**Utilizarea Listelor în Android**

Lista este un concept des utilizat în aplicațiile Android, acestea permițând o structurare clară a opțiunilor/informațiilor oferite de o aplicație. De exemplu lista de mesaje, lista de contacte, lista melodiilor preferate etc., permit o afișare structurată a informațiilor. În plus, o opțiune din listă (idem) poate să aibă o interacțiune cu utilizatorul, astfel încât selectarea în sine să conducă la altă activitate a aplicației. De exemplu, dacă selectăm din lista de contacte o persoană, se deschide o altă activitate care prezintă detalii despre persoana respectivă, precum nume, număr de telefon, adresă de email etc.

Într-o aplicație Android, lista permite și o dezvoltare de tip **top to down** a activităților, astfel:

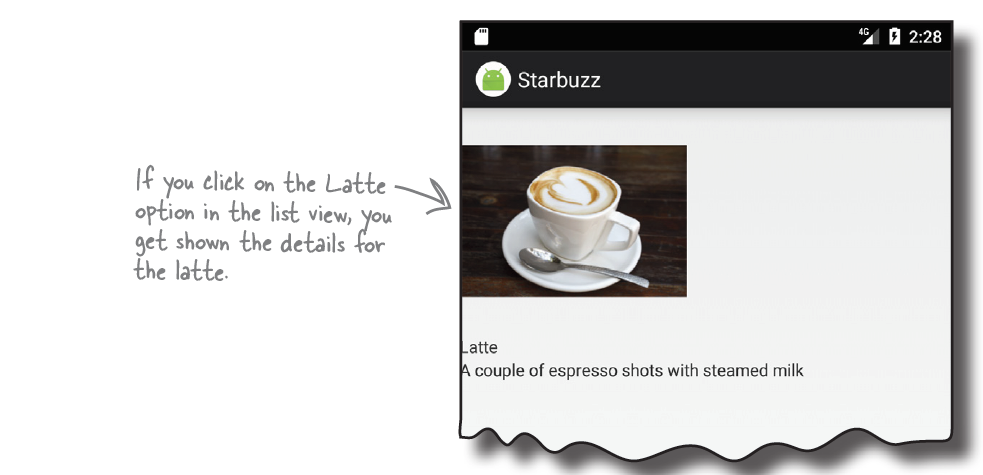
* nivelul superior (top) poate fi prima activitate (ecran) pe care utilizatorul o vizualizează la deschiderea aplicației;
* utilizatorul va naviga în lista de opțiuni având posibilitatea de a selecta o opțiune (un item din listă);
* odată selectată opțiunea, se poate deschide o altă activitate (partea to down) care permite afișarea unei alte activități/ecran.

Componenta grafică Android (view) pentru a defini o listă de opțiuni este **ListView.** View-ul list permite afișarea mai mult opțiuni așa cum se poate observa în exemplu de mai jos (fig. 1):



**Fig. 1 - Exemplu ListView**

În lista de mai sus, sunt listate băuturile comercializate de către Starbuzz. Utilizatorul poate selecta o opțiune, de exemplu Latte, deschizându-se astfel o altă activitate care oferă detalii despre item-ul selectat (Fig 2).

**Fig.2 – Selecție item din listă**

**Definirea unei liste ListView**

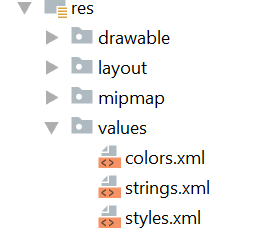
În documentul activity\_mai.xml, în modul Design se poate selecta din paleta de componente grafice view-ul ListView sau se editează documentul xml, astfel:

<**ListView  
 android:id="@+id/myListView"  
 android:layout\_width="409dp"  
 android:layout\_height="729dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"** />

Conținutul unei liste (opțiunile din listă) poate fi definit fie static, fie dinamic.

