Programming for mobile devices

Lab 2

Phd. Mafteiu-Scai Liviu Octavian

-

Note: All materials (courses, labs and resources):

https://drive.google.com/drive/folders/1mwtVbolPMdtw5k7I8cN2jOrRAIF0BhEC

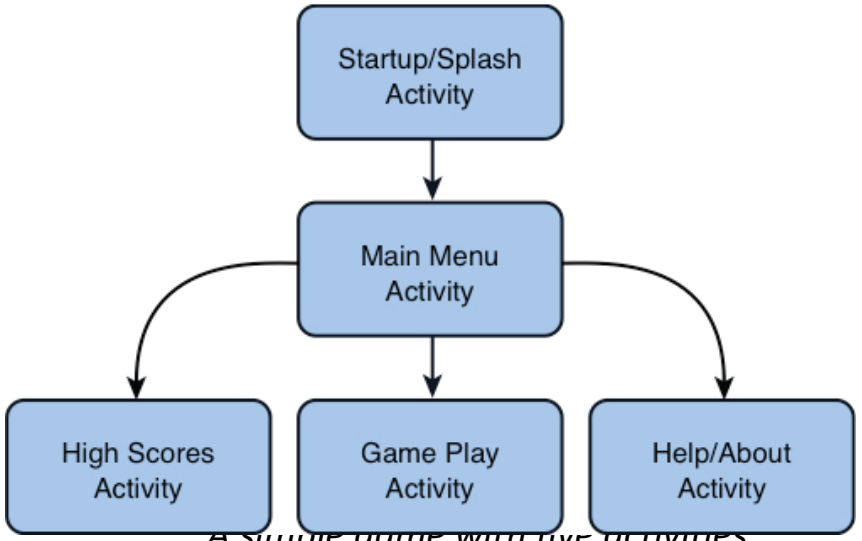
and

https://elearning.e-uvt.ro/course/view.php?id=101631#section-0 (IE2)

https://elearning.e-uvt.ro/course/view.php?id=101524#section-0 (IR2)

Android's Fundamental Components (only what is necessary for the first apps)

- **View**: user interface (UI) elements (buttons, labels, text fields, etc) that form the basic building blocks of a user interface. *Everything you see is a view.*
- **Activity**: is concept that usually represents a single screen in your application.
- **Fragment**: in the case of a large screen, it is difficult to manage all of its functionality in a single activity. *Fragments* are like sub-activities, and an activity can display one or more fragments on the screen at the same time. And a bit close to the concept of function in high-level languages



A simple game with five activities.

.

Android's Fundamental Components (only what is necessary for the first apps)

- **AVD *(Android Virtual Device)***

An AVD represents a device and its configuration. It allows developers to test their applications without using an Android device (smartphone or tablet). With AVD many different types of real devices can be emulated.

About folder structure created by *Android Studio for each project*

For each project, AS creates a complex folder structure.

……

……

the *source* folder contains source code ;

-the *java* folder contains any Java code you write. Any activities you create live here;

-the *res* folder contains system resources. The *layout* folder contains layouts and the *values* folder contains resource files for values such as strings;

-*MainActivity.java* (like any other activity) defines an activity that tells Android how the app should interact with the user and/or computes; It is like *main* function in C language.

-*activity_main.xml* defines a layout that tells Android how the app should look; In general, there is an *.xml* file for each activity.

-*string.xml* file contains strings such as app name and any defaults text values;

