

Week 2 - oefening 1: Stopafstand Android

Doelstelling

Creatie van eerste Android-applicatie

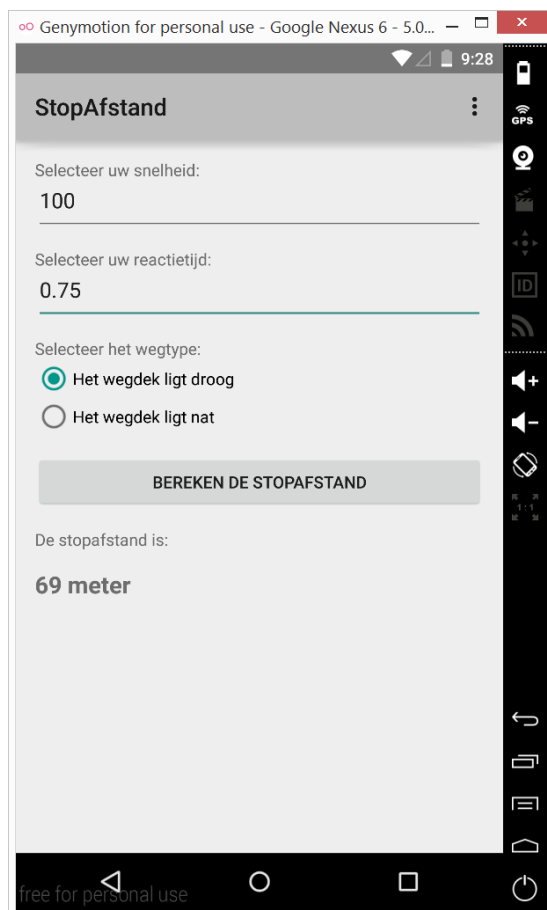
Algemene beschrijving

Voor deze en volgende opgaves is het noodzakelijk dat alle software beschreven in het document 'Installatie Software' (zie Leho) geïnstalleerd is.

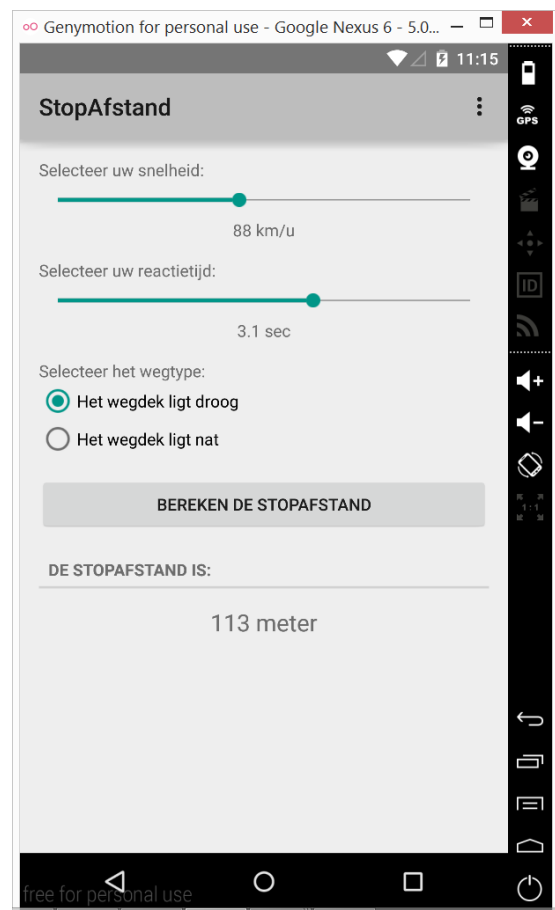
We zetten onze console java-applicatie 'StopAfstand' (cfr. Opgave 1 week 1) om in een eerste werkende Android Applicatie. De klasse StopAfstandInfo wordt hergebruikt.

