

## Week 5 - oefening 2: backstack & navigation back

### Doelstelling

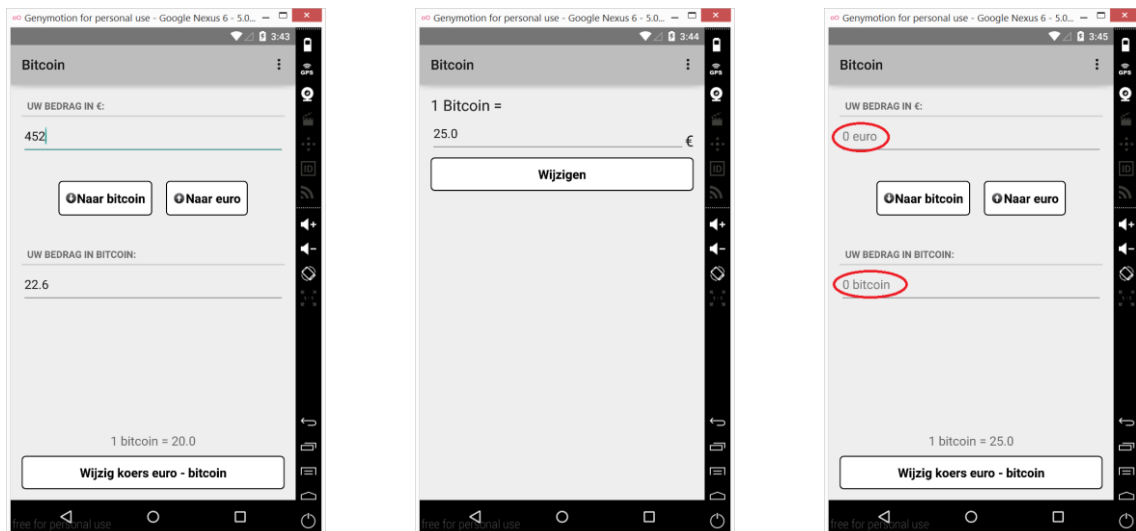
In deze opgave focussen we ons op het gebruik van de 'back'-toets en hieraan gekoppelde backstack.

### Voorbereiding

Werk verder op het Bitcoinproject van oefening 1.

### Code – aanpassing 1


Zorg ervoor dat de tekstvelden in changeFragment opnieuw ingevuld worden wanneer dit fragment opnieuw zichtbaar wordt.



Gebruik opnieuw de onSaveInstanceState-methode uit oefening 3 van week 3 om zo de toestand van Fragment bij te houden en terug te plaatsen.

<http://developer.android.com/guide/components/fragments.html>

*Also like an activity, you can retain the state of a fragment using a Bundle, in case the activity's process is killed and you need to restore the fragment state when the activity is recreated. You can save the state during the fragment's onSaveInstanceState() callback and restore it during either onCreate(), onCreateView(), or onActivityCreated(). For more information about saving state, see the Activities document.*

Wat stel je vast? Hoe verklaar je dit? 

### Code – aanpassing 2

Hierboven werden de tekstvelden niet terug gezet (ondanks het gebruik van onSaveInstanceState). Controleer de methode 'showFragmentChange' Telkens wordt er een nieuwe instantie van de ChangeFragment-klasse aangemaakt en getoond. Dit kan niet de bedoeling zijn...

Daarom maken we gebruik van de backstack:

<http://developer.android.com/training/implementing-navigation/temporal.html>

In vele gevallen kunnen we via de backstack activities of fragments bijhouden tijdens de navigatie door jouw applicatie.

Pas volgende aan:

- Waar wordt de changeFragment-klasse voor het eerst gebruikt? Plaats het object op de backstack via de methode 'addToBackStack'. De string-parameter is optioneel en weerspiegelt een naam voor de bewaarde toestand

```
if (savedInstanceState == null) {
    getFragmentManager().beginTransaction()
        .add(R.id.container, ChangeFragment.newInstance(currentRateBitcoinInEuro), "changeFragment").addToBackStack("start_ChangeFragment")
        .commit();
}
```

- Pas de methode 'showFragmentChange' aan zodat er geen nieuwe instantie meer van het ChangeFragment aangemaakt wordt, maar de reeds bestaande van de backstack opgevraagd wordt. Geef de nieuwe wisselkoers via een set-methode door

```
ChangeFragment fragment = (ChangeFragment) getFragmentManager().findFragmentByTag("changeFragment");
fragment.setRate1BitcoinInEuro(newBitcoinRate);
```

Controleer of opgegeven waarden terug geplaatst worden wanneer de gebruiker terugkeert in ChangeFragment

Controleer of de nieuwe wisselkoers getoond wordt.

## Code – aanpassing 3

Bestudeer de werking van de 'back'-toets in de app.


Wat stel je vast? Welk gedrag zou je veronderstellen


- bij het klikken op de back-toets wanneer changeFragment zichtbaar is?
- Bij het klikken op de back-toets wanneer bitcoinRateFragment zichtbaar is?

Algemene info is te vinden op:

<http://developer.android.com/guide/components/tasks-and-back-stack.html>

**Stap 1:** Om ervoor te zorgen dat de gebruiker van bitcoinRateFragment via de Back-toets terug kan navigeren naar changeFragment, dient ook bitcoinRateFragment op de backstack komen te staan. Vul aan in de methode 'showFragmentBitcoinRate'. Controleer!

Kan de gebruiker terug navigeren? 

Wat gebeurt er als de gebruiker indien de gebruiker nogmaals op de back-toets drukt? 



**Stap 2:** zolang de app draait is steeds één changeFragment op de backstack. Om te vermijden dat de gebruiker een lege activity ziet bij nogmaals op de back-toets te drukken, overschrijven we het gedrag 'onBackPressed'

```
//er zal steeds minimum 1 fragment op de backstack zitten.
@Override
public void onBackPressed() {
    if (getFragmentManager().getBackStackEntryCount() == 1) {
        finish();
    }
    else {
        super.onBackPressed();
    }
}
```