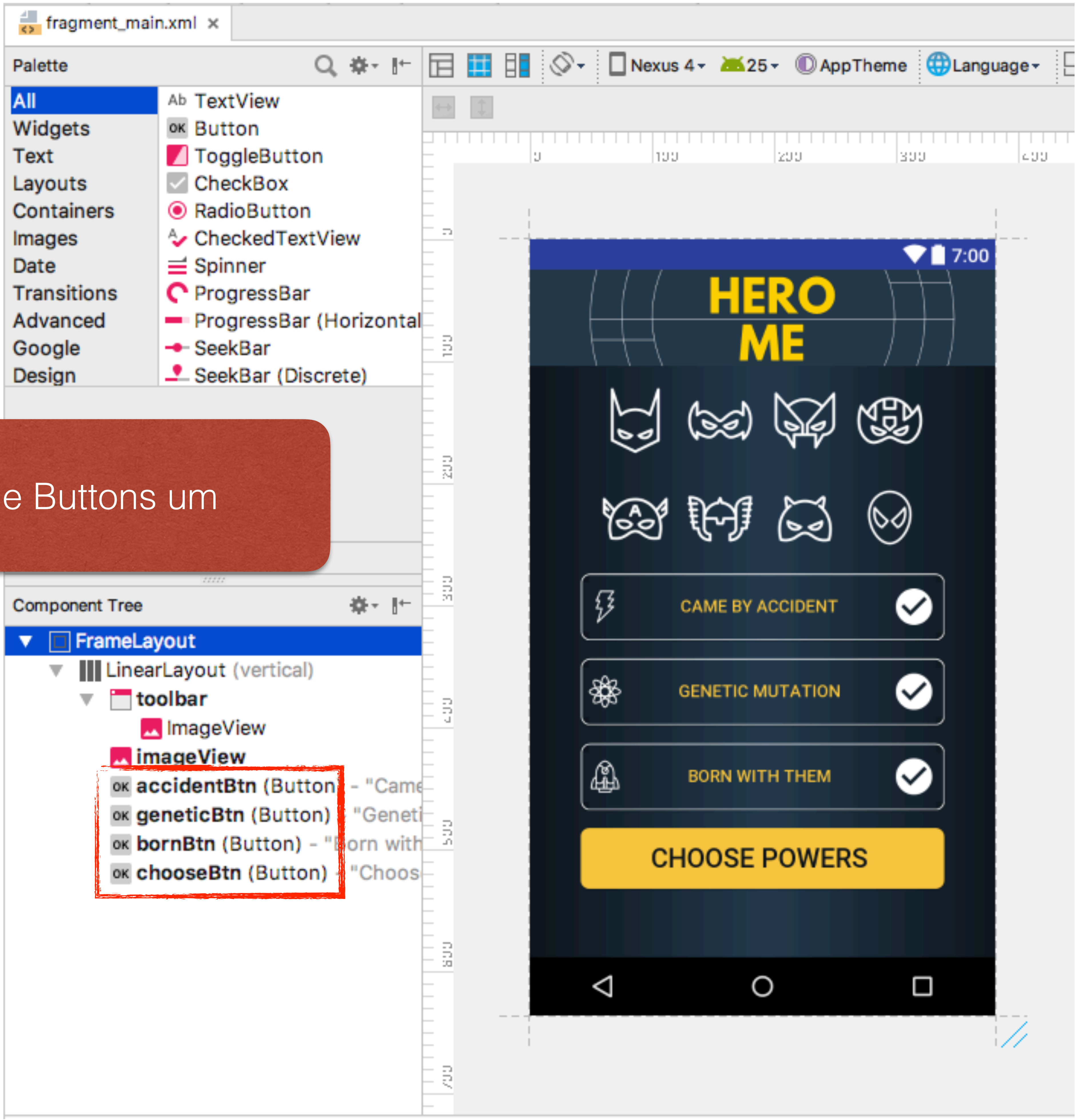
Fragments

HeroMe

Teil 2



```
FrameLayout
  LinearLayout (vertical)
    toolbar
      ImageView
    imageView
    accidentBtn (Button) - "Came
    geneticBtn (Button) - "Geneti
    bornBtn (Button) - "Born with
    chooseBtn (Button) - "Choos
```

