```java
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.PrintStream;
import java.util.ArrayList;

public class Matrices {

        public final static String ERROR_FILAS   = "ERROR1: Todas las filas de la matriz deben tener la misma dimension";
        public final static String ERROR_INVERSA = "ERROR2: La operacion inversa() requiere una matriz cuadrada";
        public final static String ERROR_ADJUNTA = "ERROR3: La operacion adjunta() requiere una matriz cuadrada";
        public final static String ERROR_PROD    = "ERROR4: La operacion producto() requiere coincidencia de filas y columnas";
        public final static String ERROR_SUMA    = "ERROR5: La operacion suma() requiere matrices de iguales dimensiones";

        /**
         * Devuelve la inversa de una matriz
         * @param matriz
         * @return
         */
    public static double[][] inversa(double[][] matriz) {
        double det=1/determinante(matriz);
        double[][] nmatriz=adjunta(matriz);
        nmatriz = producto(det,nmatriz);
        return nmatriz;
    }

    /**
     * Producto de un escalar por una matriz
     * @param n
     * @param matriz
     */
    public static double[][] producto(double n, double[][] matriz) {
        double[][]  salida = new double[matriz.length][matriz[0].length];
        for(int i=0;i<matriz.length;i++)
            for(int j=0;j<matriz[0].length;j++)
                salida[i][j] = n*matriz[i][j];
        return salida;
    }

    /**
     * Producto de dos matrices
     * @param matriz1
     * @param matriz2
     */
    public static double[][] producto(double[][] matriz1, double[][] matriz2) {
        double[][] salida = new double[matriz1.length][matriz2[0].length];
        for(int i=0;i<matriz1.length;i++)
            for(int j=0;j<matriz2[0].length;j++) {
                salida[i][j] = 0;
                        for(int k=0;k<matriz1[0].length; k++)
                                salida[i][j] += matriz1[i][k] * matriz2[k][j];
            }
        return salida;
    }

```

```java
    /**
     * Suma de un escalar por una matriz
     * @param n
     * @param matriz
     */
    public static double[][] suma(double n, double[][] matriz) {
        double[][] salida = new double[matriz.length][matriz[0].length];
        for(int i=0;i<matriz.length;i++)
            for(int j=0;j<matriz[0].length;j++)
                salida[i][j] = n+matriz[i][j];
        return salida;
    }

    /**
     * Suma de dos matrices
     * @param matriz1
     * @param matriz2
     */
    public static double[][] suma(double[][] matriz1, double[][] matriz2) {
        double[][] salida = new double[matriz1.length][matriz1[0].length];
        for(int i=0;i<matriz1.length;i++)
            for(int j=0;j<matriz2[0].length;j++)
                salida[i][j] = matriz1[i][j] + matriz2[i][j];
        return salida;
    }

    /** Matriz adjunta de otra matriz
     * @param matriz
     * @return
     */
    public static double[][] adjunta(double [][] matriz){
        return transpuesta(cofactores(matriz));
    }

    /**
     * Matriz de cofactores
     * @param matriz
     * @return
     */
    public static double[][] cofactores(double[][] matriz){
        double[][] nm=new double[matriz.length][matriz.length];
        for(int i=0;i<matriz.length;i++) {
            for(int j=0;j<matriz.length;j++) {
                double[][] det=new double[matriz.length-1][matriz.length-1];
                double detValor;
                for(int k=0;k<matriz.length;k++) {
                    if(k!=i) {
                        for(int l=0;l<matriz.length;l++) {
                            if(l!=j){
                                int indice1=k<i ? k : k-1 ;
                                int indice2=l<j ? l : l-1 ;
                                det[indice1][indice2]=matriz[k][l];
                            }
                        }
                    }
                }
                detValor=determinante(det);
                nm[i][j]=detValor * (double)Math.pow(-1, i+j+2);
            }
```

```
114         }
115         return nm;
116     }
117 
118     /**
119      * Matriz transpuesta
120      * @param matriz
121      * @return
122      */
123     public static double[][] transpuesta(double [][] matriz){
124         double[][]salida=new double[matriz[0].length][matriz.length];
125         for(int i=0; i<matriz[0].length; i++){
