phd. lecturer Liviu Octavian Mafteiu-Scai

- User Interface and User eXperience

- Multiple Activities and Intents

User Interface
and
User eXperience

Phd. Lecturer ***Liviu Octavian Mafteiu-Scai***

The goal of this part

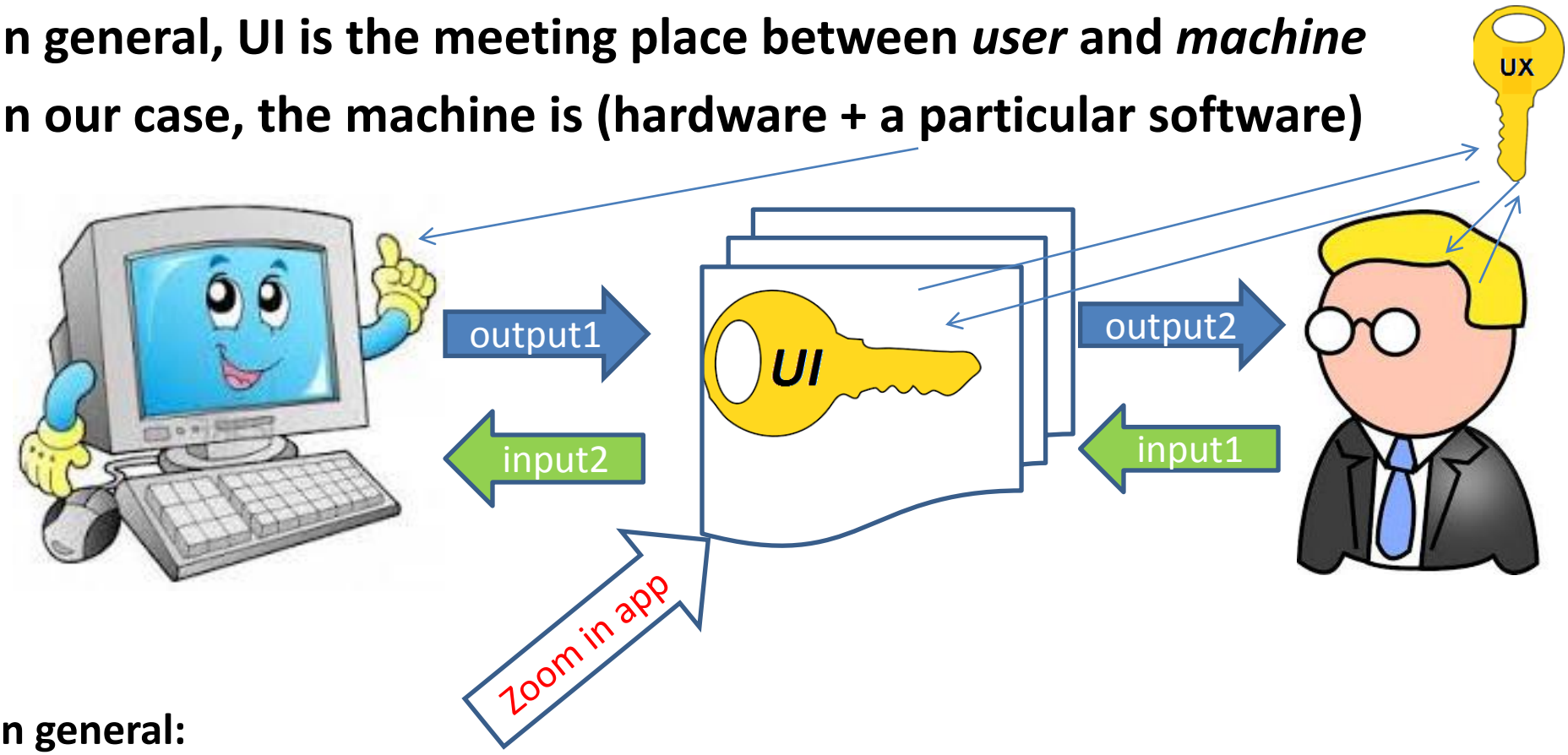
The terms UI and UX are very important for software developers, especially for those involved in "front-end" department of a company.

Who don’t understand the importance of these and don’t know basic rules of them, can not performs good software apps

What is User Interface (UI)?

In general, UI is the meeting place between *user* and *machine*

In our case, the machine is (hardware + a particular software)



In general:

Input1 and Output2 = mechanical, audio, optical or electrical signals

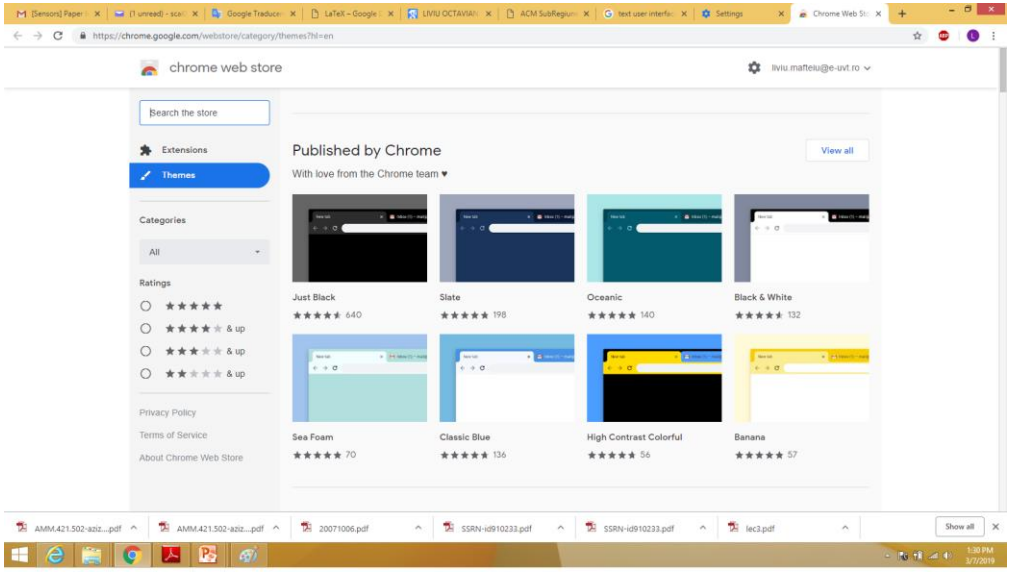
Input2 and Output1 = chars, numerical values (single, arrays, matrices etc, derived from math representation of the problem) (binary values)

User Interface (UI)?