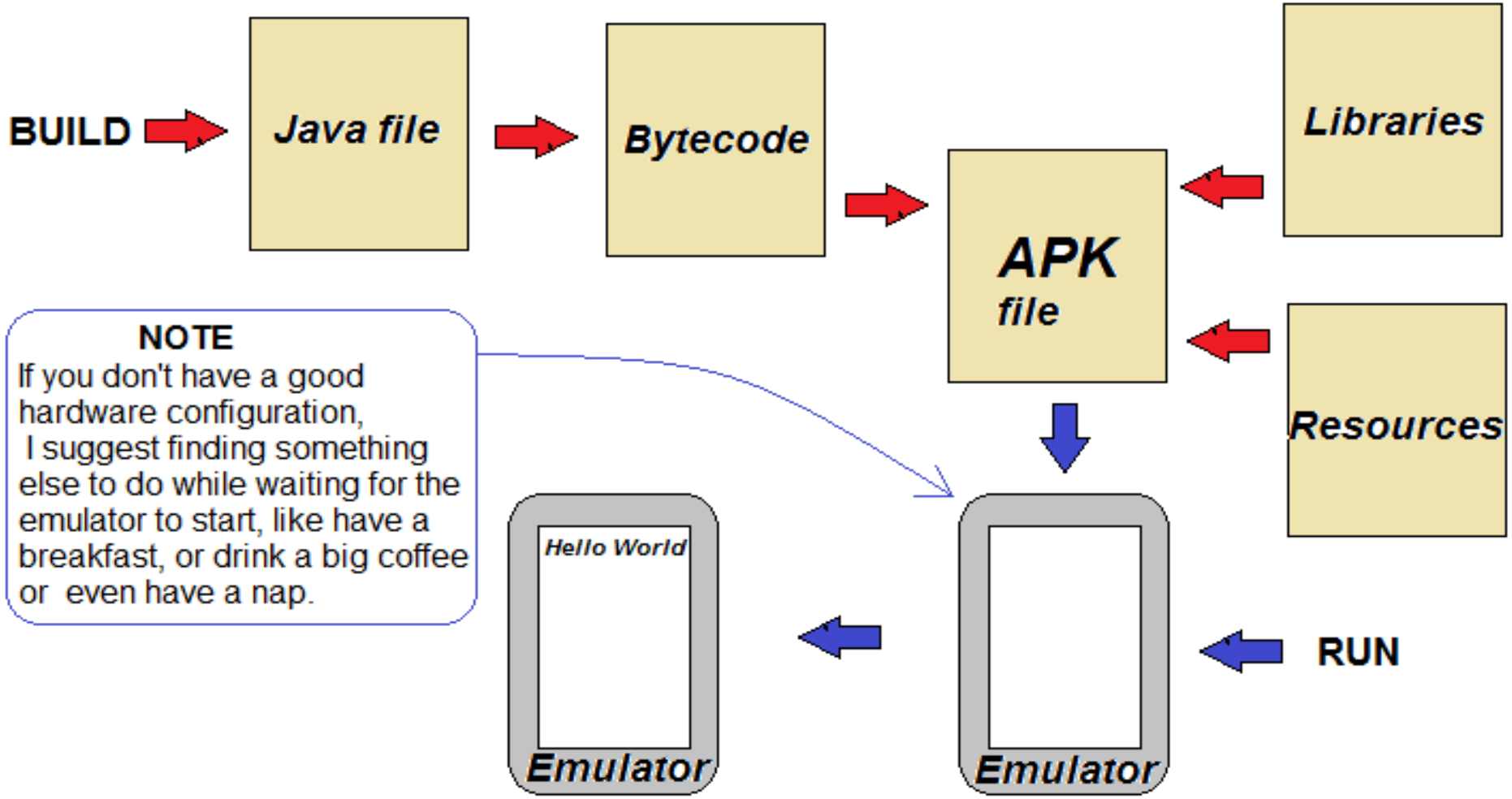
…..

…..

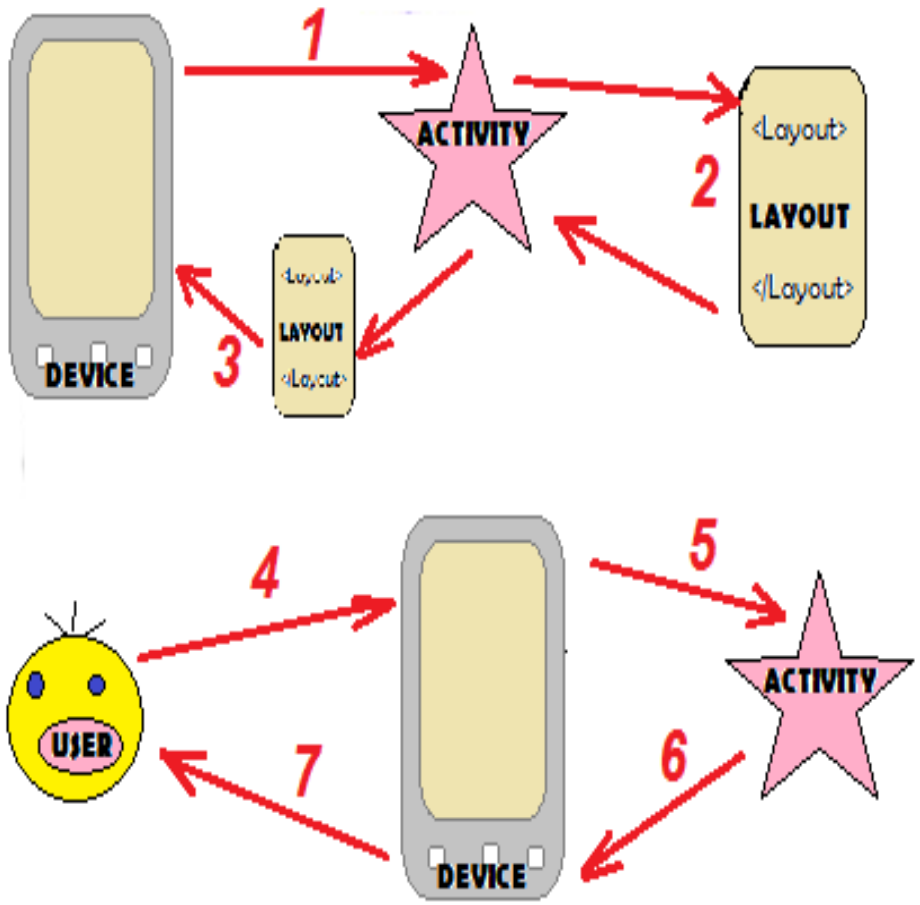
Understanding the Android emulator

Compile, Package, Deploy, Run

An APK file is an Android application package. It's basically a JAR or ZIP file for Android app.



