Android Studio

West University of Timisoara, Romania
Computer Science Department
IE3, Fall 2015
Dr. Liviu Octavian Mafteiu-Scai

Working with files

Solving linear equations systems using mobile devices

FileOutputStream: an output stream that writes bytes to a file. If the output file exists, it can be replaced or appended to. If it does not exist, a new file will be created.

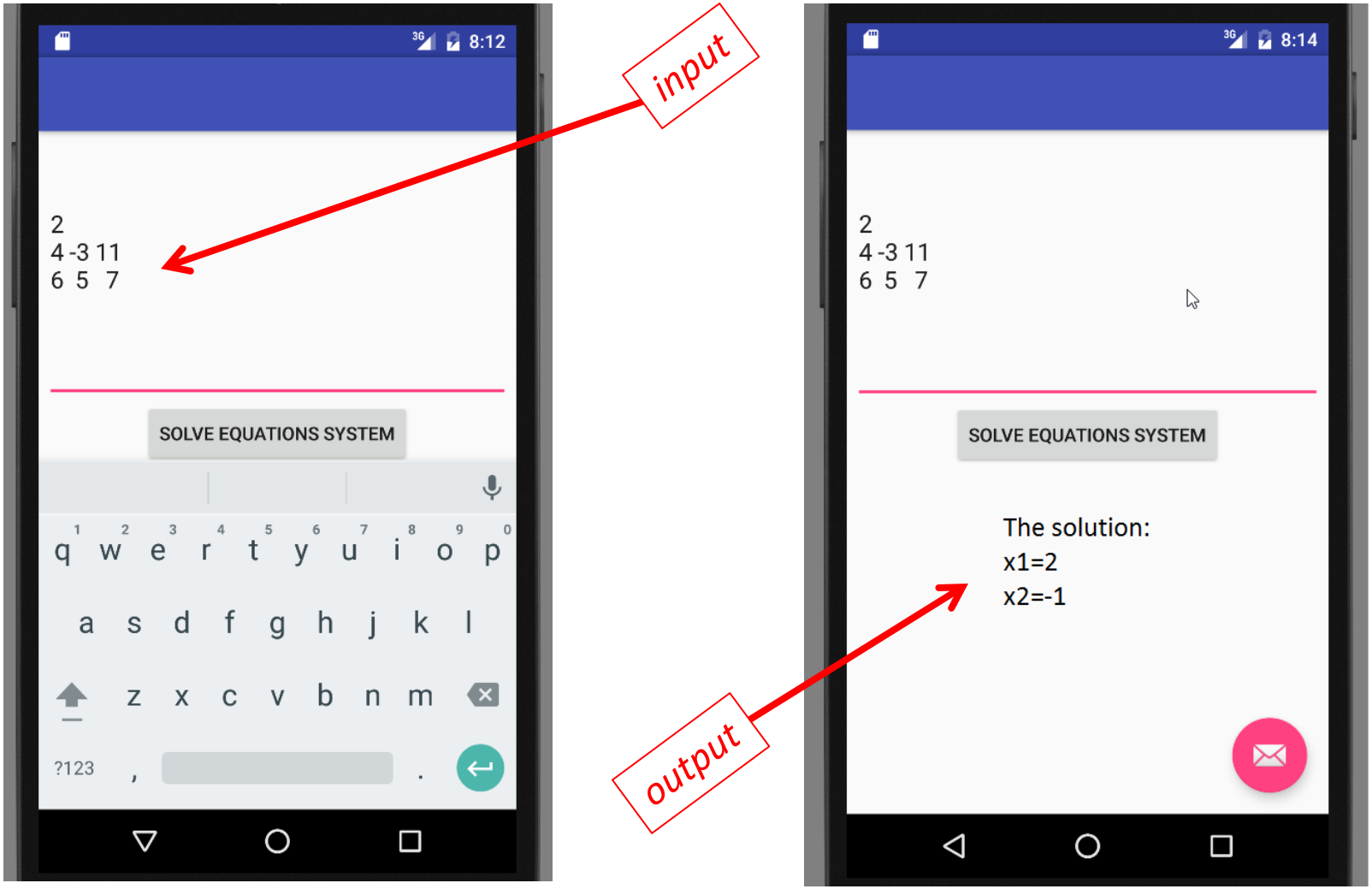
openFileOutput: read/write from a text file