In past *the user had to somehow adapt to the system*
(no UI or very poor)

```
MAIN MENU

        G - Go Start Communications
        H - Hangup Phone
        I - Enter Site ID
        L - Load Parameters from Disk File
        S - Save Parameters on Disk File

                Sub-Menus:

        P - Parameter               O - Option
        U - Text File Upload        D - Text File Download
        B - Binary File Xfer        M - Macro Definition
        C - Command Processor       F - Character Filter
        T - Special Features

        X - Exit to Operating System

Enter option (? for help):
```

At present *the system/app must adapt to the user.*



User Interface (UI)?

A good communication between user and machine is ensured only if we have a good UI

Important questions for programmers (front-end area)

- Is UI a software component designed for the users?
- What users want? In general they want apps that meet their needs.

- **An app must be easy to use**. If an app is difficult to use, the users stop to use it and they will seek a similar app, even your software is technically superior.

- It is know that a good UI allows user who understand the problem domain to **work with an app without having to read the app's manual or receive training**. So, a first advantage of a good UI is that it reduce the user's training costs.

- To design **a good UI** must have **a good understanding of the user's**, his mental and physical model, his physical and psychological abilities.

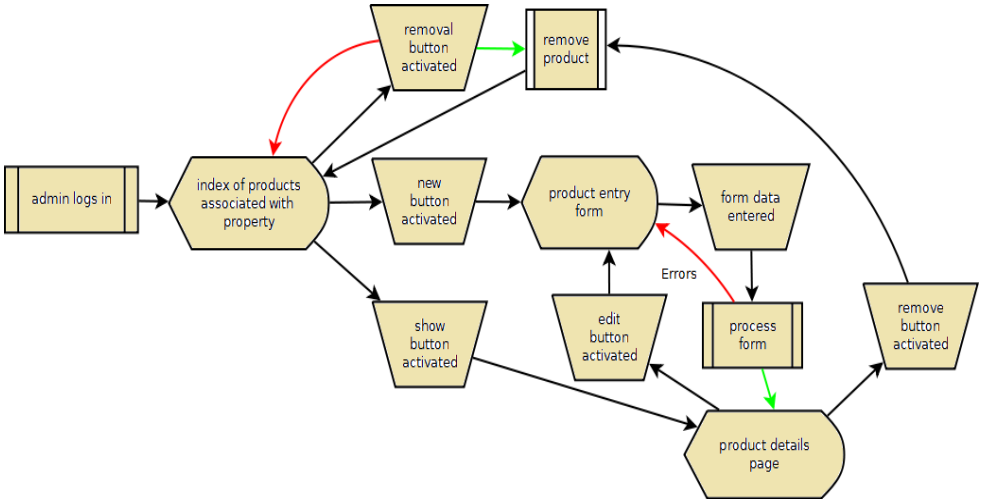
User Interface – Interface-flow diagrams

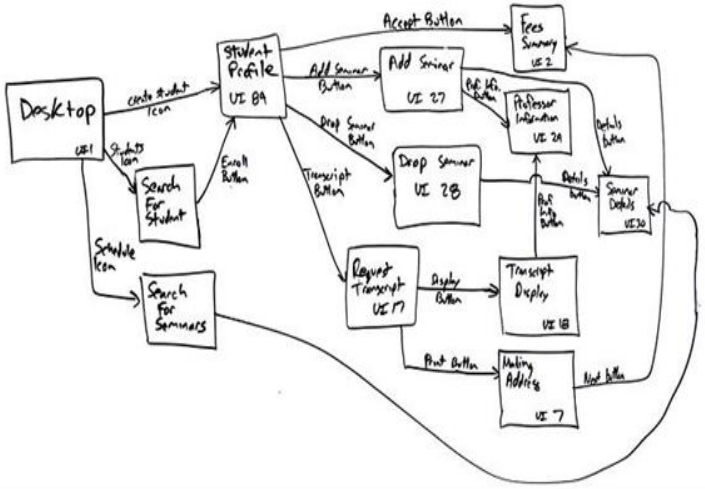
Before to making an Ui for an app, you must to do a project of it,

i.e. paint an ***interface-flow diagram*** (see *Software engineering* course)

Interface-flow diagrams:

- show the relationships between the user interface components (screens/inputs and reports/outputs, buttons, input boxes, messages ……controls…)
- allow to gain a high-level overview of the interface for apps
- help in understanding of how the system is expected to work
- will alow the validation/testing of the UI design.





Legend: *boxes-> user interface objects (screens, reports, or forms)*

arrows-> possible flows between screens.

User Interface - Rules

R1 Consistency. This will create an accurate mental model for users.

- the same action in different places must have the same effect (a click on mouse on *Yes* button)
- a word, a message or a warning must have the same meaning.
- a consistent color scheme must be used all over

R2 Standards. It is strong recommended to respect standards.

-IBM (*Common User Access* (CUA) 1987 or *Advanced CUA* (for graphical interfaces) in 1990)

-Microsoft (*The Windows Interface Guideline for Software Design*, 1995) set useful standards.

R3 Difficulty level. A good UI must be easy to use both by novices and experts.

R4 Working rules of apps. The rules of how an app works should be simple and few in number.

R5 Text: must be consistently and positively

? Examples for *positively* and *not positively* messages for users

User Interface - Rules

R6 Clustering : think in terms of clusters of app's interface objects.

? Example

R7 UI objects: should look and behave exactly like the real-world objects that they represent. Look at these and identify how users work with them.

R8 Contrast rule: *dark text on light backgrounds* and/or *light text on dark backgrounds*.
A good combination seems to be *yellow-blue.*

R8 Reduce users' memory load because the capabilities and the human memory are limited. (few rules)

R9 Aligns: *right justify numbers*, *left justify strings*, *left justify edit fields, right justify labels*.

R10 Density: don't create busy/crowded screens.

R11 Provide Defaults, Undo, and Redo

R12 Provide Interface Shortcuts (F for **File**, **E** for **Edit**, **V** for **View**, and **H** for **Help**.**)**
(not useful for mobile apps)

User Interface - Rules

R12 Menus: The menus, more exactly the pop-up menus, shouldn't be the only source of functionality. But, don't put menu in a dialogue box. This is too complex/complicated for users.

