Java code onderdelen

Variabelen

Java maakt zoals elke programmeertaal gebruik van variabelen. Variabelen zijn geheugenplaatsen waarvan de inhoud kan gewijzigd worden.

Je kan op verschillende plaatsen in de Java code deze variabelen gebruiken: in een klasse of in functies. Met het woord "private" geef je aan dat de inhoud van deze variabelen bereikbaar is binnen die klasse maar niet van buitenaf.

In functies moet je het woord "private" niet gebruiken. Deze variabelen zijn enkel bereikbaar binnen die functie.

Voor elke variabele moet je het datatype opgeven van de inhoud. Je kan kiezen uit zeer veel mogelijkheden. Deze datatypes zijn gekoppeld aan een bibliotheek (zie hieronder bij "imports").

Met het woord "final" geef je aan dat de inhoud van de variabele niet meer wijzigt nadat er een waarde is aan toegekend.

Je kan bij de declaratie van variabelen er reeds een inhoud aan toekennen (initialiseren).

Hoewel niet verplicht is het aan te raden de variabelen een goede naam te geven. Een voorvoegsel gebruiken waarvan het datatype kan afgeleid worden maakt jouw Java code beter leesbaar.

```
private ListView lvitems;
private ArrayList<String> lvstrings = new ArrayList<String>();
private ArrayAdapter<String> lvadapter;
private Spinner spitems;
private ArrayList<String> spstrings = new ArrayList<String>();
private ArrayAdapter<String> spadapter;
private CalendarView cal1;
private Button btn1;
private Button btn2;
private CharSequence mDrawerTitle;
private CharSequence mTitle;
private String[] mFragmentTitles;
private Long rowid;
private EditText ettitel;
private DBHelper objDbHelper;
final String mTitle = appfolder;
String etinput = input.getText().toString();
FileWriter writer;
SQLiteDatabase db = this.getWritableDatabase();
ContentValues values = new ContentValues();
```

Constanten

Een constante is een geheugenplaats waarvan de inhoud niet meer wijzigt nadat er een waarde is ingestopt.

Je kan de "scope" (bereikbaarheid) aangeven door het woord "private" of "public". Publieke constanten zijn in meerdere functies van de klasse te gebruiken.

```
private static String url_lijst = "http://domein.be/.../lijst.php";
private static final String TAG_SUCCESS = "success";
public static final String FLD_NOTA_TITEL = "titel";
```

Imports

Voor elk onderdeel in jouw Java code moet je een bibliotheek importeren. Als je een onderdeel voor de eerste keer gebruikt zal dit in de editor in het rood aangeduid worden. Klik je erop en gebruik je de toets combinatie Alt+Enter dan wordt de juiste "import" automatisch toegevoegd in de lijst.

```
import android.app.Activity;
import android.support.v7.app.ActionBarActivity;
import android.support.v7.app.ActionBar;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.os.Build;
import android.os.Bundle;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.support.v4.widget.DrawerLayout;
import android.widget.ArrayAdapter;
import android.widget.TextView;
import java.util.ArrayList;
import java.util.List;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import android.app.AlertDialog;
import android.os.AsyncTask;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.Spinner;
import android.widget.Toast;
```

If (...) { } else { }

Om de inhoud van variabelen te testen op een bepaalde waarde kan je het "if" statement gebruiken. Tussen de ronde haakjes kan je de voorwaarde(n) plaatsen.

Met de operator "==" test je of de inhoud gelijk is aan een bepaalde waarde. Uiteraard kan je ook andere operatoren gebruiken (bvb. >, <, <=, >=, !=, enz…)

Met een logische operator kan je meerdere testen samenvoegen.