Wir benennen die Buttons um

MainFragment.java

```
public class MainFragment extends Fragment {

    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";

    private String mParam1;
    private String mParam2;

    private Button accidentBtn;
    private Button geneticBtn;
    private Button bornButton;
    private Button chooseButton;

    private MainFragmentInteractionListener mListener;

    public MainFragment() {
        // Required empty public constructor
    }
```

MainFragment.java

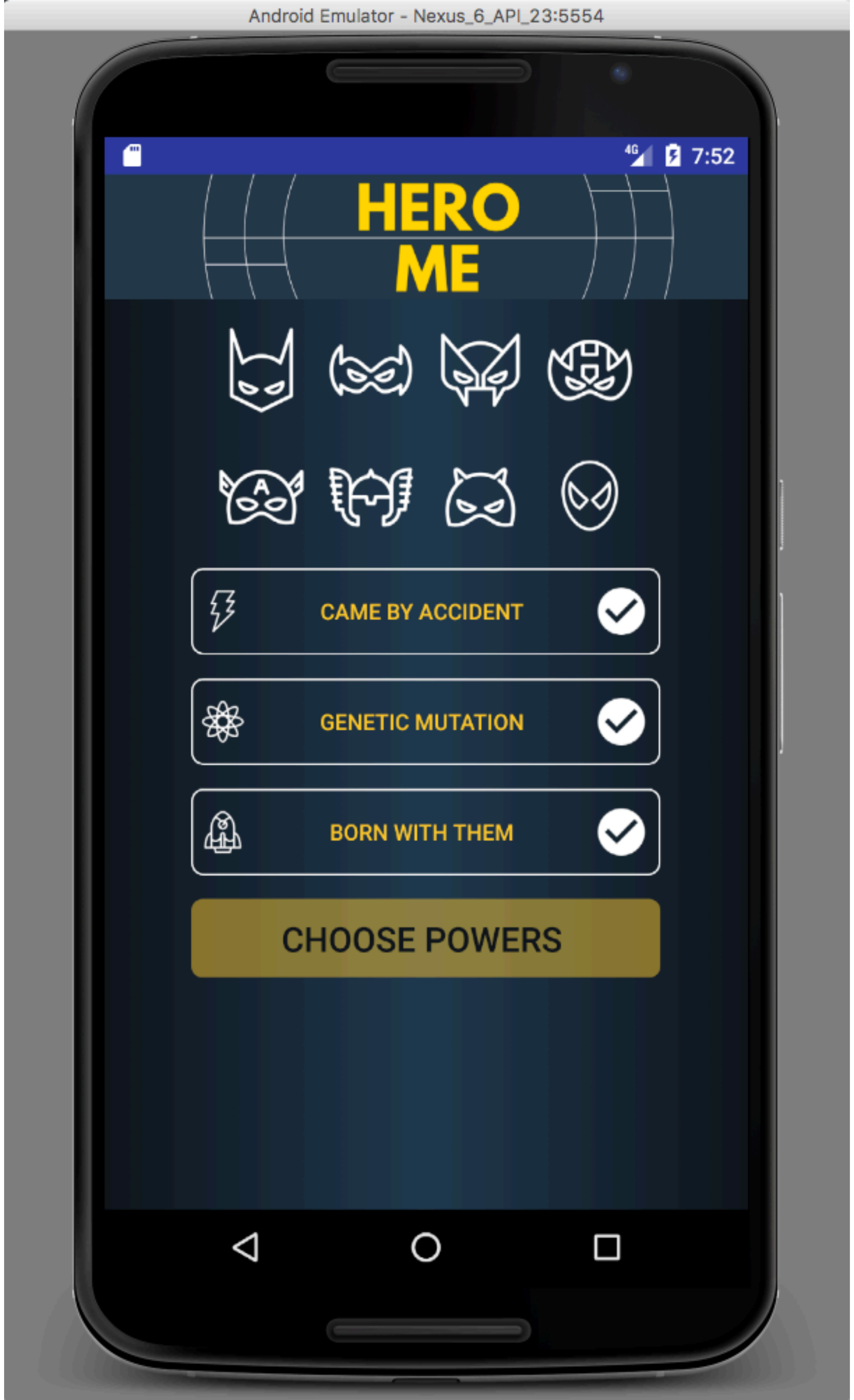
```java
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {

    View view = inflater.inflate(R.layout.fragment_main, container, false);

    accidentBtn = (Button) view.findViewById(R.id.accidentBtn);
    geneticBtn = (Button) view.findViewById(R.id.geneticBtn);
    bornBtn = (Button) view.findViewById(R.id.bornBtn);
    chooseBtn = (Button) view.findViewById(R.id.chooseBtn);

    // Inflate the layout for this fragment
    return view;
}
```