R13 Main objects: in each UI (especially in GUI) there are some main objects. So, the next problems must be solved:

- Identify the main objects.
- Check if these main objects are appropriate for the target users?
- Place main objects in the centre.
- Ideal: only one main object on a screen
- Give largest space to the main objects (this is very important in mobile apps)

R14 Use screen space effectively, avoid large empty spaces in a screen/window.

R15 Check boxes or radio buttons? Use check boxes for non-exclusive options, and radio buttons for exclusive options.

? Examples

R16 Radio buttons or drop-down list? Use radio buttons for a short list, e.g., < 10 and use drop-down list for a long list.

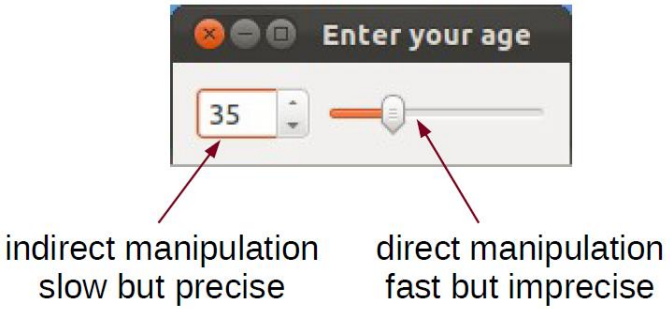
User Interface - Rules

R17 Remember,

This is a golden rule in software development: ***"less is more…"***

"Make it simple, but no simpler." (Albert Einstein)

R18 Manipulation: Direct manipulation is intuitive and easy to use but difficult to repeat precisely (examples: positioning, resize, alignment). Indirect manipulation is less intuitive but easy to repeat precisely.



? Examples for good and bad user's input for each control in part

R19 Grouping: Group related items together and separate unrelated items.

Place "unsafe" elements away from "safe" elements. The "safe" element must be default.

Use group boxes and whitespace to group logically related items on the screen.

User Interface - Rules

R20 Colours: Colour is used in UI (like in real life) to communicate:

- Meaning:

- gray: neutral
- blue: start, commit
- red: error, stop, vulnerable, critical, restricted , problems
- yellow: warning, caution, questionable
- green: go, proceed, progress, safe

- State: selected vs. not selected
- *Differentiation*: use the same colour for related items.
- *Emphasis*: use distinct and power colours to draw attention.

R21 Colour contrast: use contrasting colours for clarity.

R22 Colour matching: don't put together colours that look bad. Looks unprofessional

? Question only for girls: examples for combinations of unsuitable colors

User Interface - Rules

R23 Texts and Messages: Texts provide descriptive info and feedback to users.

Texts and messages have two attributes:

- *visual appearance*: font, size, colour
- *semantic*: use of terms, sentence construction

Provide informative feedback

Don't be potentially destructive.

Use familiar terms

Don't use programming jargon. Avoid messages like this *"The password is too short. It must be at least 1024 bytes long!* OR *Type the string of chars again! OR End with the EOL character!"*

Use terms appropriate to the context.

Write in user's vocabulary, provide suggestions.

Use fonts that are easy to read and avoid shadow and other font effects in main text.

R24 Let users *think* they are in control. Allow users to customize the interface (Themes that can be modified by user)

R25 Even if the interface on mobile apps is designed to be optimized for TOUCH/TAP and DRAG, sometimes, a keyboard (even virtual) could be useful and more safe.

R26 Allow Users to Change Focus

UI should be designed so users are able to interrupt their current actions or tasks (by a phone call…) and either continue later.

User eXperience

What is User eXperience (UX)?

Another golden rule for UI design, guided by UX

It matters what the user wants, no matter what programmers want

To have a successful project you need a shared vision of the project's future. It is known that all the details of a project are very important, but getting too caught up in the details can lead to an inconsistent UI over time.

Unfortunately, many projects start without a clear sense of the final objectives. The main goal of a project must be known before its start, but often the user goals are not articulated, that leads to frustrate or dissatisfied users.

In general, **UX** encompasses all aspects of the end-user's interaction with a company, its services, and its products.

A good understanding of UX is often the main key to a product's success.

What is User eXperience (UX)?

Every painting or movie you see, product you use, and even place you go, also has UX. You could have a bad experience in traffic, a good one on a eMAG shop, a bad experience on an exam, a good one in a restaurant….

Bad experiences must be erased from user mind. Good ones must be amplified

In a good UX design it is necessary to build something that meets clients' needs and goals.

Psychology is used to find UX' element. To create a good UX, you must ***analyze user behavior.***

What is User eXperience (UX)?

How to find what users want?

There are two methods:

- **Analytics**

  examples: 325,090 clicks, 45 views per month, 12%

  How: specialized companies can be used for survey

- **Relationships**

  examples: studied CS, has 2 kids, hasn't to much free time

  How: get face to face time with your users (forums, mail lists, direct presentation meetings …)

What is User eXperience – UI is not UX

If someone knows what to do exactly with the UX, he will be able to create extraordinary apps, because the usability of an app can be improved using a UX strategy.

!!!

<u>UI is not UX</u>:

- UI refers to the actual system they interact with users
- UX is about the users' emotions and induced behavior by emotions.

? Examples?

User experience differs from 'experiences in a general sense', in that it explicitly refers to the experience(s) derived from encountering systems.

What is User eXperience ?

UX as a phenomenon can be described as follows:

- UX is a subset of experience as a general concept. UX is more specific, since it is related to the experiences of using a given system

- UX includes encounters with systems – not only active, personal use, but also being confronted with a system in a more passive way, for example, observing someone else using a system
- Sometimes, UX of others could be included in our UX

?Example?

- UX is unique to an individual

- UX is influenced by prior experiences and expectations based on those experiences

- UX is rooted in a social and cultural context

What is User eXperience ?

What is not UX?