1. **Lista statică**

De regula, conținutul unei liste se definește printr-un tablou de string-uri. Acesta se definește in resursele proiectului, în fișierul res/values/strings.xml



**Fig 3 – Fișierul strings.xml**

Astfel, în fișeirul strings.xml, pentru a defini tablouri de string-uri se folosește atributul string – array

<**resources**>  
 <**string name="app\_name"**>Lista1</**string**>  
 <**string-array name = "options"**>  
 <**item**> Optiune1 </**item**>  
 <**item**> Optiune2 </**item**>  
 <**item**> Optiune2 </**item**>  
 </**string-array**>>  
</**resources**>

După ce s-a definit în resurse tabloul de string-uri, lista de proprietăți ale listei se completează cu atributul android:entries, așa cum se poate vedea mai jos:

<**ListView  
 android:id="@+id/myListView"  
 android:layout\_width="409dp"  
 android:layout\_height="729dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 android:entries="options"**/>

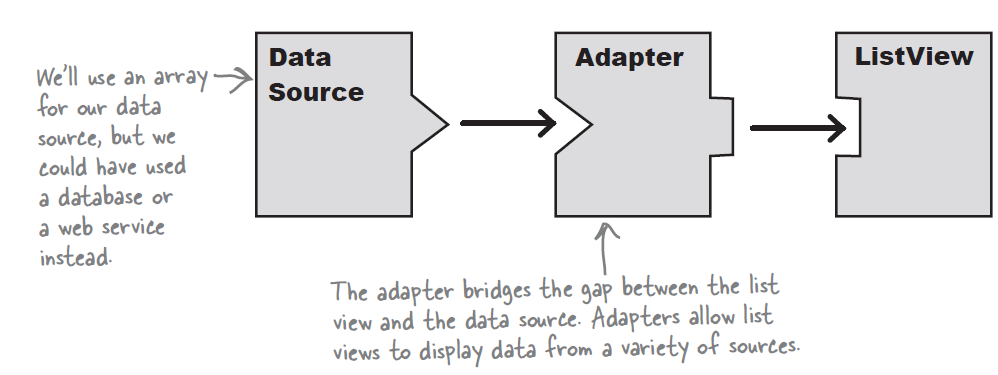
După ce se rulează aplicația se poate observa lista cu opțiunile specificate prin elementele tabloului de șiruri de caractere.

**Aplicație practică 1**: Construiți o aplicație Android care afișează la pornirea sa o listă de culori preferate.

1. **Lista dinamică**

De cele mai multe ori în aplicații se dorește ca lista să aibă un conținut dinamic, definit prin cod. Cu alte cuvinte, sursa unei liste nu mai definită în fișierul strings.xml, ci poate să definită prin cod Java, sau poate să fie încărcată dintr-un fișier/bază de date.

Pentru a defini o sursă dinamică a unei liste se folosește conceptul **adaptor.** Un adaptor realizează legătura dintre o sursă (un tablou în Java, o baza de date etc.) și componenta grafică ListView.



**Fig. 4 – Adaptorul unei liste**

În platforma Android sunt definiți mai mulți adaptori pentru o listă, mai departe voi exemplifica un adaptor pentru un tablou de string-uri în Java, denumit **array adaptor**.

În exemplul următor voi defini un adaptor de tip tablou pentru a defini o lista de culori.

În fișierul main\_activity.java, definim mai întâi un obiect al clasei ListView pentru a realiza legătura cu view-ul de tip list definit în layout\_activity.xml.

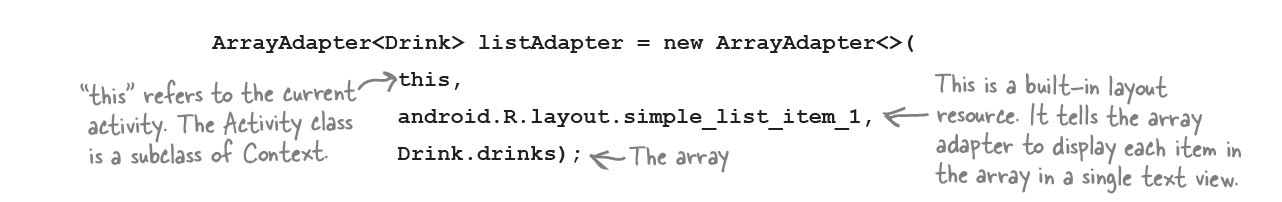
**public class** MainActivity **extends** AppCompatActivity {  
@Override  
**protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
  