126             for(int j=0; j<matriz.length; j++)
127                 salida[i][j]=matriz[j][i];
128         }
129         return salida;
130     }
131 
132     /**
133      * Determinante de una matriz
134      * @param matriz
135      * @return
136      */
137     public static double determinante(double[][] matriz){
138         double det;
139         if(matriz.length==2){
140             det=(matriz[0][0]*matriz[1][1])-(matriz[1][0]*matriz[0][1]);
141             return det;
142         }
143         double suma=0;
144         for(int i=0; i<matriz.length; i++){
145         double[][] nm=new double[matriz.length-1][matriz.length-1];
146             for(int j=0; j<matriz.length; j++){
147                 if(j!=i){
148                     for(int k=1; k<matriz.length; k++){
149                         int indice=-1;
150                         if(j<i)
151                             indice=j;
152                         else if(j>i)
153                             indice=j-1;
154                             nm[indice][k-1]=matriz[j][k];
155                     }
156                 }
157             }
158             if(i%2==0)
159                 suma+=matriz[i][0] * determinante(nm);
160             else
161                 suma-=matriz[i][0] * determinante(nm);
162         }
163         return suma;
164     }
165 
166     /**
167      * Imprime un escalar
168      * @param n
169      */
170     public static void print(double n) {
171         System.out.println(String.format("%4.2f", n));
172     }
```

```
173
174     /**
175      * Imprime una matriz
176      * @param matriz
177      */
178     public static void print(double[][] matriz) {
179         for(int i=0; i<matriz.length; i++) {
180             for(int j=0; j<matriz[0].length; j++)
181                 System.out.print(String.format("%7.2f",matriz[i][j]));
182             System.out.println();
183         }
184     }
185
186     /**
187      * Imprime una matriz
188      * @param matriz
189      */
190     public static void print(ArrayList<ArrayList<Double>> arrayList) {
191         print(toArray(arrayList));
192     }
193
194     /**
195      * Numero de filas de una matriz
196      * @param matriz
197      */
198     public static int filas(double[][] matriz) {
199         return matriz.length;
200     }
201
202     /**
203      * Numero de columnas de una matriz
204      * @param matriz
205      */
206     public static int columnas(double[][] matriz) {
207         return matriz[0].length;
208     }
209
210     /**
211      * Convierte un ArrayList<ArrayList<Double>> en una matriz de tipo
    double[][]
212      * @param matriz
213      */
214     public static double[][] toArray(ArrayList<ArrayList<Double>> arrayList) {
215         int nFilas = arrayList.size();
216         int nColumnas = arrayList.get(0).size();
217         double[][] salida = new double[nFilas][nColumnas];
218         for(int i=0; i<nFilas; i++) {
219                 ArrayList<Double> fila = arrayList.get(i);
220                 for(int j=0; j<nColumnas; j++) {
221                         salida[i][j] = fila.get(j);
222                 }
223         }
224         return salida;
225     }
226
227     /**
228      * Convierte un una matriz de tipo double[][] en un
    ArrayList<ArrayList<Double>>
229      * @param matriz
```

```java
     */
    public static ArrayList<ArrayList<Double>> toArrayList(double[][] matriz) {
        ArrayList<ArrayList<Double>> arrayList = new ArrayList();
        int nFilas = filas(matriz);
        int nColumnas = columnas(matriz);
        for(int i=0; i<nFilas; i++) {
                ArrayList<Double> fila = new ArrayList();
                for(int j=0; j<nColumnas; j++) {
                        fila.add(matriz[i][j]);
                }
                arrayList.add(fila);
        }
        return arrayList;
    }

    public static boolean DEBUG = false;

    public static void main(String argv[]) {
        try {
          parser p = null;
          int i=0;
          while (i<argv.length) {
              if ("+v".equals(argv[i])) {
                  DEBUG = true;
              } else  {
                  p = new parser(new Yylex(new FileReader(argv[i])));
              }
              i++;
          }
          if (p!=null) {
              p.parse();
          }
        } catch (Exception e) {
          e.printStackTrace();
        }
      }


}
```