- UX is not technology driven, it focuses on humans behavior and experiences
- UX is not about just an individual using a system in isolation
- UX is not the same as usability, UX design is more than UI design

What is User eXperience - TIME SPANS

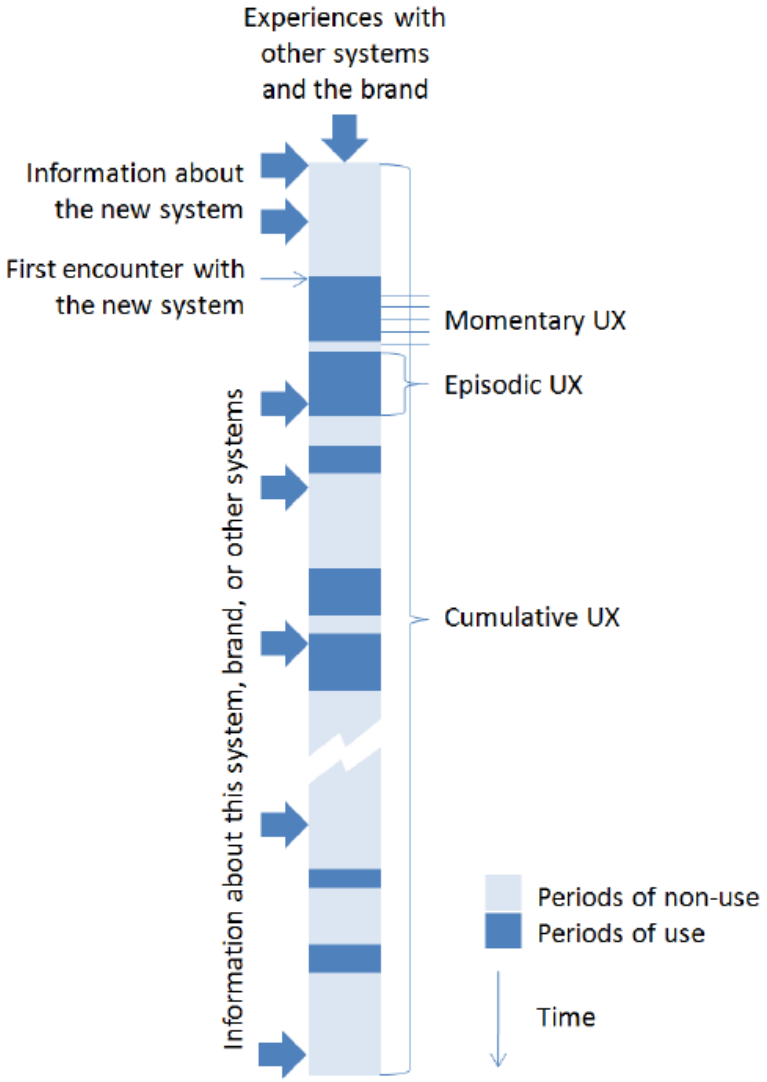


Figure 1. UX over time with periods of use and non-use

[Dagstuhl Seminar on Demarcating User Experience, September 15-18, 2010]

What is User eXperience - TIME SPANS

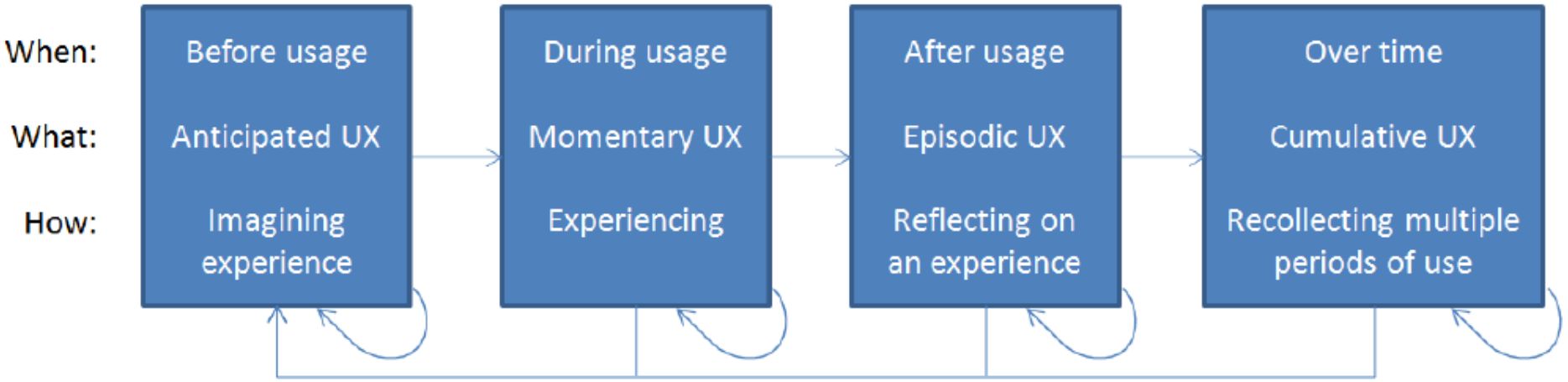


Figure 2. Time spans of user experience, the terms to describe the kind of user experience related to the spans, and the internal process taking place in the different time spans.

[Dagstuhl Seminar on Demarcating User Experience, September 15-18, 2010]

What is User eXperience - rules

You must keep in mind what's interesting to you as a user, not what's fun to do as a designer. These can be very different.

i.e.

If you were the user in a given situation, how would you feel?

Psychology is the most effective UX element. To create a good UX, you must ***analyze user behavior.***

UX is not about the icons or colors you choose. To obtain a successful result, you must develop a content strategy.

Positioning the user as a central concern in the design process

Identifying the aspects of the design that are important to the target user group

A lot of books are there about UX. One of these, that must be taken into consideration "*User experience guidelines for Universal Windows Platform (UWP) apps*" , Microsoft, August 2015

Practical advices

"***When people make a repeated error using my code, instead of asking why these people are idiots, I learned to ask what's wrong with my software that causes the nice people to look like idiots***" Cem Kaner, Professor of Software Engineering, Florida Institute of Technology

Multiple Activities and Intents

Most apps need more than one activity.

An activity is a single focused thing.

A window(layout) is created for user interaction

- Remember: activities are declared in Manifest file:

```
<manifest ... >
  <application ... >
     <activity android:name="MyActivity" />
     ...
  </application ... >
  ...
</manifest >
```

android:name ,specifies the class name of the activity.

NOTE: All mobile apps for Android OS have an (only one) *AndroidManifest.xml* file (with precisely that name). There are described a lot of important information about app.

A task represent multiple activities linked together to do something more complex.