We maken twee verschillende versies:

Versie 1



Versie 2 (uitbreiding)

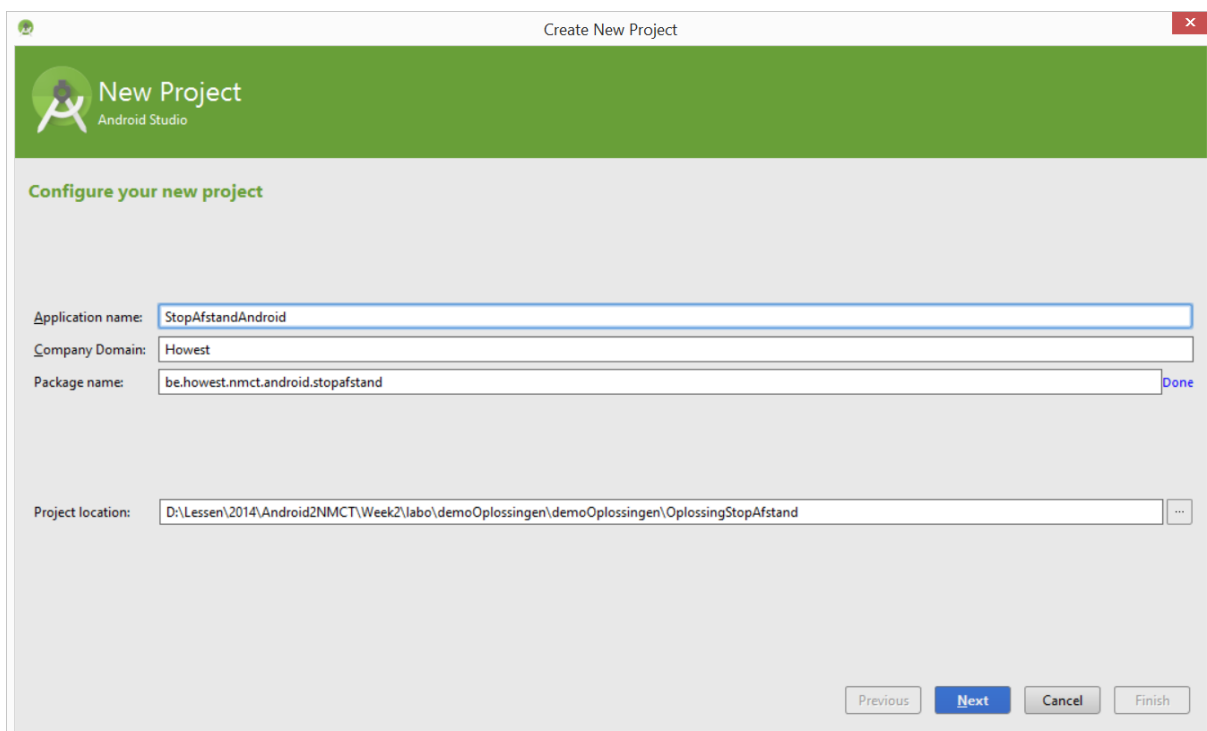
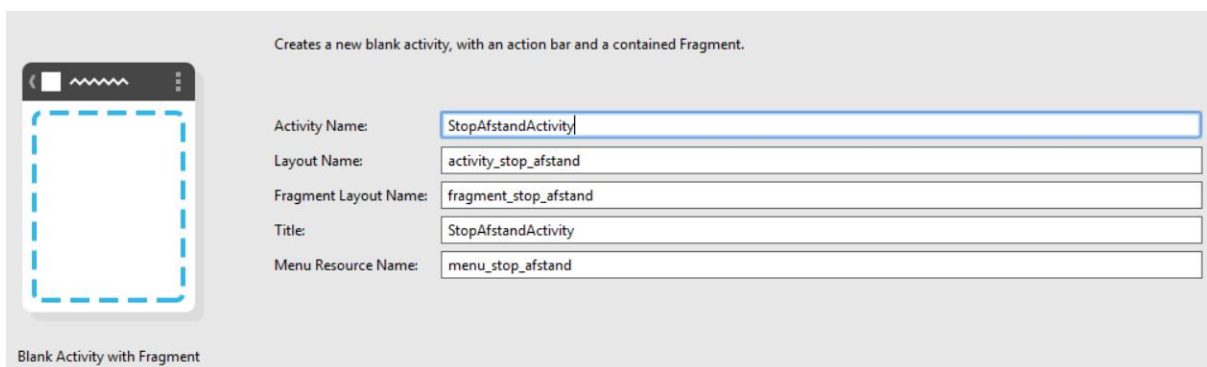


Veel documentatie is te vinden op: developer.android.com

Voorbereiding

Maak een nieuw Android Application Project aan met volgende instellingen.