How works an App in Android Studio



1. The device launches your app and creates an activity object.

2. The activity object specifies a layout.

3. The activity tells Android to display the layout on screen.

4. The user interacts with the layout that's displayed on the device.

5. The activity responds to these interactions by running application code.

6. The activity updates the display...

7. ...which the user sees on the device.

Understanding *activity_main.xml* *(some elements)*

```
<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“
             android:paddingLeft="@dimen/activity_horizontal_margin“
             android:paddingRight="@dimen/activity_horizontal_margin"
             android:paddingTop="@dimen/activity_vertical_margin"
             android:paddingBottom="@dimen/activity_vertical_margin"
            tools:context=".MainActivity">

   <TextView
      android:text="@string/hello_world"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"/>

</RelativeLayout>
```

Make the layout the same width and height as the screen size on the device.

Add padding to the screen margins.

Include a `TextView` GUI component for displaying text.

Display the text value of a string resource called `hello_world`.

Make the text wrap horizontally and vertically.

Understanding *MainActivity.java* *(some elements)*

```
package com.my.myfirstapp;

import android.app.Activity;
import android.os.Bundle;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

package name.

Android classes used in `MainActivity`.

`MainActivity` extends the Android class `android.app.Activity`.

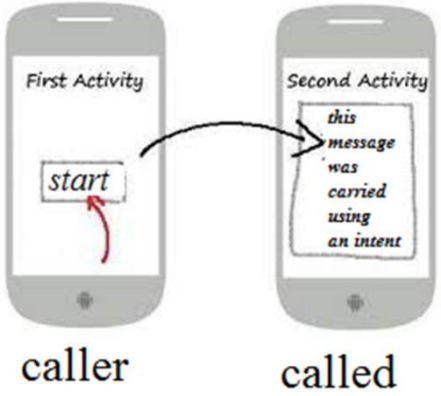