In general, applications with

multiple activities,

need

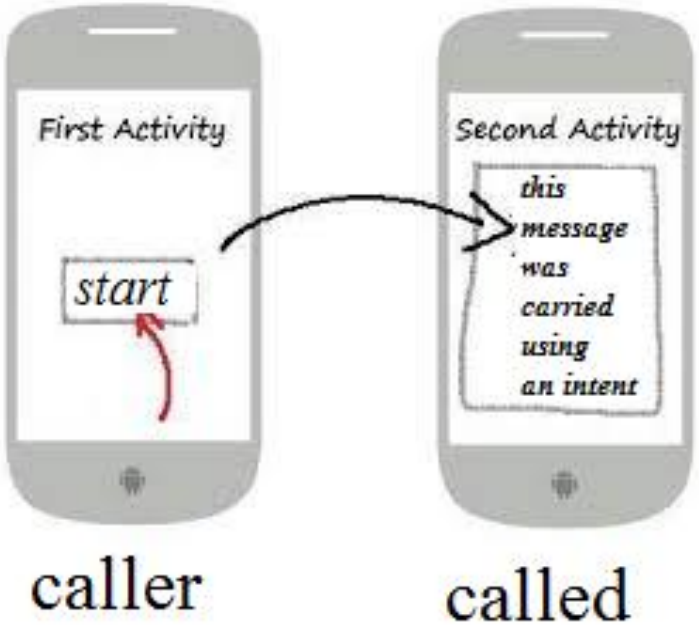
intents

to communicate to each other.

What an intent is?

An **Intent** is a messaging object used to request an action from another app component.

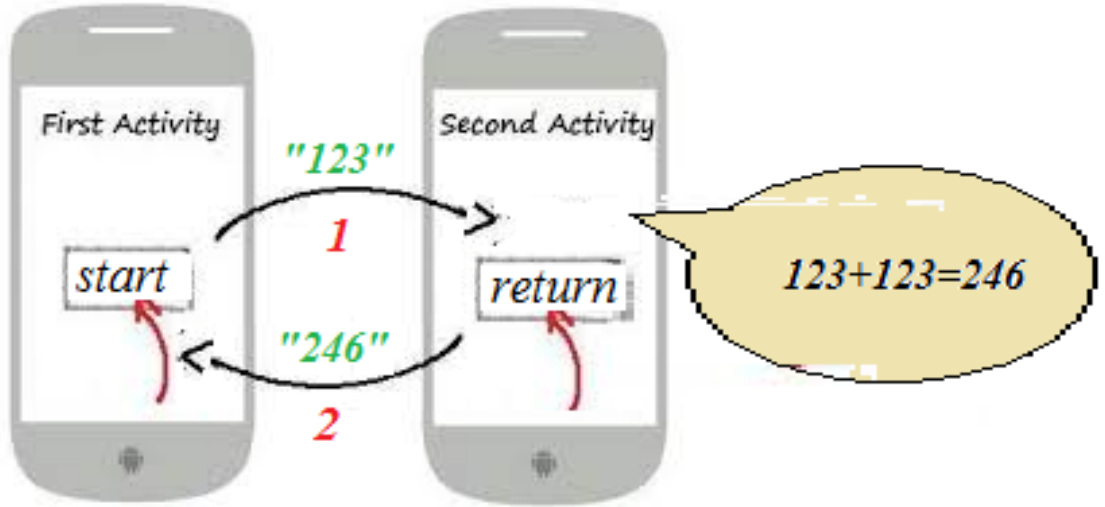
An **Intent** describes the activity to start and carries any necessary data



What an intent is? – *example -*

An **Intent** is a messaging object used to request an action from another app component.

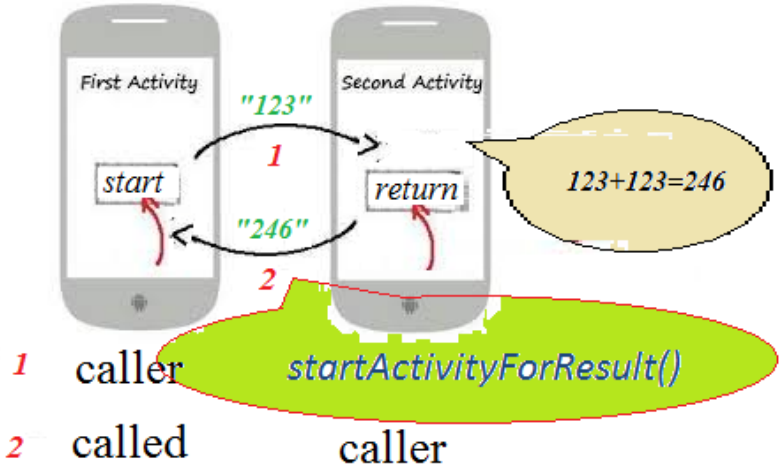
An **Intent** describes the activity to start and carries any necessary data



Intents are used to:

- **Start an activity**

! If results from activity called are needed in caller activity, the method *startActivityForResult()* must be uses.



- **Starting a service**

A *service* is a component that performs operations in the background without a user interface (music, AV programs…)

- **Delivering a broadcast**

A *broadcast* is a message that any app can receive (a low battery message)

Types of intents:

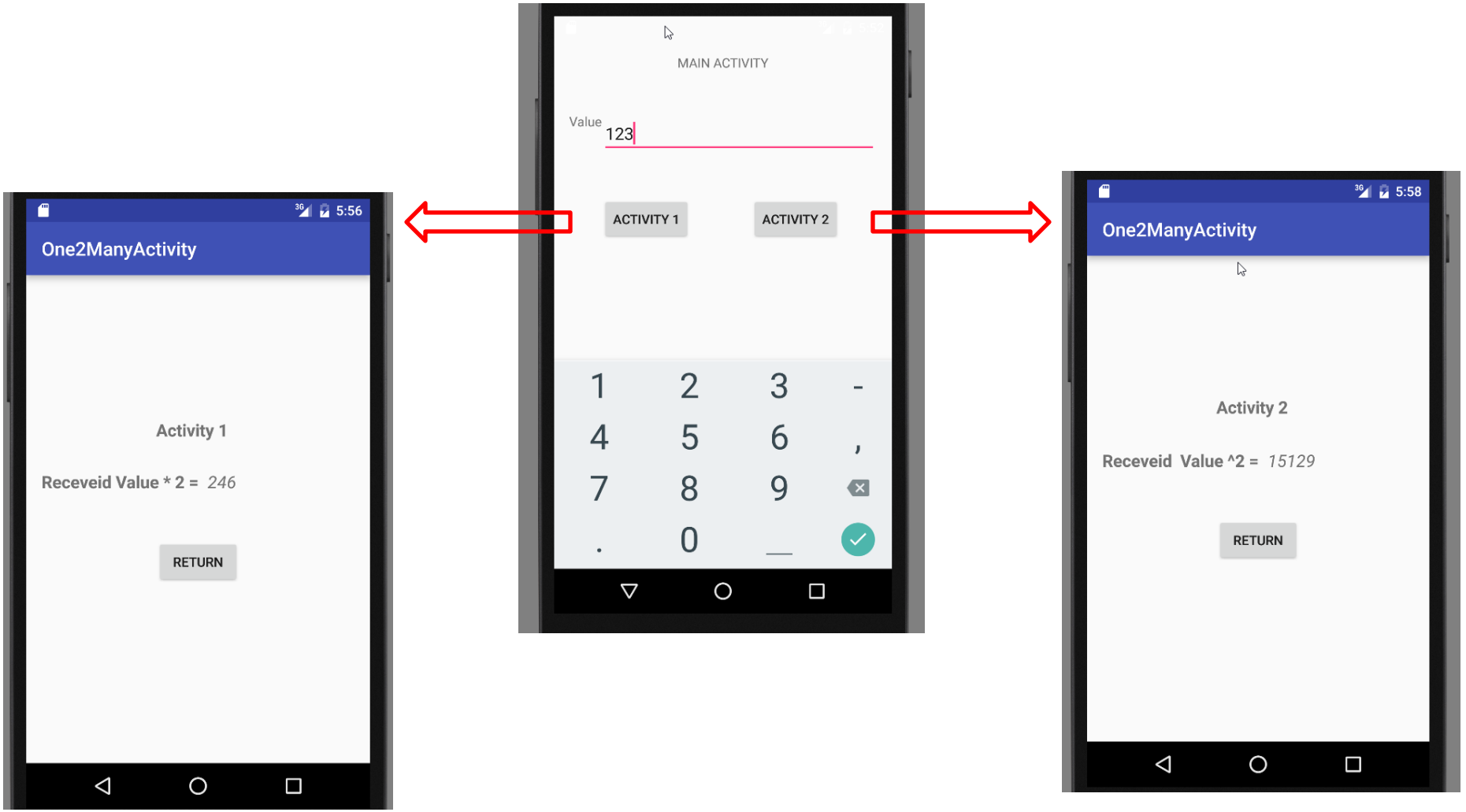
- **Explicit intents**: specify which application will satisfy the intent.

Used to start a component of app, when the class name of the activity or service are known (start an activity after a button is clicked or start a download file service in the background).

- **Implicit intents:** used to declare a general action to perform (if a geographical location is necessary in your app, the implicit intent is used to call another app that capable to find and show a location on a map)

One2TwoActivities

Goal: launching two activities from other activity (called *main activity*) and passing data from main activity to secondary activities:



One2ManyActivity App - The Main Steps

1. Create a standard Android Studio project. Name it *One2ManyActivity*, with a Blank Activity.

2. Add Button, EditText and TextView into *activity_main.xml.*

3. In src > main > java > add new fragments activity classes: *FirstActivity.java* and *SecondActivity.java.*

4. In src > main > res > layout > add new layout xml files for fragments: *activity_first.xml* and *activity_second.xml*. After, add Button and TextView for each of them.

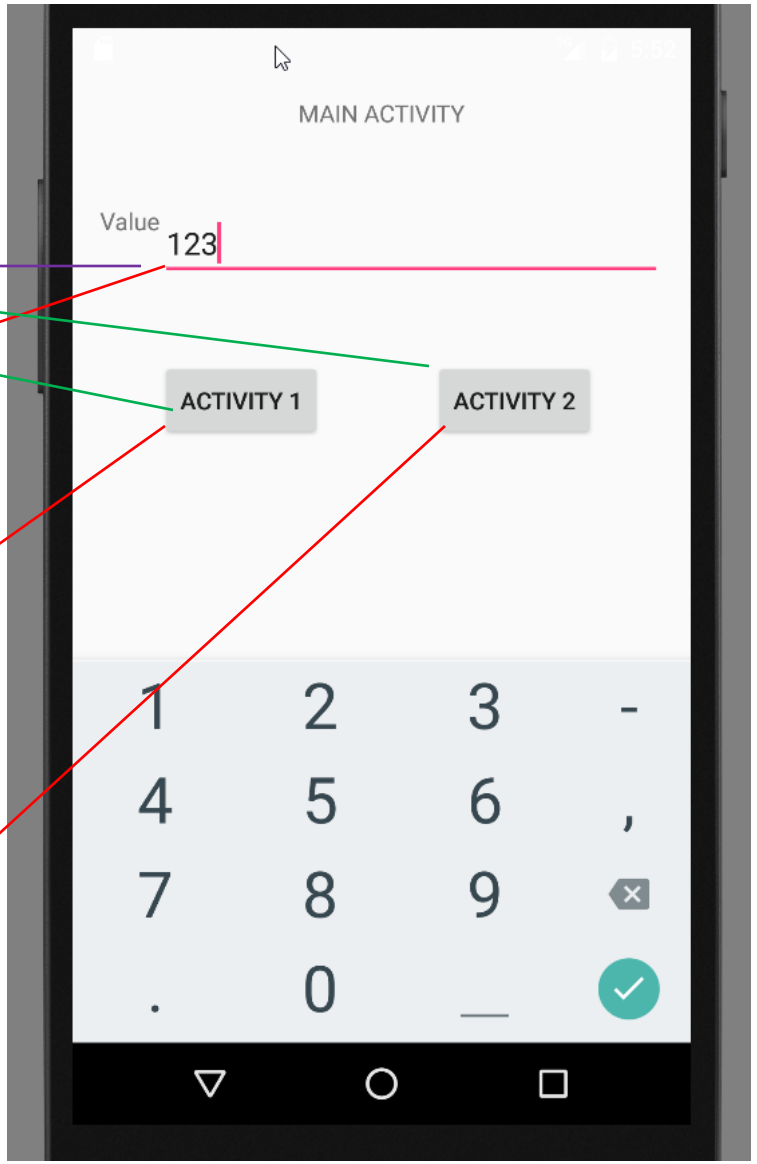
5. Rewrite *MainActivity.java* code in accordance with the requirements of the app.