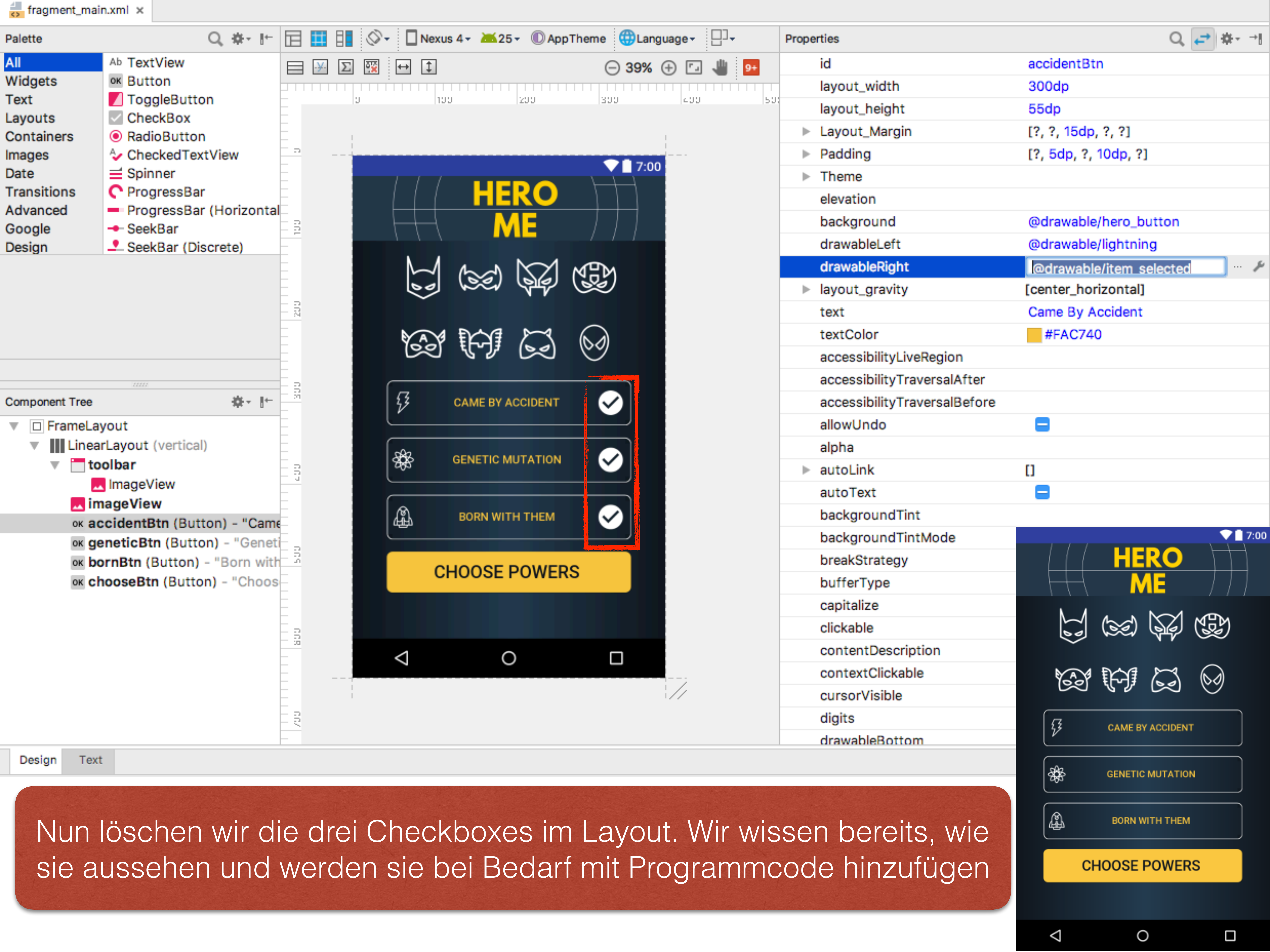
Da wir uns in einem Fragment und nicht in einer Activity befinden, reicht ein einfaches findViewById(…) nicht, sondern diese Funktion muss auf einer View ausgeführt werden. Diese View wird erstellt (inflate) und anschließend werden darauf die Buttons angelegt. Schließlich wird die erstellte View mit return zurückgegeben.