Als het resultaat van de voorwaarde "waar" ("true") is zal het gedeelte tussen de accolades uitgevoerd worden. Is het resultaat "onwaar" ("false") en er is een "else" gedeelte aanwezig dan zal de code tussen de accolades van het "else" gedeelte uitgevoerd worden.

```
if (etinput == null || etinput.trim().length() == 0) {
    etinput = appfolder;
}
```

Let op: voor een String variabele moet je de methode "equals" gebruiken om de inhoud te testen (en dus niet de operator "==") !!!

```
if (success.equals("1")) {
   new notaUpdate().execute();
} else {
   new notaInsert().execute();
}
```

Switch (...) { case ...: case ...: default: }

Als je de inhoud van een variabele met meerdere mogelijke waarden wil testen kan je gebruik maken van het "switch" statement. Voor elke mogelijke waarde voorzie je een "case" blok. Achter de dubbelepunt (":") komen de regels die moeten uitgevoerd worden.

In Android krijgt elk onderdeel automatisch een uniek nummer toegekend. Deze nummers worden verzameld in de klasse "R" en in verschillende subklassen zoals bvb. "drawable", "id", "layout", "menu", "string", "style". Android Studio genereert deze nummers voor je.

```
switch(item.getItemId()) {
case R.id.menu_settings:
   //startActivity(new Intent(this, Preferences.class));
    return true;
case R.id.menu_endapp:
   finish();
   return true;
case android.R.id.home:
   Intent intent = new Intent(this, MainActivity.class);
   intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
   startActivity(intent);
   break;
default:
    return super.onOptionsItemSelected(item);
}
```

In het "default" gedeelte wordt de code uitgevoerd als geen van de andere testen "waar" zijn.

```
switch (number) {
    case 1:
        mTitle = getString(R.string.zoeken);
        break;
    case 2:
        mTitle = getString(R.string.scholen);
        break;
    case 3:
        mTitle = getString(R.string.lokalen);
```

```
      break;
}
```

Als je meer dan 2 testen moet doen is het aangeraden om een "switch" statement te gebruiken.

For (... ; ... ; ...)

Met het "for" statement kan je een lus-structuur maken. Je geeft de startwaarde, de eindwaarde en de verhoging op tussen de ronde haakjes. De code tussen de accolades wordt herhaald volgens de waarden die zijn opgegeven tussen de ronde haakjes.

Dit statement wordt gebruikt als je op voorhand weet hoeveel keer een code blok moet uitgevoerd worden.

```
for(int i = 0; i < tekstlen; i++) {
    char kar1 = tekst.charAt(i);
    int asc1 = (int) kar1;
    int ascres;
    res = res + Character.toString ((char) ascres);
}
```

For (... : ...)

Met het "for-each" statement kan je een code blok laten uitvoeren voor elk onderdeel van een collectie. In het voorbeeld wordt de "titel" en de "id" van elke "nota" van de "notas" collectie toegevoegd aan een array.

```
for (Nota nota : notas) {
   notasarray.add(nota.gettitel());
   notasids.add(String.valueOf(nota.getid()));
}
```

While (...) { }

Zolang aan de voorwaarde die tussen de ronde haakjes staat is voldaan zullen de instructies tussen de accolades uitgevoerd worden.

```
while((read = in.read(buffer)) != -1){
  out.write(buffer, 0, read);
}
```

Bij het inlezen van een tekst lijn per lijn kan je testen of de ingelezen lijn nog bestaat met de voorwaarde "!= null" (niet gelijk aan de waarde "null").

```
while (myLine != null) {
      if (myLine.contains(";")) {
         String paknaam = myLine.substring(myLine.indexOf(";")+1);
         String titel = myLine.substring(0,myLine.indexOf(";")-1);
         titelstrings.add(titel);
         packagestrings.add(paknaam);
      }
      myLine = reader.readLine();
}
```

Do { } while (...)

De instructies tussen de accolades worden uitgevoerd zolang er aan de voorwaarde tussen de ronde haakjes is voldaan.

```
do {
  Nota nota = new Nota();
  nota.setid(Long.parseLong(cursor.getString(0)));
  nota.settitel(cursor.getString(1));
  nota.settekst(cursor.getString(2));
  notaList.add(nota);
} while (cursor.moveToNext());
```

Functies

Je kan een functie maken door de "scope" aan te geven: "public" of "private".