6. Finaly, Run and Test your app.

The source code of this app is in next slides

One2ManyActivity App - *MainActivity.java*

```
package com.example.mafteiu_scai.one2manyactivity;
import android.content.Intent;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

public class MainActivity extends ActionBarActivity {
   Button button1, button2;
   EditText editTextInt;
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      button1 = (Button) findViewById(R.id.btnCall1);
      button2 = (Button) findViewById(R.id.btnCall2);
      editTextInt = (EditText) findViewById(R.id.editText2);

      button1.setOnClickListener(new View.OnClickListener() {
         public void onClick(View view) {
          Integer i =Integer.parseInt(editTextInt.getText().toString());
           //the first activity name
           Intent intent = new Intent(getApplicationContext(),FirstActivity.class);
           // pass value to Activity 1.
           intent.putExtra("Int",i );
           startActivity(intent);
         }
      });

      button2.setOnClickListener(new View.OnClickListener() {
         public void onClick(View view) {
           Integer i =Integer.parseInt(editTextInt.getText().toString());
           //the second activity name
           Intent intent = new Intent(getApplicationContext(),SecondActivity.class);
           //pass value to Activity 2.
           intent.putExtra("Int",i );
           startActivity(intent);
         }
      });
   }
}
```



One2ManyActivity App - *MainActivity.java*

```
package com.example.mafteiu_scai.one2manyactivity;
import android.content.Intent;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

public class MainActivity extends ActionBarActivity {
   Button button1, button2;
   EditText editTextInt;
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      button1 = (Button) findViewById(R.id.btnCall1);
      button2 = (Button) findViewById(R.id.btnCall2);
      editTextInt = (EditText) findViewById(R.id.editText2);
      button1.setOnClickListener(new View.OnClickListener() {
         public void onClick(View view) {
          Integer i =Integer.parseInt(editTextInt.getText().toString());
           //the first activity name
           Intent intent = new Intent(getApplicationContext(),FirstActivity.class);
           intent.putExtra("Int",i );
           startActivity(intent);
         }
      });
      button2.setOnClickListener(new View.OnClickListener() {
         public void onClick(View view) {
           Integer i =Integer.parseInt(editTextInt.getText().toString());
           //the second activity name
           Intent intent = new Intent(getApplicationContext(),SecondActivity.class);
           //pass value to Activity 2.
           intent.putExtra("Int",i );
           startActivity(intent);
         }
      });
   }
}
```

An **Intent** is a messaging object used to request an action from another app component. More: http://developer.android.com/guide/components/intents-filters.html

getApplicationContext returns the context for the entire application (the process all the Activities are running inside of).

A *Context* is an entity that represents various environment data. It provides access to local files, databases, class loaders associated to the environment, services including system-level services, and more.

pass value *i* (from editText2) to Activity 1.

start a new activity after button1 was pressed

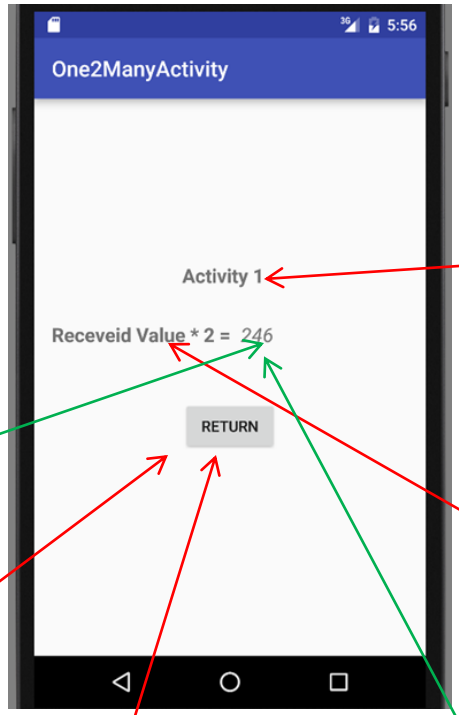
All activities are also specified in AndroidManifest.xml file:

```
<activity
   android:name=".MainActivity"
   android:label="@string/app_name"
   android:theme="@style/AppTheme.NoActionBar" >
   <intent-filter>
      <action android:name="android.intent.action.MAIN" />
     <category android:name="android.intent.category.LAUNCHER" />
   </intent-filter>
</activity>
<activity android:name=".FirstActivity" >
</activity>
<activity android:name=".SecondActivity" >
</activity>
```

One2ManyActivity App - *FirstActivity.java* (similar for SecondActivity.java)

```
package com.example.mafteiu_scai.one2manyactivity;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.content.Intent;
import android.view.View;
import android.widget.TextView;

public class FirstActivity extends AppCompatActivity {
   TextView txtInteger;

   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_first);
      txtInteger =  (TextView) findViewById(R.id.textView4);
      //read data from main ctivity
      Intent intent = getIntent();
      Integer i = intent.getIntExtra("Int", 0);
      i=2*i;
      txtInteger.setText(i.toString());

      findViewById(R.id.btnClose).setOnClickListener( new
View.OnClickListener() {

         public void onClick(View v) {
            //close Activity 1 and return to main activity
            finish();
         }
      });
   }
}
```

Finish the curent/secondary activity




```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Return"
    android:id="@+id/btnClose"
    android:layout_marginTop="48dp"
    android:layout_below="@+id/textView4"
    android:layout_centerHorizontal="true" />

</RelativeLayout>
```