Implement the `onCreate()` method from the `Activity` class. This method is called when the activity is first created.

Specifies which layout to use.

The Anatomy of an Android Application : *Android Activities*

ONE App = one ar more *Activities* linked together, to do one or more tasks

Remember: ***an Activity is not a Task***



Definition: An activity is a single, standalone module of application functionality that usually has a single user interface screen (a view) and its corresponding functionality (layout).

Example: a game:

- an activity screen (main activity) that displays game's scene, score, user's account.. - a second activity (second): a screen where the user types their personal data.

Each activity is implemented as a single class that extends the Android Activity base classes (ex. Activity or AppCompatActivity or…)

The Anatomy of an Android Application : *Android Activities*

In fact, most mobile apps consist of multiple screens.

! For each screen we have an activity !

Example: a text messaging application might have one screen that shows a list of contacts to send messages to, a second screen to write the message to the chosen contact, and other screens to review old messages. Each of these screens would be implemented as an activity.

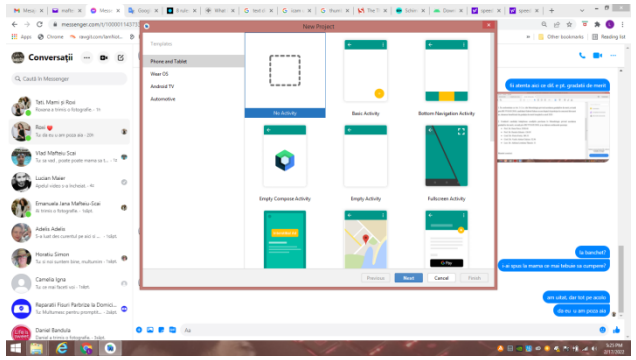
Moving to another screen means starting a new activity.

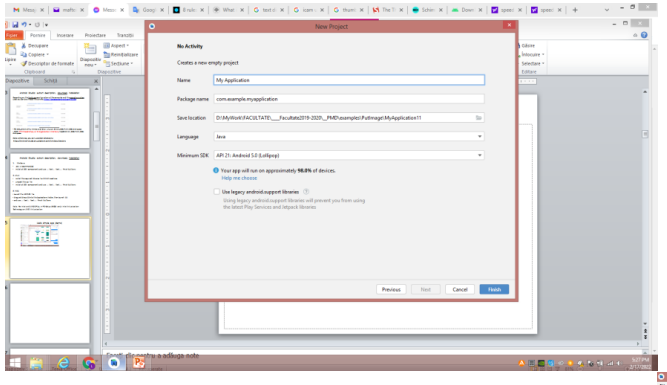
In some cases an Activity may return a value/object to the previous activity.

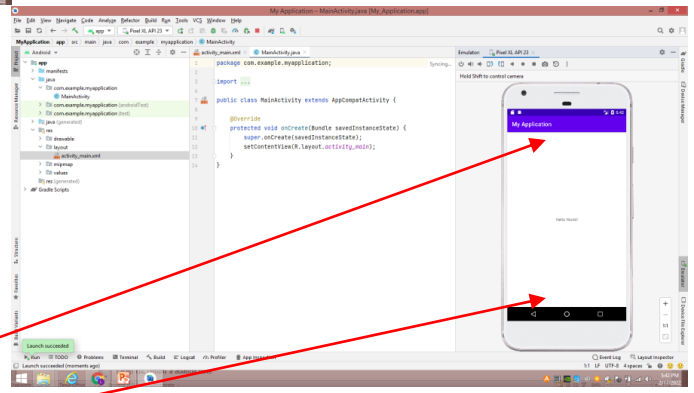
Example: an activity that lets the user pick a photo in called activity would return the chosen photo to the caller.