**ListView myListView = (ListView) findViewById(R.id.*myListView*);**

}

Ulterior, se definește un tablou de string-uri ce conține opțiunile dorite

**final** ArrayList<String> colors = **new** ArrayList<String>();  
 colors.add(**"Rosu"**);  
 colors.add(**"Albastru"**);  
 colors.add(**"Alb"**);  
 colors.add(**"Verde"**);

Adaptorul de tip tablou se definește printr-o instanță a clasei ArrayAdaptor, astfel:



ArrayAdapter<String> arrayAdapter = **new** ArrayAdapter<String>(**this**, android.R.layout.***simple\_expandable\_list\_item\_1***, colors);

După ce adaptorul este definit, acesta se poate atașa unei liste ListView prin metoda ListView setAdaptor()

myListView.setAdapter(arrayAdapter);

**Aplicație practică 2**: Construiți o aplicație Android care afișează la pornirea sa o listă de culori preferate. Lista va fi definită dinamic.

**Evenimente și modalitatea de tratare a lor**

Evenimentele reprezintă o modalitate de a colecta informații despre interacțiunea utilizatorului asupra unei componente grafice (apăsarea unui buton, selecția unei opțiuni dintr-o listă, editarea unui câmp text etc).

Managementul evenimentelor în Android cuprinde trei concepte principale:

* **Event Listeners** (Ascultător de evenimente)– este o interfață implementată de către clase specifică unui view care conține o singură metodă callback. Metoda va fi apelată de către platforma Android în momentul în care se înregistrează o interacțiune a utilizatorului cu o componentă grafică.
* **Event Listeners Registration**(Înregistrarea unui ascultător de evenimente)– este procesul prin care un gestionar de evenimente se înregistrează la un ascultător de evenimente
* **Event Handlers** – la apariția unui eveniment, după ce s-a înregistrat un ascultător pentru evenimentul respectiv, ascultătorul de evenimente apelează un event handlers, adică metoda care tratează evenimentul

Exemple de evenimente și ascultători

|  |  |
| --- | --- |
| **Event Handler** | **Event Listener** |
| **onClick()** | **OnClickListener()**  Metoda este apelată atunci când utilizatorul efectuează un click sau o acțiune touche asupra unui view, precum un buton, un text, o imagine etc. |
| **onLongClick()** | **OnLongClickListener()**  Metoda este apelată atunci când un utilizator efectuează un click sau o acțiune touche într-un timp mai mare decât o secundă |
| **onFocusChange()** | **OnFocusChangeListener()**  Metoda este apelată atunci când utilizatorul este concentrat asupra unui item și apasă sau eliberează o cheie hardware pe dispozitiv |
| **onItemSelected()** | **OnItemSelectedListener()**  Metoda este apelată atunci când un utilizator efectuează un click sau o acțiune touche asupra unui item dintr-o listă |

Mai multe exemple de evenimente și ascultători sunt prezentate aici: <https://www.tutorialspoint.com/android/android_event_handling.htm>

Implementarea unui ascultător de evenimente se poate realiza prin mai multe modalități, cele mai utilizate sunt următoarele:

1. se utilizează fișierul activity\_main.xml pentru a specifica un event handler
2. clasa specifică activității implementează interfața de tip Listener

public calss ExpEctivity extends Activity implements OnClickListener

{ …

Button

button=(Button)findViewById(R.id.buttonNext); button.setOnClickListener(this);

…

public void onClick(View v) { }

}

1. utilizarea unei clase inner și anonimă

Button btn = (Button)findViewById(R.id.btn)

btn.setOnClickListener(new OnClickListener()

{ @Override

public void onClick(View view)

{ // Event management } });

În aplicațiile dezvoltate până în prezent am utilizat prima modalitate, definind pentru fiecare eveniment o metodă onClick(). În cele ce urmează voi exemplifica a treia modalitate, realizând implementarea metodei care tratează evenimentul folosind o clasă anonimă.

**Aplicație practică 3**: Construiți o aplicație Android care afișează la pornirea sa o listă de culori preferate, iar la selecția unui opțiuni din listă afișează un mesaj de tip Toast ce conține culoarea selectată.