MainFragment.java

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {

    View view = inflater.inflate(R.layout.fragment_main, container, false);

    accidentBtn = (Button) view.findViewById(R.id.accidentBtn);
    geneticBtn = (Button) view.findViewById(R.id.geneticBtn);
    bornBtn = (Button) view.findViewById(R.id.bornBtn);
    chooseBtn = (Button) view.findViewById(R.id.chooseBtn);

    chooseBtn.setEnabled(false);
    chooseBtn.getBackground().setAlpha(128);

    // Inflate the layout for this fragment
    return view;
}
```

Nun word der „CHOOSE POWERS“ Button diabled. Da dies dem BEnutzer aber nicht sichtbar gemacht wird, wird zusätzlich noch die Transparenz des Buttons halbiert (0-255)


Android Emulator - Nexus_6_API_23:5554
7:52
HERO ME
CAME BY ACCIDENT
GENETIC MUTATION
BORN WITH THEM
CHOOSE POWERS



fragment_main.xml

Property	Value
id	accidentBtn
layout_width	300dp
layout_height	55dp
Layout_Margin	[?, ?, 15dp, ?, ?]
Padding	[?, 5dp, ?, 10dp, ?]
Theme	
elevation	
background	@drawable/hero_button
drawableLeft	@drawable/lightning
drawableRight	@drawable/item_selected
layout_gravity	[center_horizontal]
text	Came By Accident
textColor	#FAC740
accessibilityLiveRegion	
accessibilityTraversalAfter	
accessibilityTraversalBefore	
allowUndo	
alpha	
autoLink	[]
autoText	
backgroundTint	
backgroundTintMode	
breakStrategy	
bufferType	
capitalize	
clickable	
contentDescription	
contextClickable	
cursorVisible	
digits	
drawableBottom	

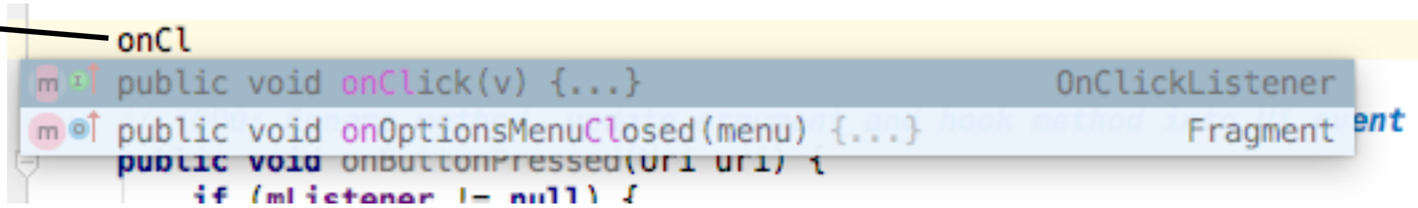
Component Tree
- FrameLayout
  - LinearLayout (vertical)
    - toolbar
      - ImageView
    - imageView
    - accidentBtn (Button) - "Came
    - geneticBtn (Button) - "Geneti
    - bornBtn (Button) - "Born with
    - chooseBtn (Button) - "Choos

Nun löschen wir die drei Checkboxes im Layout. Wir wissen bereits, wie sie aussehen und werden sie bei Bedarf mit Programmcode hinzufügen

OnClickListener

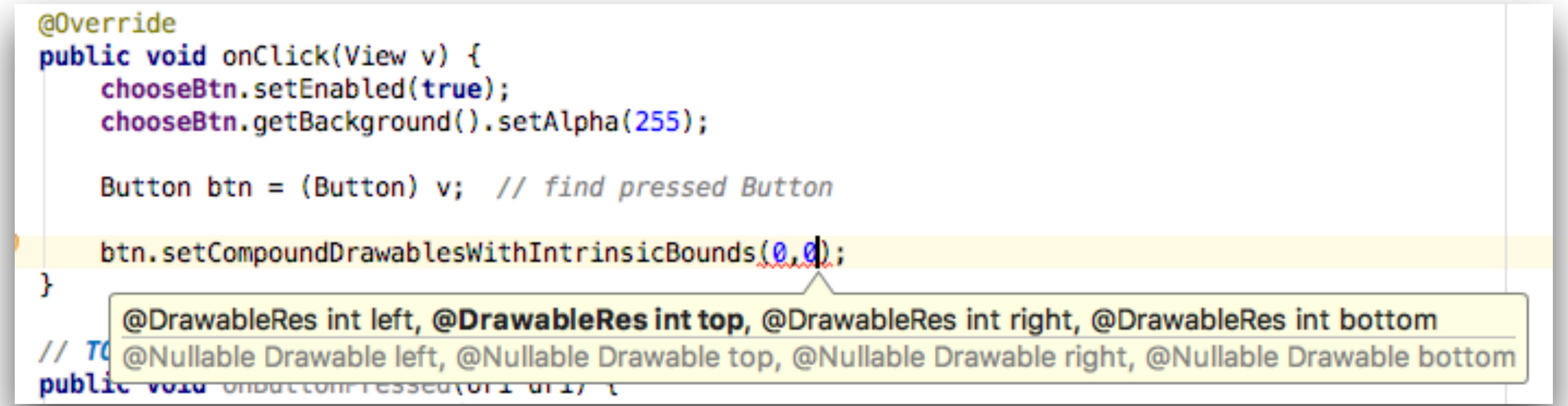
```
public class MainFragment extends Fragment implements View.OnClickListener {
    ...
    public MainFragment() { }