First example: Hello simple app (demo)

Android Studio -> New project -> Empty activity OR Empty Project (depends on AS version)







Student’s task: add/display two new texts on screen

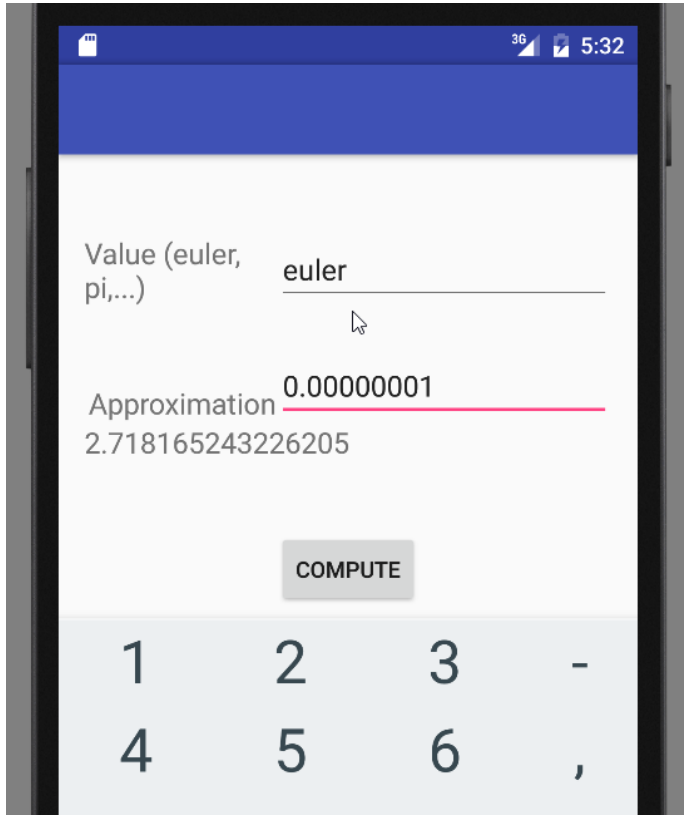
Second example: Computations App. -project *goal*-

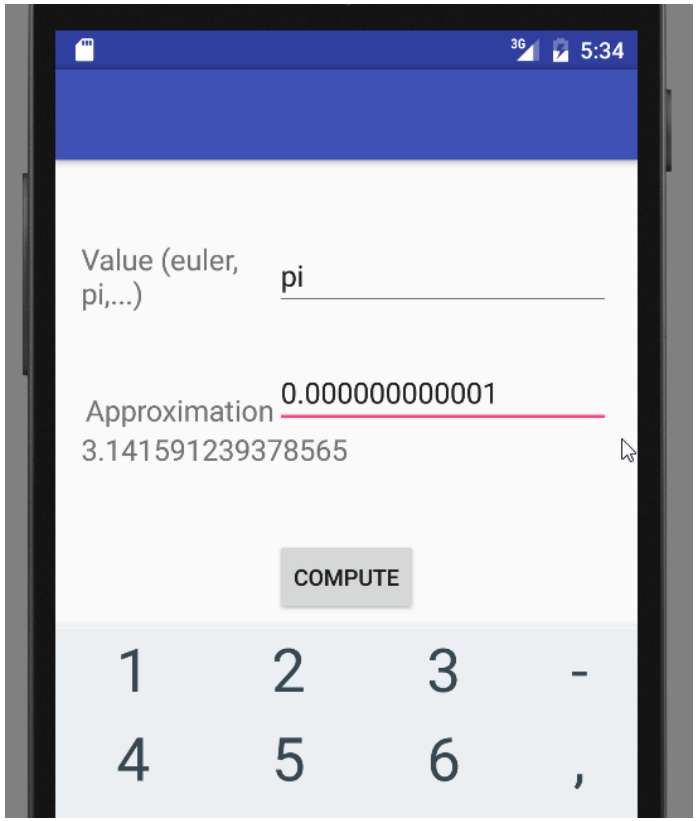
Utility: compute PI and Euler values

Users: mathematicians and of course CS students

IDE: Android Studio

Users' view: see the figures below





Computations App. - math theory – Euler and PI computation

Euler/Napier constant (given by Napier in honour of Euler)

$$e = \lim_{n\to\infty}\left(1+\frac{1}{n}\right)^n \quad (1)$$

$$e = \sum_{k=0}^{\infty}\frac{1}{k!} = \frac{1}{0!}+\frac{1}{1!}+\frac{1}{2!}+\frac{1}{3!}+\cdots. \quad (2)$$

The convergence of this series is very slow so, many iterations are needed to obtain a good / desired approximation of it

n=1, e=2.0

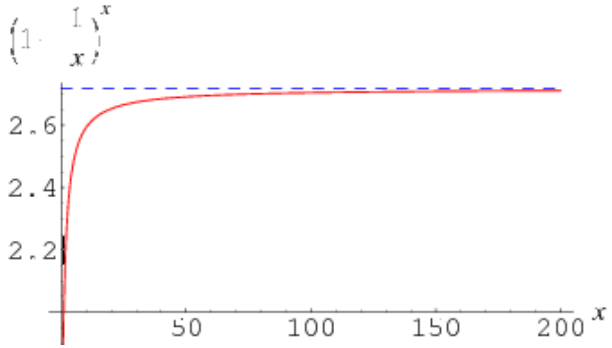
n=10, e=2.593

n=100, e=2.7048

n=1000, e=2.71692

n=10000, e=2.718145

……. =2.71828182845904523….



What a good approximation means? ***aprox < $e_i - e_{i-1}$***

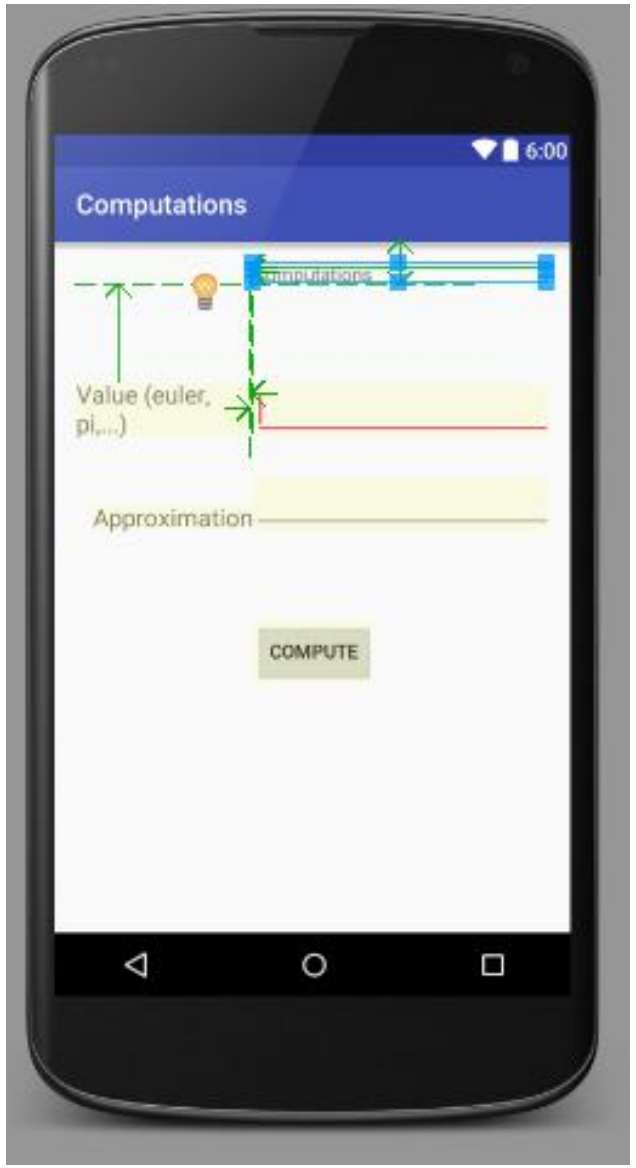
PI constant

$$\pi = 4\sum_{k=0}^{\infty}\frac{(-1)^k}{2k+1} = \frac{4}{1}-\frac{4}{3}+\frac{4}{5}-\frac{4}{7}+\frac{4}{9}-\frac{4}{11}\cdots. \quad (3)$$

What a good approximation means? ***approx = f(number of terms)***

Computations App example

Layout -> ***content_main.xml*** file Design form



Drag and drop using Design mode
NO Editor mode: it's only for details

?
What type of control should each of these be?
?

Computations App

Layout -> ***content_main.xml*** file Edit form