**Rezolvare:**

Pentru implementarea listei se va folosi metoda dinamică, iar pentru tratarea evenimentului se va folosi modalitatea a treia, respectiv utlizarea unei clase anonime.

**Pas 1:** Se deschide un nou proiect Android de tip Empty Activity

**Pas 2:** Se deschide fișierul activity\_main.xml, și se selectează din plateta de componente grafice view-ul ListView

În urma acestei acțiuni activity\_main.xml conține detalii despre componenta grafică activity\_main.xml

<**ListView  
 android:id="@+id/myListView"  
 android:layout\_width="409dp"  
 android:layout\_height="729dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"** />

Ulterior, se stabilește un id pentru view-ul lista.

**Pas 3:** În fișierul activity\_mai.java, realizam următoarele etape:

* mai întâi legatura dintre obiectul ListView și lista definita în fișierul xml.

ListView myListView = (ListView) findViewById(R.id.***myListView***);

* definim un obiect de tip array pentru a specifica lista de culori

**final** ArrayList<String> colors = **new** ArrayList<String>();  
 colors.add(**"Rosu"**);  
 colors.add(**"Albastru"**);  
 colors.add(**"Alb"**);  
 colors.add(**"Verde"**);

* definim un adaptor de tip tablou pentru a realiza legătura dintre tabloul Java si view-ul lista

ArrayAdapter<String> arrayAdapter = **new** ArrayAdapter<String>(**this**, android.R.layout.***simple\_expandable\_list\_item\_1***, colors);

* setăm pentru view-ul listă adaptorul creat anterior

myListView.setAdapter(arrayAdapter);

* setăm ascultătorul specific selecției unei opțiuni din listă și tratăm evenimentul (afișarea unui mesaj de tip Toast), implementând metoda abstractă onItemClick

myListView.setOnItemClickListener(**new** AdapterView.OnItemClickListener() {  
 @Override  
 **public void** onItemClick(AdapterView<?> parent, View view, **int** position, **long** id) {  
 Toast.*makeText*(MainActivity.**this**, colors.get(position), Toast.***LENGTH\_SHORT***).show();  
  
 }

});

Se poate observa cum metoda setOnItemClickListener care setează un ascultător este de tip callback.

Metoda onItemClick()are următoarele argumente:

1. AdapterView<?> parent prin care se specifică containerul în care are loc evenimentul (aici este containerul parinte)
2. un obiect de tip View, clasă generală pentru orice componentă grafică din android, prin care se specifică view-ul asupra căruia are loc interacțiunea
3. un argument de tip intreg position care specifică idexul opțiunii din listă (indexarea începe de la 0)
4. un argument de tip long id prin care se specifică id-ul rândului din listă

Mai jos este listată, integral, sursa Java:

**public class** MainActivity **extends** AppCompatActivity {  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
  
 ListView myListView = (ListView) findViewById(R.id.***myListView***);  
  
 **final** ArrayList<String> colors = **new** ArrayList<String>();  
 colors.add(**"Rosu"**);  
 colors.add(**"Albastru"**);  
 colors.add(**"Alb"**);  
 colors.add(**"Verde"**);  
  
  
 ArrayAdapter<String> arrayAdapter = **new** ArrayAdapter<String>(**this**, android.R.layout.***simple\_expandable\_list\_item\_1***, colors);  
 myListView.setAdapter(arrayAdapter);  
  
 myListView.setOnItemClickListener(**new** AdapterView.OnItemClickListener() {  
 @Override  
 **public void** onItemClick(AdapterView<?> parent, View view, **int** position, **long** id) {  
 Toast.*makeText*(MainActivity.**this**, colors.get(position), Toast.***LENGTH\_SHORT***).show();  
  
 }  
 });  
  
 }  
}

**Aplicații propuse:**

1. Implementați o aplicație Android care afișează următorul ecran ce conține un logo și o listă. Să se afișeze un mesaj cu denumirea băuturii selectate.



1. Implementați o aplicație Android care conține un TimerSpiner și completează dinamic o listă cu multiplii numărului de pe bara Spiner.



**Hint:** componenta grafică folosită seekBarTimer, iar obiectul java este de tip SeekBar.