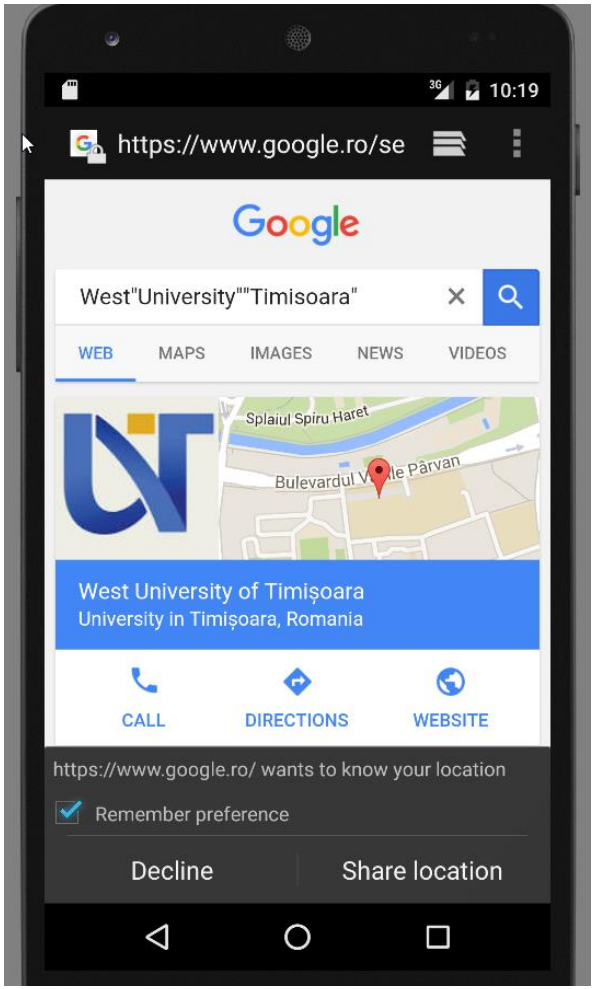
```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   android:paddingBottom="@dimen/activity_vertical_margin">

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textAppearance="?android:attr/textAppearanceMedium"
      android:text="Activity 1"
      android:id="@+id/textView"
      android:layout_marginTop="134dp"
      android:textStyle="bold"
      android:layout_alignParentTop="true"
      android:layout_alignStart="@+id/btnClose" />

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textAppearance="?android:attr/textAppearanceMedium"
      android:text="Receveid Value * 2 = "
      android:id="@+id/textView2"
      android:textStyle="bold"
      android:layout_below="@+id/textView"
      android:layout_alignParentStart="true"
      android:layout_marginTop="30dp" />

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textAppearance="?android:attr/textAppearanceMedium"
      android:id="@+id/textView4"
      android:layout_alignTop="@+id/textView2"
      android:textStyle="italic"
      android:layout_alignParentRight="true"
      android:layout_alignParentEnd="true"
      android:layout_toRightOf="@+id/textView2"
      android:paddingLeft="5dp" />
```

Example 2 A BrowserSearch App

- Goal: Launches a search process from a browser

Main activity

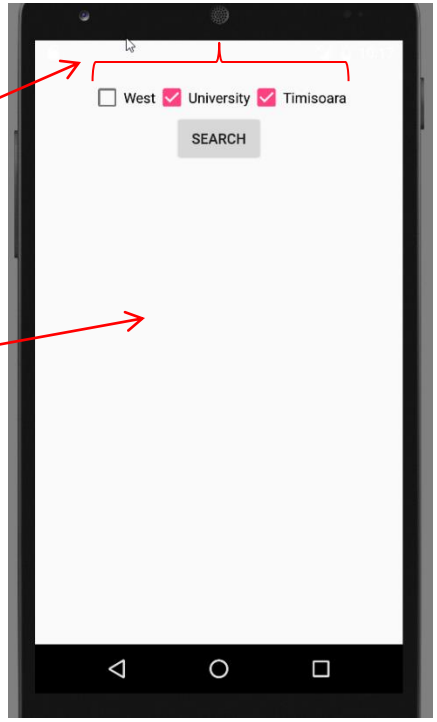
Second activity

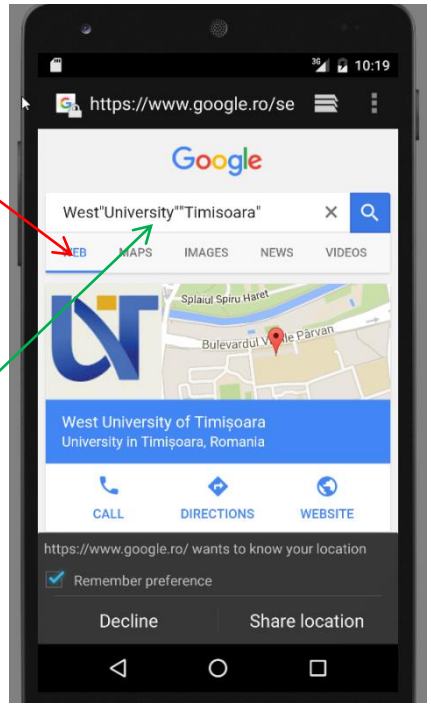


BrowserSearch App - MainActivity.java

```
package com.example.mafteiu_scai.browsersearch;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.webkit.WebView;
import android.widget.CheckBox;
public class MainActivity extends Activity {
   CheckBox placeBox, titleBox, cityBox;
   WebView webView;
protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      placeBox = (CheckBox) findViewById(R.id.checkBox);
      titleBox = (CheckBox) findViewById(R.id.checkBox2);
      cityBox = (CheckBox) findViewById(R.id.checkBox3);
      webView = (WebView) findViewById(R.id.webView);
   }
   public void onButtonClick(View view) {
      StringBuilder str = new StringBuilder("");
      if (placeBox.isChecked()) {
        str.append("West");
      }
      if (titleBox.isChecked()) {
         str.append("\"University\"");
      }
      if (cityBox.isChecked()) {
         str.append("\"Timisoara\"");
      }
      if (str.length() == 16) {          str.insert(4, '+');       }
      if (str.length() == 21) {          str.insert(10, '+');      }
      if (str.length() == 26) {  str.insert(4, '+');    str.insert(17, '+');  }
      if (str.length() == 0){ str.append("University of  Timisoara");
}
      webView.loadUrl
            ("http://www.google.com/search?q=" + str.toString());
   }
}
```

!





BrowserSearch App - activity_main.xml & reusable_layout.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"
   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">

   <include
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      layout="@layout/reusable_layout"
      android:layout_alignParentTop="true"
      android:layout_alignParentRight="true"
      android:layout_alignParentEnd="true"
      android:layout_alignParentLeft="true"
      android:layout_alignParentStart="true"
      android:id="@+id/include"/>

   <WebView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/webView"
      android:layout_below="@+id/include"
      android:layout_centerHorizontal="true"/>
</RelativeLayout>
```

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:orientation="vertical"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_alignParentTop="true"
   android:layout_centerHorizontal="true"
   android:gravity="center_horizontal"
   tools:showIn="@layout/activity_main">

   <LinearLayout
      android:orientation="horizontal"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_gravity="center_horizontal"
      android:gravity="center_horizontal">
      <CheckBox
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:text="West"
         android:id="@+id/checkBox"
         android:checked="false"/>
      <CheckBox
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:text="University"
         android:id="@+id/checkBox2"
         android:checked="false"/>
      <CheckBox
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:text="Timisoara"
         android:id="@+id/checkBox3"
         android:checked="false" />
   </LinearLayout>

   <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Search"
      android:id="@+id/button"
      android:layout_gravity="center_horizontal"
      android:onClick="onButtonClick"/>

</LinearLayout>
```