```
<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"
   android:paddingBottom="@dimen/activity_vertical_margin"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin">

   <TextView
      android:id="@+id/textView1"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="Computations"
      android:layout_alignParentTop="true"
      android:layout_toRightOf="@+id/textView2"
      android:layout_toEndOf="@+id/textView2" />
```

RelativeLayout
is a view group that displays child views in relative positions (vs. LinearLayout, ConstraintLayout…)

paddingBottom in *dimens.xml* file
<dimen name="activity_horizontal_margin">16dp</dimen>

1) The constant ***fill_parent*** was replaced with ***match_parent*** in Android 2.2 .

Or, use the value ***–1***.

fill_parent ⇔the view wants to be as big as its parent (minus padding)

2) wrap_content ⇔ some similar with "Autosize" from Windows Form Control.

toRightOf is a property of RelativeLayout: no effect in LinearLayout.
Positions the left edge of this view to the right of the given anchor view ID.

Computations App

Layout -> ***content_main.xml*** file Edit form

```
<Button
   android:id="@+id/btnCompute"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Compute"
   android:layout_below="@+id/txtResult"
   android:layout_centerHorizontal="true"
   android:layout_marginTop="43dp" />


<EditText
   android:id="@+id/txtApprox"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:ems="2"
   android:inputType="number|numberDecimal"
   android:layout_alignBottom="@+id/textView3"
   android:layout_alignLeft="@+id/txtValue"
   android:layout_alignStart="@+id/txtValue" />



……
```

In MainActivity.java file:
btnCompute = (Button)findViewById(R.id.*btnCompute*);

"Compute": the string on the button

In MainActivity.java file:
secondApprox = (EditText)findViewById(R.id.***txtApprox***);

ems is a typography term, that controls text size. The em is the font size.
In TextView there is an attribute named android:ems. The description is "Makes the TextView be exactly this many ems wide"

Computations App

Activity -> ***Main_Activity.java*** file

package name: launch the application using its package name.

A class for mapping from String values to various Parcelable types.

This class represents the basic building block for user interface components. A View occupies a rectangular area on the screen and is responsible for drawing and event handling. View is the base class for widgets, which are used to create interactive UI components (buttons, text fields, etc.).

```
package com.example.mafteiu_scai.computations;

import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.support.v7.app.AppCompatActivity;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
```

Interface definition for a callback to be invoked when a view is clicked

Base class for activities that use the support library action bar features.

Represents a push-button widget. Push-buttons can be pressed or clicked by the user to perform an action.

Displays text to the user and optionally allows them to edit it.

EditText is a thin/layer over TextView that configures itself to be editable

Computations App

Activity -> ***Main_Activity.java*** file

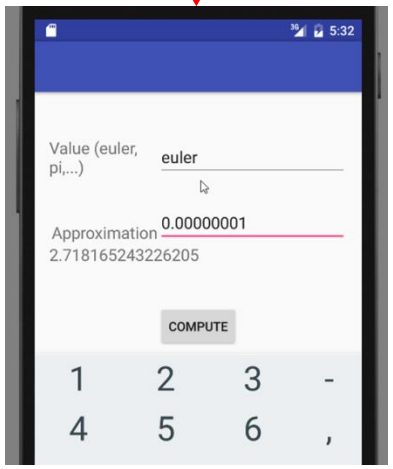
Base class for activities that use the support Library action bar features, derived from:

```
java.lang.Object
  ↳android.content.Context
    ↳android.content.ContextWrapper
      ↳android.view.ContextThemeWrapper
        ↳android.app.Activity
          ↳android.support.v4.app.FragmentActivity
            ↳  android.support.v7.app.AppCompatActivity
```

```
public class MainActivity extends AppCompatActivity {

  // Variable Declaration

  EditText firstValue;
  EditText secondApprox;
  TextView computeResult;
  Button btnCompute;

  double approx, sum;
  String value;

  double prev_term;
  double curr_term;
  int n, i;
  double aux;
```

Variables from controls
In connection with *content_main.xml* file

```
<EditText
   android:id="@+id/txtApprox"

<Button
   android:id="@+id/btnCompute“

<EditText
   android:id="@+id/txtApprox"
```



corresponding variables of … used in java processing

Other variables used in computation

Computations App

Activity -> ***Main_Activity.java*** file

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    firstValue = (EditText)findViewById(R.id.txtValue);
    secondApprox = (EditText)findViewById(R.id.txtApprox);
    computeResult = (TextView)findViewById(R.id.txtResult);
    btnCompute = (Button)findViewById(R.id.btnCompute);

btnCompute.setOnClickListener(new OnClickListener() {
public void onClick(View v) {
    value = firstValue.getText().toString();
    approx = double.parseDouble(secondApprox.getText().toString());
```

onCreate: a method that initialize the activity.