The Context constant **MODE_PRIVATE** makes the file inaccessible to other apps

OutputStreamWriter: a class for turning a character stream into a byte stream. It contains a buffer of 8 Kbytes to be written to target stream and converts these into characters as needed.

FileInputStream: an input stream for read file.

InputStreamReader: a class for turning a byte stream into a character stream. The buffer size is 8K.


8:12
2
4 -3 11
6 5 7
SOLVE EQUATIONS SYSTEM
input
output
8:14
2
4 -3 11
6 5 7
SOLVE EQUATIONS SYSTEM
The solution:
x1=2
x2=-1

1. Implement solving systems of equations using Cramer's rule (2x2 and 3x3 system)

2. The same for Gaussian elimination (nxn system)

The theory is presented in next slides

Cramer's Rule 2x2 system

ax + by = e

cx + dy = f

x=(ed-bf)/(ad-bc)

y=(af-ec)/(ad-bc)

$$\det(A) = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}$$

Sarrus

$$= a_{11}a_{22}a_{33} + a_{21}a_{32}a_{13} + a_{31}a_{12}a_{23} - a_{11}a_{32}a_{23} - a_{21}a_{12}a_{33} - a_{31}a_{22}a_{13}.$$

Cramer's Rule 3x3 system

$$\begin{aligned} a_1x + b_1y + c_1z &= d_1 \\ a_2x + b_2y + c_2z &= d_2 \\ a_3x + b_3y + c_3z &= d_3 \end{aligned}$$

with

$$D = \begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix} \neq 0 \quad D_x = \begin{vmatrix} d_1 & b_1 & c_1 \\ d_2 & b_2 & c_2 \\ d_3 & b_3 & c_3 \end{vmatrix} \quad D_y = \begin{vmatrix} a_1 & d_1 & c_1 \\ a_2 & d_2 & c_2 \\ a_3 & d_3 & c_3 \end{vmatrix} \quad D_z = \begin{vmatrix} a_1 & b_1 & d_1 \\ a_2 & b_2 & d_2 \\ a_3 & b_3 & d_3 \end{vmatrix}$$

then the solution of this system is:

$$x = \frac{D_x}{D} \qquad y = \frac{D_y}{D} \qquad z = \frac{D_z}{D}$$

! Correct: Gauss-Jordan elimination

Main idea: the main idea is to add or substract linear combination of the given equations until each equations contains only one unknowns, thus giving an immediate solution.

There are three elementary row operations:

- swapping two rows;
- multiplying a row by a non-zero number;
- adding a multiple of one row to another row.

It is obvious that these operations don't change the solution set of the equation system!

Finaly, it is obtaining an upper triangular matrix

Algorithm complexity: $O(n^3)$ in case of a nxn system, ie **very very very big**

The complexity arising from:

$n(n+1) / 2$ divisions +

$(2n^3 + 3n^2 - 5n)/6$ multiplications +

$(2n^3 + 3n^2 - 5n)/6$ subtractions

$= 2n^3 / 3$ operations.