Daarna geef je het datatype op van de terugkeerwaarde (return value). Dit kan "void" (leeg) zijn als er geen waarde moet teruggegeven worden aan het einde van de functie. Als er toch een datatype is opgegeven dan moet er een return statement in de functie staan met daarachter een variabele of waarde van het juiste datatype.

```
public NotasFragment() {
}
@Override
public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setHasOptionsMenu(true);
}
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
      Bundle savedInstanceState) {
   return inflater.inflate(R.layout.fragment_notas, container, false);
}
public void onActivityCreated(Bundle savedInstanceState) {
   super.onActivityCreated(savedInstanceState);
   try {
      ...
   } catch (Exception e) {
     Toast.makeText(getActivity(), e.toString(), Toast.LENGTH_SHORT).show();
   }
}
```

Je kan ook eigen functies maken. Tussen de ronde haakjes kan je argumenten (parameters) meegeven indien nodig.

```
public void leegmaken_velden() {
  ettitel.setText("");
  ettekst.setText("");
  ettitel.requestFocus();
  rowid = (long) 0;
}
```

```
public boolean existsactiviteit(String naam) {
   String selectQuery = "SELECT  * FROM " + TABLE_ACTIVITEITEN + " WHERE
                     naam='" + naam + "'";
   SQLiteDatabase db = this.getReadableDatabase();
   Cursor cursor = db.rawQuery(selectQuery, null);
   if (cursor.getCount() > 0) {
```

```
        return true;
    }
    cursor.close();
    db.close();
    return false;
}
```

```
public void haalrecord(String id) {
    rowid = id;
    new tabelSelect().execute();
}
```

Try { } catch (Exception e) { }

Als je niet zeker bent dat de instructies die je geschreven hebt goed zullen uitgevoerd worden kan je de instructies in een "try"-"catch" blok zetten.

Als er in het "try" gedeelte een fout optreedt tijdens de uitvoering dan zal het programma verdergaan in het gepaste "catch" gedeelte en de fout registreren of tonen.

```
try {
   ...

} catch (UnsupportedEncodingException e) {
    e.printStackTrace();
} catch (ClientProtocolException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
} catch (Exception e) {
    Log.e("Get content", "Error = " + e.toString());
}
```

Multi-threading

Je kan een stukje programma in de achtergrond laten uitvoeren met een "AsyncTask". De instructies in die klasse zullen in een aparte thread van het Android besturingssysteem uitgevoerd worden.

In een "AsyncTask" moeten 3 functies (methodes) aanwezig zijn: onPreExecute(), doInBackground() en onPostExecute().

```
class tabelSelect extends AsyncTask<String, String, String> {

    @Override
    protected void onPreExecute() {
        super.onPreExecute();
    }
    protected String doInBackground(String... args) {
        List<NameValuePair> params = new ArrayList<NameValuePair>();
        params.add(new BasicNameValuePair("LokaalID", rowid));
        JSONObject json = jParser.makeHttpRequest(url_select, "POST", params);
        Log.d("Select: ", json.toString());
        try {
```

```
            int success = json.getInt(TAG_SUCCESS);
            if (success == 1) {
                JSONArray LokaalObj = json.getJSONArray(TAG_TABEL);
                jobjtabel = LokaalObj.getJSONObject(0);
            } else {
                // geen record
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }
        return null;
    }
    protected void onPostExecute(String file_url) {
        if (jobjtabel != null) {
            getActivity().runOnUiThread(new Runnable() {
                public void run() {
                    etlokaalnaam = (EditText)
                                    getActivity().findViewById(R.id.et_lokaalnaam);
                    etlokaalindelingurl = (EditText)
                               getActivity().findViewById(R.id.et_lokaalindelingurl);
                    try {
                      etlokaalnaam.setText(
                                        jobjtabel.getString(TAG_LOKAALNAAM));
                       etlokaalindelingurl.setText(
                                 jobjtabel.getString(TAG_LOKAALINDELINGURL));
                    } catch (JSONException e) {
                        e.printStackTrace();
                    }
                }
            });
        }
    }
}
```

Met de functie runOnUiThread() van de activiteit kan je de resultaten van de AsyncTask functie doInBackground() gebruiken om bvb. tekstvakken in te vullen.