Application Name:	StopAfstand
Package Name:	be.howest.nmct.android.stopafstand
Target SDK:	21
Activity Name:	StopAfstandActivity
Path:	<zelf te kiezen>

Ga zelf na waar in jouw project deze informatie bewaard wordt. Probeer ook even andere vermelde informatie te achterhalen. Waar wordt het gebruikte launch-icoon geplaatst?

Doorgrond de structuur van jouw project. Som de verschillende mappen op. Ga map per map na wat al dan niet aanwezig is.

Maak onder de src-dir een nieuwe package 'be.howest.nmct.cars' aan. Kopieer jouw klasse 'StopAfstandInfo' uit week 1 hieronder.

Design versie 1

Ga na waar de xml-layout file van zowel Activity als Fragment zich situeren:

- Wat is de layout van de activity?
- Wat is de layout van het fragment?

Voor deze versie hebben we voldoende met één Fragment. De verdere ontwikkeling concentreert zich volledig op dit ene Fragment.

Meer informatie over de Layout is te vinden op:

<http://developer.android.com/guide/topics/ui/layout/relative.html>

Meer informatie over de verschillende controls ('views') is te vinden op:

<http://developer.android.com/guide/topics/ui/controls.html>

Maak gebruik van Android Studio om de verschillende controls ('views') te selecteren. Bestudeer ook altijd het resultaat in het gegenereerde xml-file. Na verloop van tijd is het handiger/snelser om het xml-file rechtstreeks te bewerken.

Geef elke (nuttige) view een goede naam. Bestudeer aandachtig de xml-file!

Ga na waar string-constanten (bv. voor de tekst van een TextView) bewaard worden. Wat is het nut hiervan?

Zorg ervoor dat de twee EditText-views resp. enkel gehele (voor snelheid) en enkel decimale (voor reactietijd) getallen toestaan. Meer info is te vinden op:

<http://developer.android.com/guide/topics/ui/controls/text.html>

Tot slot, lees ook even enkele stijlregels na op:

<http://developer.android.com/design/style/metrics-grids.html>

Meer info wordt in de theorieles gegeven.

Code versie 1

Bekijk de Activity-klasse horend bij de activity-layout: ga waar de layoutfile van het fragment opgeroepen wordt.

Dit zullen we wijzigen: we wensen de code horend bij een fragment in een afzonderlijke klasse plaatsen.

Er zijn twee verschillende werkwijzes:

1. Een klasse toevoegen en deze laten overerven van de klasse Fragment.
2. Via Android studio een Blank Fragment laten toevoegen. Voor- of nadeel is dat hierbij ook heel wat meer code dan nodig automatisch toegevoegd wordt. Deze methode is handig als men ook communicatie tussen verschillende fragments/activities wens te verzorgen.

Methode 1:

Volg hiertoe volgende stappen:

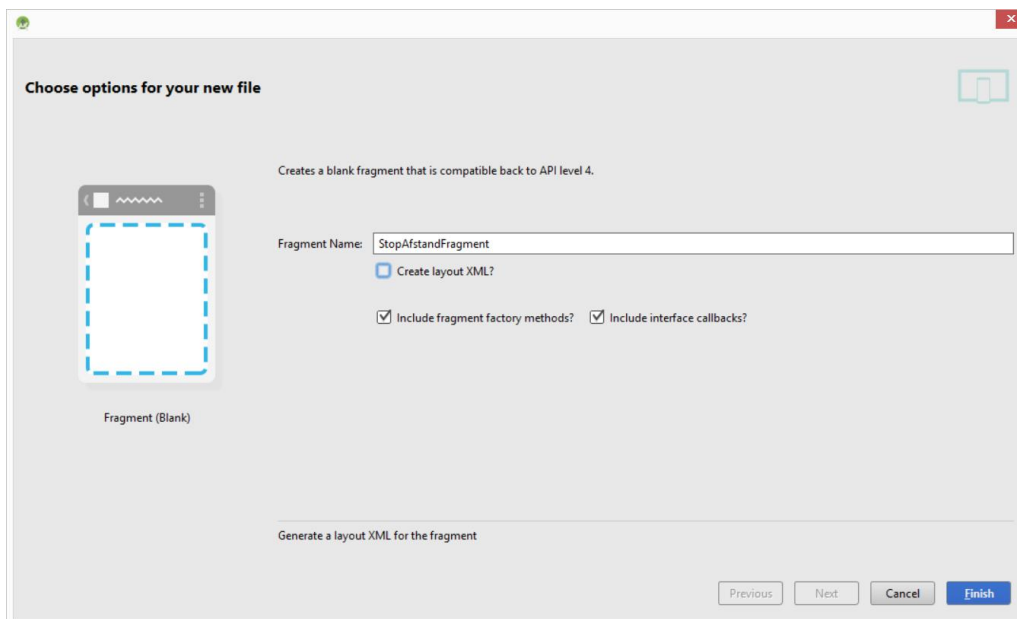
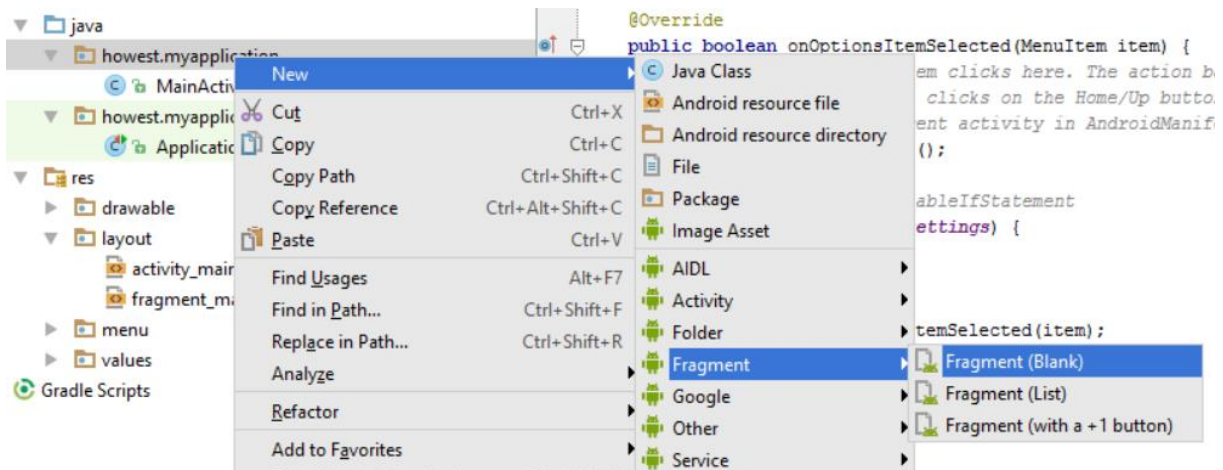
1. Voeg een nieuwe klasse 'StopAfstandFragment' in dezelfde package van de activity toe.
2. Laat de klasse overerven van de klasse Fragment. Opmerking (voor later): de klasse moet steeds over een defaultconstructor bezitten.
3. Overschrijf de methode 'onCreateView'.

Methode 2:

Volg hiertoe volgende stappen:

1 Selecteer de juiste package

2 Voeg een 'Blank Fragment' toe, met als naam 'StopAfstandFragment'. Zorg dat er **geen** layout extra wordt gegenereerd!



Vanaf hier lopen beide methodes opnieuw samen:

In de methode 'onCreateView' roepen we de correcte (reeds bestaande) layout-file op:

Info: <http://developer.android.com/reference/android/view/LayoutInflater.html>

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                        Bundle savedInstanceState) {
    //TextView textView = new TextView(getActivity());
    //textView.setText(R.string.hello_blank_fragment);

    // Inflate the layout for this fragment
    View v = inflater.inflate(R.layout.fragment_stop_afstand, container, false);

    return v;
}
```

In de activity passen we de code in de onCreate-methode aan:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_stop_afstand);
    if (savedInstanceState == null) {
        getSupportFragmentManager().beginTransaction()
            .add(R.id.container, new StopAfstandFragment()) //<---!!!
            .commit();
    }
}
```

We wensen de stopafstand te berekenen wanneer de gebruiker op de resp. button klikt. Hiervoor is het noodzakelijk alle informatie uit de juiste views op te halen.