To improve numericsl stability (reduce truncation errors)

=>

pivoting technique (first exchanges rows to move the entry with the largest absolute value to the pivot position)

Mobile App Development 10, Android Studio *Gaussian elimination-the source code java*

```
public class GaussianElimination {
   private static final double EPSILON = 1e-10;

   // Gaussian elimination with partial pivoting
   public static double[] lsolve(double[][] A, double[] b) {
      int N  = b.length;

      for (int p = 0; p < N; p++) {

         // find pivot row and swap
         int max = p;
         for (int i = p + 1; i < N; i++) {
            if (Math.abs(A[i][p]) > Math.abs(A[max][p])) {
               max = i;
            }
         }
         double[] temp = A[p]; A[p] = A[max]; A[max] = temp;
         double   t    = b[p]; b[p] = b[max]; b[max] = t;

         // singular or nearly singular
         if (Math.abs(A[p][p]) <= EPSILON) {
            throw new RuntimeException("Matrix is singular or nearly
singular");
         }

         // pivot within A and b
         for (int i = p + 1; i < N; i++) {
            double alpha = A[i][p] / A[p][p];
            b[i] -= alpha * b[p];
            for (int j = p; j < N; j++) {
               A[i][j] -= alpha * A[p][j];
            }
         }
      }

      // back substitution
      double[] x = new double[N];
      for (int i = N - 1; i >= 0; i--) {
         double sum = 0.0;
         for (int j = i + 1; j < N; j++) {
            sum += A[i][j] * x[j];
         }
         x[i] = (b[i] - sum) / A[i][i];
      }
      return x;
   }


   public static void main(String[] args) {
      int N = 3;
      double[][] A = { { 0, 1,  1 },
                       { 2, 4, -2 },
                       { 0, 3, 15 }
                     };
      double[] b = { 4, 2, 36 };
      double[] x = lsolve(A, b);


      // print results
      for (int i = 0; i < N; i++) {
         System.out.println(x[i]);
      }

   }

}
```

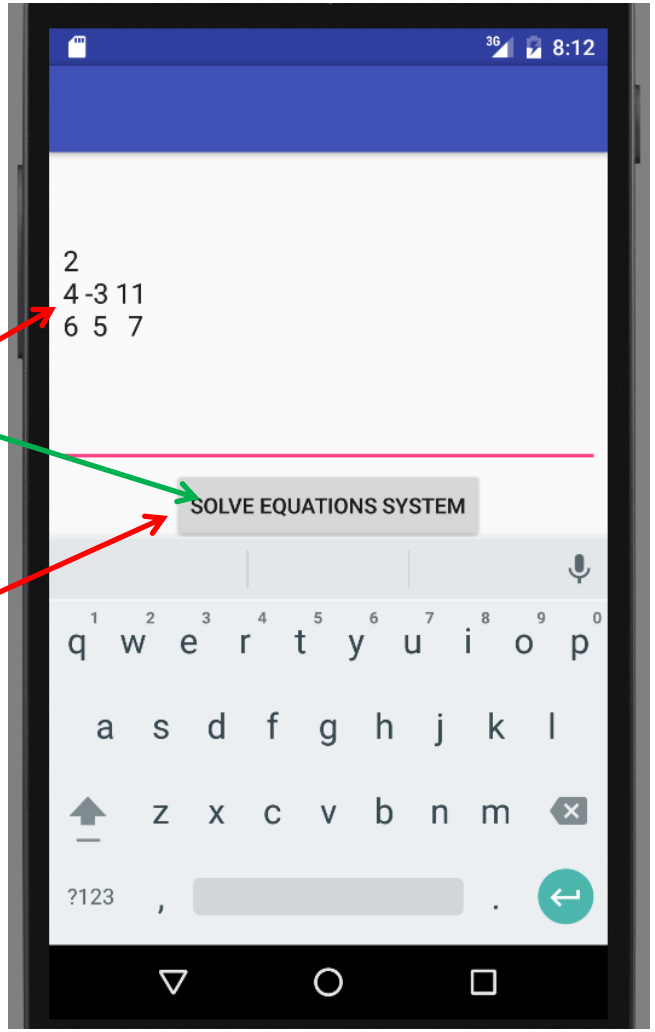
Source:
http://introcs.cs.princeton.edu/java/95linear/GaussianElimination.java.html

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context="{relativePackage}.${activityClass}" >

   <TextView
      android:id="@+id/textView1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentLeft="true"
      android:layout_alignParentRight="true"
      android:background="#008080"
      android:padding="5dp"
      android:text="Solving Linear EquationSystem"
      android:textColor="#fff" />