setContentView(int) with a layout resource define the UI

findViewById(int) are used to retrieve the widgets in UI, needed to interact with java program

Create click listener object

onClick, the method that will be invoked when the button is clicked.

converting String to Double in Android

getText() a method that can be used to get the copied text from the clipboard

Computations App

Activity -> ***Main_Activity.java*** file - ***euler*** **computing part**

Compare two string
Compares the given object to a string and returns true if they are equal.

```
if(value.equals("euler")) {     //refer e from ln
    double prev_term = 2;
    double curr_term = 1.5*1.5;
    n=2;
    while (curr_term - prev_term > approx) {
        prev_term = curr_term;
        n++;
        aux = 1 + 1.0/n;
        curr_term = aux;
        for(int j=2;j<=n;j++)
            curr_term *=aux;
    }
    sum = curr_term;
    computeResult.setText(Double.toString(sum));
}
```

Algorithm implementation for computing ***e*** constant (base of ln)

$$e = \lim_{n\to\infty}\left(1+\frac{1}{n}\right)^n$$

approximation equal with **approx**

Replace text in **computeResult** (in TextView) with converted value of **sum**.

Computations App

Activity -> ***Main_Activity.java*** file ***PI*** **computing part**

```
else
    if(value.equals("pi")) {    //pi
        double prev_term = 1;
        double curr_term = -1.0/3;
        sum = prev_term + curr_term;
        i=2;
        int sign=-1;
        while (Math.abs(prev_term) - Math.abs(curr_term)  > approx)
        {
            prev_term = curr_term;
            i++;
            sign *= (-1);
            curr_term = sign * 1.0 / (2*i-1);
            sum += curr_term;
        }
        sum = 4.0 * sum;
        computeResult.setText(Double.toString(sum));
    }
    else
        computeResult.setText("Incorrect string for Value");
}
```

PI computing algorithm

$$\pi = 4\sum_{k=0}^{\infty}\frac{(-1)^k}{2k+1} = \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11}\cdots.$$

Student’s task:

Modify the app to compute three constant value:

-the new one is your choice

Example 3

Working with .txt files

Working with .txt files – basic

Open the file:

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.

Input/read from file:

FileInputStream: an input stream for read file.

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

Output/write:

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.

Working with .txt files

App's goal/ utility: solve linear systems of equations

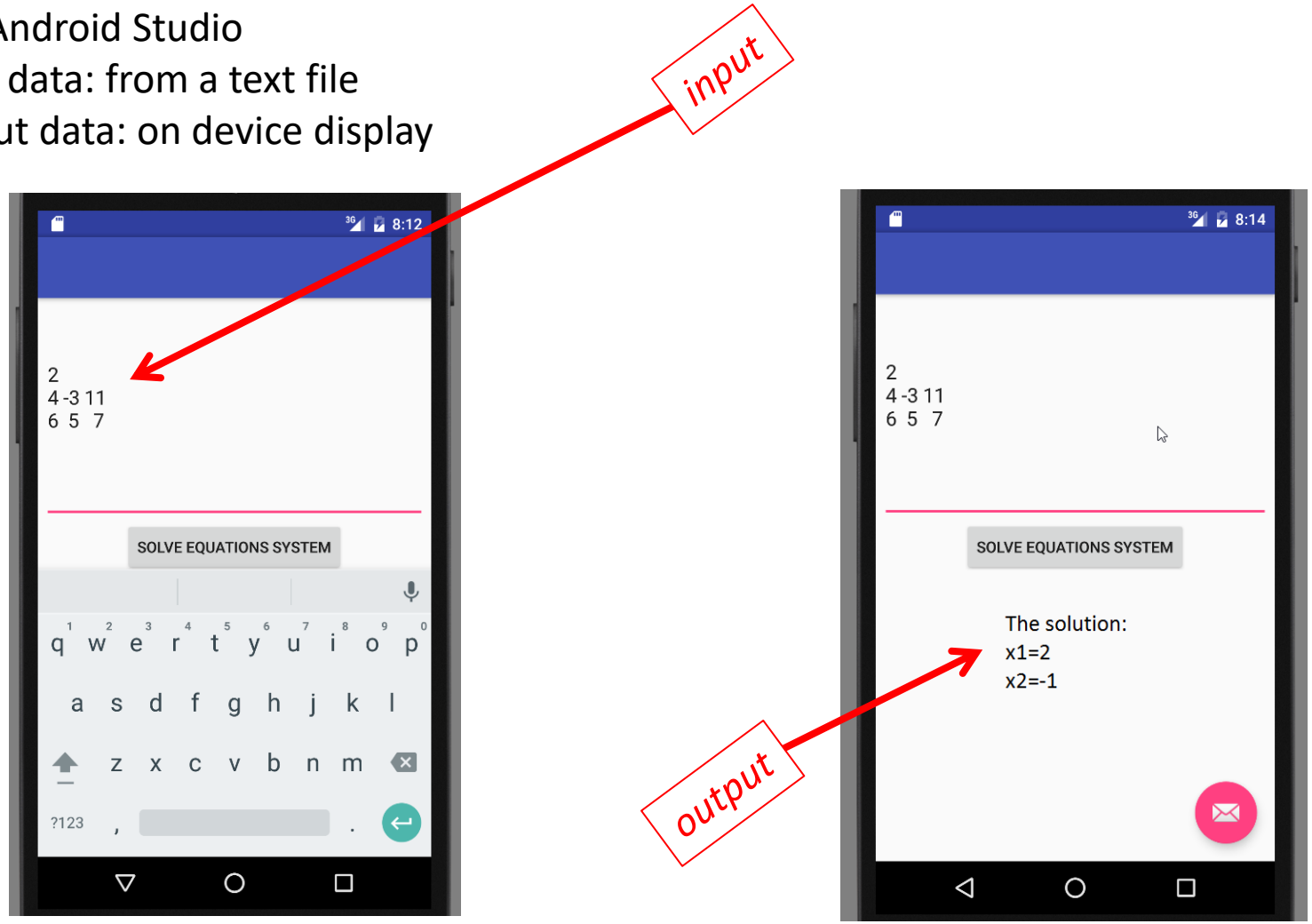
Q: why files are needed in this kind of apps?

