```java
import java.util.Scanner;

public class InverseMatrix {
    public static void main(String argv[]) {
        Scanner input = new Scanner(System.in);
        System.out.println("Enter the dimension of matrix: ");
        int n = input.nextInt();
        double a[][]= new double[n][n];
        System.out.println("Enter the elements of matrix: ");
        for(int i=0; i<n; i++)
            for(int j=0; j<n; j++)
                a[i][j] = input.nextDouble();

        double d[][] = invert(a);

        System.out.println("The inverse is: ");
        for (int i=0; i<n; ++i) {
            for (int j=0; j<n; ++j) {
                System.out.print(d[i][j]+"  ");
            }
            System.out.println();
        }
        input.close();
    }

    public static double[][] invert(double a[][]) {
        int n = a.length;
        double x[][] = new double[n][n];
        double b[][] = new double[n][n];
        int index[] = new int[n];
        for (int i=0; i<n; ++i)
            b[i][i] = 1;

        // Transform the matrix into an upper triangle
        gaussian(a, index);

        // Update the matrix b[i][j] with the ratios stored
        for (int i=0; i<n-1; ++i)
            for (int j=i+1; j<n; ++j)
                for (int k=0; k<n; ++k)
                    b[index[j]][k]
```

```
42                              -= a[index[j]][i]*b[index[i]][k];
43
44          // Perform backward substitutions
45          for (int i=0; i<n; ++i) {
46              x[n-1][i] = b[index[n-1]][i]/a[index[n-1]][n-1];
47              for (int j=n-2; j>=0; --j) {
48                  x[j][i] = b[index[j]][i];
49                  for (int k=j+1; k<n; ++k) {
50                      x[j][i] -= a[index[j]][k]*x[k][i];
51                  }
52                  x[j][i] /= a[index[j]][j];
53              }
54          }
55          return x;
56      }
57
58      // Carry out the partial-pivoting Gaussian elimination.
59      public static void gaussian(double a[][], int index[]) {
60          int n = index.length;
61          double c[] = new double[n];
62
63          // Initialize the index
64          for (int i=0; i<n; ++i)
65              index[i] = i;
66
67          // Find the rescaling factors, one from each row
68          for (int i=0; i<n; ++i) {
69              double c1 = 0;
70              for (int j=0; j<n; ++j) {
71                  double c0 = Math.abs(a[i][j]);
72                  if (c0 > c1) c1 = c0;
73              }
74              c[i] = c1;
75          }
76
77          // Search the pivoting element from each column
78          int k = 0;
79          for (int j=0; j<n-1; ++j) {
80              double pi1 = 0;
81              for (int i=j; i<n; ++i) {
82                  double pi0 = Math.abs(a[index[i]][j]);
```

```
83                  pi0 /= c[index[i]];
84                  if (pi0 > pi1) {
85                      pi1 = pi0;
86                      k = i;
87                  }
88              }
89
90              // Interchange rows according to the pivoting order
91              int itmp = index[j];
92              index[j] = index[k];
93              index[k] = itmp;
94              for (int i=j+1; i<n; ++i) {
95                  double pj = a[index[i]][j]/a[index[j]][j];
96
97                  // Record pivoting ratios below the diagonal
98                  a[index[i]][j] = pj;
99
100                 // Modify other elements accordingly
101                 for (int l=j+1; l<n; ++l)
102                     a[index[i]][l] -= pj*a[index[j]][l];
103             }
104         }
105     }
106 }
```