   <EditText
      android:id="@+id/editText1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentLeft="true"
      android:layout_alignParentRight="true"
      android:layout_below="@+id/textView1"
      android:layout_marginTop="5dp"
      android:ems="10"
      android:layout_marginLeft="5dp"
      android:layout_marginRight="5dp"
      android:layout_above="@+id/button1">
   </EditText>

   <Button
      android:id="@+id/button1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Solve equations system"
      android:onClick="WriteBtn"
      android:layout_centerHorizontal="true"
      android:layout_centerVertical="true" />
</RelativeLayout>
```



Mobile App Development 10, Android Studio ***Working with files –*** MainActivity.java

```
package com.example.mafteiu_scai.myapplication;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;

import android.app.Activity;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends Activity {

   EditText textmsg;
   String matrixtext;  //matrix in string format
   int n;           //system dimension

   int [][] matrix = new int[10][11];       //matrix in numerical format

   static final int READ_BLOCK_SIZE = 100;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      textmsg=(EditText)findViewById(R.id.editText1);
   }
```

```
// write text to file
   public void WriteBtn(View v) {
      // add-write text into file
      try {
         FileOutputStream fileout=openFileOutput("system.txt", MODE_PRIVATE);
         OutputStreamWriter outputWriter=new OutputStreamWriter(fileout);
         outputWriter.write(textmsg.getText().toString());
         outputWriter.write("\nThe solution:\n");
         matrixtext=textmsg.getText().toString();
         // outputWriter.write(matrixtext.charAt(0)); //print first char of Stringmatrixtext
         //outputWriter.write(Integer.toString(matrixtext.length())); //print the length of string matrixtext
         String nstring=""; //string for n value
         int i=0;
         while(matrixtext.charAt(i)!=' ' && matrixtext.charAt(i)!='\n') {
            nstring+=matrixtext.charAt(i);
            i++;
         }
         n=Integer.parseInt(nstring);
        // n=n*n;   //only for test conversion
         nstring=Integer.toString(n);
         outputWriter.write(nstring);
         //convert matrixtext to a numerical matrix : first is the dimension n
         i++;
         for(int l=0;l<n;l++)   //number of matrix lines
         {
            outputWriter.write('\n');
            for(int c=0;c<n;c++)   //Attention: the number of columns must be (n + 1) or put free terms in other
array
            {
               String nelement=""; //string for generic matrix element
               while(matrixtext.charAt(i)!=' '  && matrixtext.charAt(i)!='\n' && matrixtext.charAt(i)!= 0) {
                  nelement+=matrixtext.charAt(i);
                  i++;
               }
               i++;
               //outputWriter.write(nelement);
               matrix[l][c]=Integer.parseInt(nelement);
               nelement=Integer.toString(matrix[l][c]);
               outputWriter.write(nelement+' ');
            }
         }
```

```
        //code for solving equations system

        //convert numerical solution to string solution

        //print solution to output
        //outputWriter.write("x1=\n");


        outputWriter.close();

    } catch (Exception e) {
        e.printStackTrace();
    }
    try {
        FileInputStream fileIn=openFileInput("system.txt");
        InputStreamReader InputRead= new InputStreamReader(fileIn);

        char[] inputBuffer= new char[READ_BLOCK_SIZE];
        String s="";
        int charRead;

        while ((charRead=InputRead.read(inputBuffer))>0) {
            // char to string conversion
            //readstring=String.copyValueOf(inputBuffer,0,charRead);
            String  readstring=String.copyValueOf(inputBuffer,0,charRead);
             s +=readstring;
        }
        InputRead.close();
        Toast.makeText(getBaseContext(), s,Toast.LENGTH_SHORT).show();
        //A toast is a view containing a quick little message for the user.

    } catch (Exception e) {
        e.printStackTrace();
    }
  }
}
```

Ta-Ta for now!