Users: engineers, …..

IDE: Android Studio

Input data: from a text file

Output data: on device display



Solving systems of equations using:

1. Cramer's rule (2x2 and 3x3 system)
2. Gaussian elimination (nxn system)

Note: many many many… methods exist for solving linear systems of equation….

Cramer's rule – math theory

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

$$a_1x + b_1y + c_1z = d_1$$
$$a_2x + b_2y + c_2z = d_2$$
$$a_3x + b_3y + c_3z = d_3$$

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:

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

Gaussian elimination- math theory

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

TOTAL= $2n^3$ / 3 operations.

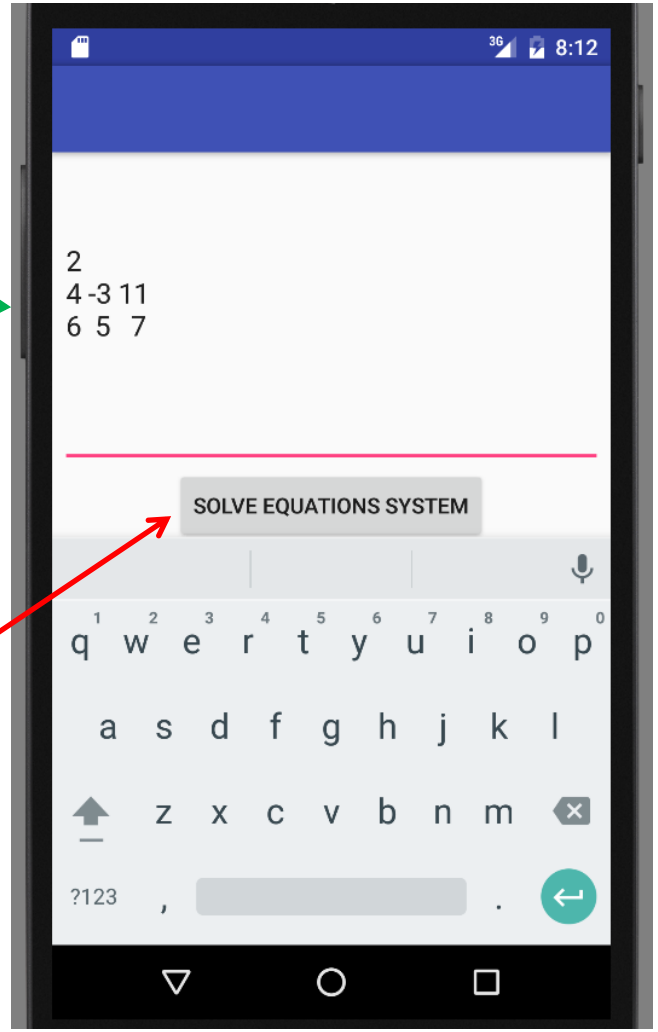
The method's problem: truncation errors => pivoting technique (first exchanges rows to move the entry with the largest absolute value to the pivot position)

Gaussian elimination-the basic 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]);
      }

   }

}
```

Working with .txt files – source code content_main.xml

```
<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>
```



Working with .txt 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);
    }
```

Working with .txt files MainActivity.java

```
// 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+' ');
            }
         }
```

Working with files MainActivity.java

```
            //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();
        }
    }
}
```