    public static MainFragment newInstance(String param1, String param2) {...}

    @Override
    public void onCreate(Bundle savedInstanceState) {...}

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {

        View view = inflater.inflate(R.layout.fragment_main, container, false);

        accidentBtn = (Button) view.findViewById(R.id.accidentBtn);
        geneticBtn = (Button) view.findViewById(R.id.geneticBtn);
        bornBtn = (Button) view.findViewById(R.id.bornBtn);
        chooseBtn = (Button) view.findViewById(R.id.chooseBtn);

        accidentBtn.setOnClickListener(this);

        chooseBtn.setEnabled(false);
        chooseBtn.getBackground().setAlpha(128);

        // Inflate the layout for this fragment
        return view;
    }

    @Override
     public void onClick(View v) {

     }

    public void onButtonPressed(Uri uri) {...}

    @Override
    public void onAttach(Context context) {...}

    @Override
    public void onDetach() {...}


    public interface MainFragmentInteractionListener {...}
}
```

Wir wollen nun einen gemeinsamen Listener für alle Buttons erstellen. Dazu implementieren wir im Fragment den OnClickListener. Wir registrieren nun das Fragment beim Button (this). Dadurch wird die implementierte Methode onClick() aufgerufen



```
onCl
public void onClick(v) {...}                        OnClickListener
public void onOptionsMenuClosed(menu) {...}                Fragment
public void onButtonPressed(Uri uri) {
    if (mListener != null) {
```

MainFragment.java

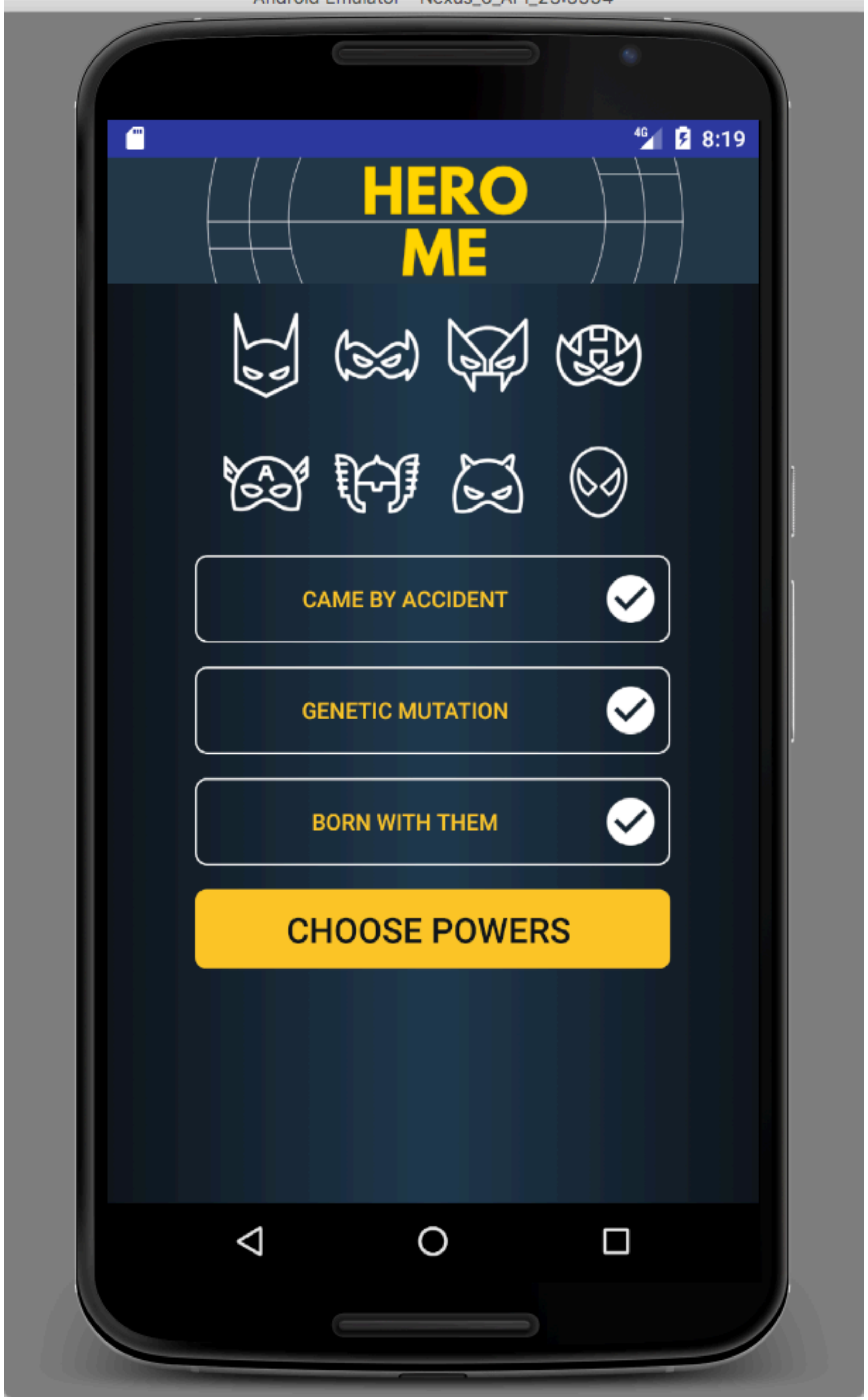


```
@Override
public void onClick(View v) {
    chooseBtn.setEnabled(true);
    chooseBtn.getBackground().setAlpha(255);