Pas daarom volgende strategie toe:

1. Declareer bovenaan in het fragment de noodzakelijke views als attributen.

```
private EditText uwSnelheidView;
private EditText uwReactietijdView;
private TextView uwStopafstandView;
private RadioButton droogWegTypeView;
private Button buttonStopafstand;
```

2. In de onCreateView-methode worden de aangemaakte attributen gekoppeld aan de views beschreven in de layout-file. Dit gebeurt via de methode `findViewById`: hierbij wordt een specifieke View via het 'Id'-attribuut uit het overeenkomstige xml-layout file opgehaald.

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                        Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    View v = inflater.inflate(R.layout.fragment_stop_afstand, container, false);
    this.uwSnelheidView = (EditText) v.findViewById(R.id.jouw_snelheid);
    //...
```

3. Het click-event op een button gaan we in de code via een anonieme klasse afhandelen:

```
this.buttonStopafstand = (Button) v.findViewById(R.id.buttonStopafstand);
this.buttonStopafstand.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        toonStopAfstand(v);
    }
});
```

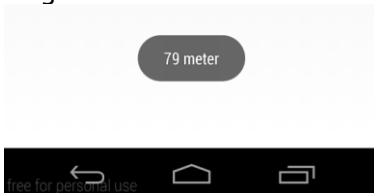
<http://developer.android.com/guide/topics/ui/controls/button.html#ClickListener>

4. Hierdoor kunnen de properties van de verschillende attributen in andere methodes opgevraagd worden.

```
public void toonStopAfstand(View v) {
    int snelheid = Integer.parseInt(this.uwSnelheidView.getText().toString());
    //...
```

Vervolledig de methode dat het click-event van de button afhandelt. Zorg dat de stopafstand weergegeven wordt. Maak nu gebruik van jouw klasse StopAfstandInfo!

Opmerking: soms kan het handig zijn om specifieke informatie snel via een soort van 'pop-up' aan de gebruiker te tonen. Binnen android spreekt men over een Toast.



De basis hierover vind je op: <http://developer.android.com/guide/topics/ui/notifiers/toasts.html>

Probeer dit zeker even uit!

Indien er interactie met de gebruiker nodig is, dan zijn Notifications geschikter. Dit komt later aan bod.

Design versie 2

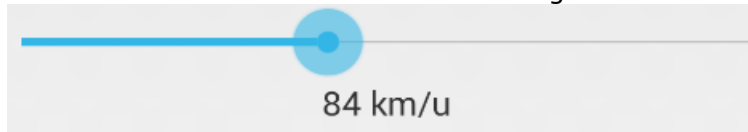
Maak in het zelfde project onder de package 'be.howest.nmct.android.stopafstand' een nieuw Fragment aan. Merk op dat er een extra xml-layout file meegegenereerd wordt.

Het tweede design verschilt in weinig van het eerste design. Je kan het design vanuit het ene xml-file in het andere kopiëren.

Volgende zaken zijn evenwel verschillend:

- Gebruik een seekBar om de gebruiker zijn snelheid en reactietijd te laten opgeven. Meer info over het gebruik van de seekbar is te vinden op: <http://developer.android.com/reference/android/widget/SeekBar.html>
Van welke klasse is de Seekbar afgeleid?
Zoek zelf de juiste attributen om minimum, maximum en (start)waarde in te stellen.

- Plaats onder elke Seekbar een extra TextView waarin we de waarde van de seekbar wensen te tonen. Geef de seekbar en textview een logische naam.



Vul de code aan zodat we een analoge werking als Versie 1 hebben.

Zorg ervoor dat bij het wijzigen van de seekbar het bijhorende TextView aangepast wordt. Hiervoor hebben we een listener nodig. Maak een innerklasse die de juiste interface implementeert. Maak een object hiervan dat luistert naar de Seekbar.

<http://developer.android.com/reference/android/widget/SeekBar.OnSeekBarChangeListener.html>

Een goede start vind je alvast hier:

<http://webtutsdepot.com/2011/12/03/android-sdk-tutorial-seekbar-example/>