiftare tutti gli elementi in alto */
                int i,j;
```

```
                i=1;
                /* vedo se la componente da memorizzare esiste
gia' */
                while (info[i].riga<r && i<=this.lastValue)
                        i++;
                //esco sulla riga r
                while (info[i].riga == r && info[i].colonna < c &&
i<=this.lastValue)
                        i++;
                /* se il valore da memorizzare e' il dominante e
se esiste gia' un valore nella
                  matrice, allora devo eliminare la componente */
                if (n==info[0].valore){
                        if (info[i].riga == r && info[i].colonna
== c){ //elimino l'el. in posizione i
                                for (j=i+1; j<this.dim; j++)
                                        info[j-1] = info[j];
//sposto tutto l'oggetto Terna
                                this.lastValue = this.lastValue -
1;
                        }
                        /* se il valore da memorizzare e' il
dominante e se non esiste gia' un valore nella
                       matrice, allora non devo fare niente */
                }
                else {//il val. da memorizzare non e' dominante:
se gia' esiste sovrascrivo altrimenti aggiungo
                        if (info[i].riga == r && info[i].colonna
== c) // sovrascrivo
                                info[i].valore = n;
                        else //se c'e' posto aggiungo un elemento
                                if (this.lastValue == this.dim-1)
                                        System.out.println("**
Memorizzazione impossibile **");
                                else{
                                        for( j = this.lastValue;
j>= i; j--)
                                                this.info[j+1] =
this.info[j];
                                        this.lastValue =
this.lastValue + 1;
                                        info[i] = new
Terna(r,c,n);
                                }

                }
        }


/*************************************************************/
        public String toString() {
```

```
                int i;
                String res;
                res = "";
                for (i=0; i<=lastValue; i++)
                        res = res +(info[i].toString()) + "\n";
                return res;
        }

}



class MatriceConArray implements MatriceDiInteri {
        private int[][] mat;
        private int nrighe;
        private int ncolonne;

        public MatriceConArray (int r, int c) {
                int i,j;
          this.mat = new int[r][c]; //costruttore
          //inizializza a zero;
          for (i=0; i<r; i++)
                for(j=0; j<c; j++)
                        this.mat[i][j] = 0;
          this.nrighe = r;
          this.ncolonne = c;
        }

        public int numRighe(){
        // post: numero di righe della matrice
                return this.nrighe;
        }

        public int numColonne(){
        // post: numero di colonne della matrice
                return this.ncolonne;
        }


         public boolean isEmpty() {
        //post: restituisce true sse la matrice ha dimensioni
nulle
                return (this.nrighe==0 && this.ncolonne==0);
        }


        public int accedi(int r, int c){
        //pre: r,c>=0
        //post: restituisce l'lemento in pos izione r,c
                return this.mat[r][c];
        }
```

```
        public void  memorizza(int r, int c, int n){
        //pre: r,c>=0
        //post: memorizza l'elemento n in posizione r,c
                this.mat[r][c] = n;
        }

        public String toString(){
                int i,j;
                String res = "";
                for (i=0; i<nrighe; i++){
                        for(j=0; j<ncolonne; j++)
                                res = res + mat[i][j] +" ";
                        res = res +"\n";
                }
                return res;
        }

}
interface MatriceDiInteri {
        boolean isEmpty();
  //post: restituisce true sse la matrice e'  vuota r=c=0

        int numRighe();
        // post: numero di righe della matrice

        int numColonne();
        // post: numero di colonne della matrice

        int accedi(int r, int c);
        //pre: r,c>=0
        //post: restituisce l'lemento in pos izione r,c

        void memorizza(int r, int c, int n);
        //pre: r,c>=0
        //post: memorizza l'elemento n in posizione r,c
}

class UsaMatrici{

        public static void main(String[] args){
        /*MatriceConArray emme;

        emme = new MatriceConArray(4,6);
        System.out.println(emme.toString());
        emme.memorizza(1,2,9);
        emme.memorizza(3,4,7);
        emme.memorizza(2,2,8);
        System.out.println(emme.toString());
 */
```

```
        MatriceCompatta emme;
        emme = new MatriceCompatta(5,4,6,0);
        System.out.println(emme.toString());
        System.out.println("inserisco in riga 1, colonna 2 il
valore 9");
        emme.memorizza(1,2,9);
        System.out.println(emme.toString());
        System.out.println("inserisco in riga 3, colonna 4 il
valore 7");
        emme.memorizza(3,4,7);
        System.out.println(emme.toString());
        System.out.println("inserisco in riga 2, colonna 2 il
valore 8");
        emme.memorizza(2,2,8);
        System.out.println(emme.toString());
        /*System.out.println("inserisco in riga 2, colonna 2 il
valore 0");
        emme.memorizza(2,2,0);
        System.out.println(emme.toString());
        System.out.println("inserisco in riga 2, colonna 2 il
valore 8");
    emme.memorizza(2,2,8);
    System.out.println("inserisco in riga 2, colonna 3 il valore
5");
    emme.memorizza(2,3,5);
    System.out.println("inserisco in riga 3, colonna 3 il valore
7");
    emme.memorizza(3,3,7);
        System.out.println(emme.toString());*/
}
}
```