    Button btn = (Button) v;  // find pressed Button

    btn.setCompoundDrawablesWithIntrinsicBounds(0,0);
}

// TO
public void onButtonPressed(Uri uri) {
```

@DrawableRes int left, **@DrawableRes int top**, @DrawableRes int right, @DrawableRes int bottom
@Nullable Drawable left, @Nullable Drawable top, @Nullable Drawable right, @Nullable Drawable bottom

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {

    View view = inflater.inflate(R.layout.fragment_main, container, false);

    accidentBtn = (Button) view.findViewById(R.id.accidentBtn);
    geneticBtn = (Button) view.findViewById(R.id.geneticBtn);
    bornBtn = (Button) view.findViewById(R.id.bornBtn);
    chooseBtn = (Button) view.findViewById(R.id.chooseBtn);

    accidentBtn.setOnClickListener(this);
    geneticBtn.setOnClickListener(this);
    bornBtn.setOnClickListener(this);

    chooseBtn.setEnabled(false);
    chooseBtn.getBackground().setAlpha(128);

    // Inflate the layout for this fragment
    return view;
}

@Override
public void onClick(View v) {
    chooseBtn.setEnabled(true);
    chooseBtn.getBackground().setAlpha(255);

    Button btn = (Button) v;  // find pressed Button

    btn.setCompoundDrawablesWithIntrinsicBounds(0,0, R.drawable.item_selected, 0);
}
```

Egal, welcher Button geclickt wird - die Methode onClick(View v) wird aufgerufen.



Problem 1:
Man kann mehrere Checkboxen gleichzeitig aktivieren

Problem 2:
Man verliert das linke Icon dabei

Wir müssen also in der onClick(View v) - Methode darauf reagieren, welche Button geklickt wurde.

MainFragment.java

```java
@Override
public void onClick(View v) {
    chooseBtn.setEnabled(true);
    chooseBtn.getBackground().setAlpha(255);

    // Alle Checkboxes werden gelöscht
    accidentBtn.setCompoundDrawablesWithIntrinsicBounds(R.drawable.lightning,0,0,0);
    geneticBtn.setCompoundDrawablesWithIntrinsicBounds(R.drawable.atomic,0,0,0);
    bornBtn.setCompoundDrawablesWithIntrinsicBounds(R.drawable.rocket,0,0,0);

    Button btn = (Button) v;  // find pressed Button
    int leftDrawable = 0;

    // Beim Button, bei dem die Checkbox gesetzt wird, muss das Icon links neu gesetzt
werden
    if (btn == accidentBtn) {
        leftDrawable = R.drawable.lightning;
    } else if (btn == geneticBtn) {
        leftDrawable = R.drawable.atomic;
    } else if (btn == bornBtn) {
        leftDrawable = R.drawable.rocket;
    }

    btn.setCompoundDrawablesWithIntrinsicBounds(leftDrawable,0, R.drawable.item_selected,
0);
}
```

MainFragment.java

Refactoring

```
@Override
public void onClick(View v) {
    chooseBtn.setEnabled(true);
    chooseBtn.getBackground().setAlpha(255);

    // Alle Checkboxes werden gelöscht
    accidentBtn.setCompoundDrawablesWithIntrinsicBounds(R.drawable.lightning,0,0,0);
    geneticBtn.setCompoundDrawablesWithIntrinsicBounds(R.drawable.atomic,0,0,0);
    bornBtn.setCompoundDrawablesWithIntrinsicBounds(R.drawable.rocket,0,0,0);

    Button btn = (Button) v;  // find pressed Button
    // drawables for the left, top, right, and bottom borders
    Drawable leftDrawable = btn.getCompoundDrawables()[0];

    btn.setCompoundDrawablesWithIntrinsicBounds(
            leftDrawable,
            null,
            v.getResources().getDrawable(R.drawable.item_selected),
            null);
}
```

Was ist wohl der Unterschied?
Hat dieses Vorgehen Vorteile?

Neues Fragment „öffnen“

- Als nächsten Schritt wollen wir durch Anklicken des Buttons „CHOOSE POWERS“ ein neues Fragment am Screen darstellen

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {

    ...

    accidentBtn.setOnClickListener(this);
    geneticBtn.setOnClickListener(this);
    bornBtn.setOnClickListener(this);

    chooseBtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            MainActivity mainActivity = (MainActivity) getActivity();
            FragmentManager fm = mainActivity.getSupportFragmentManager();
        }
    });

    ...
    return view;
}
```

Man kann aus dem Fragment auf die aufrufende Activity zugreifen und sich den FragmentManager holen.

Allerdings kann es bei vielen Fragments zu Poblemen kommen.

REGEL: Ein Fragment darf nur von der aufrufenden Activity verwaltet werden !!!

Schritt 1: Methode in Activity anlegen

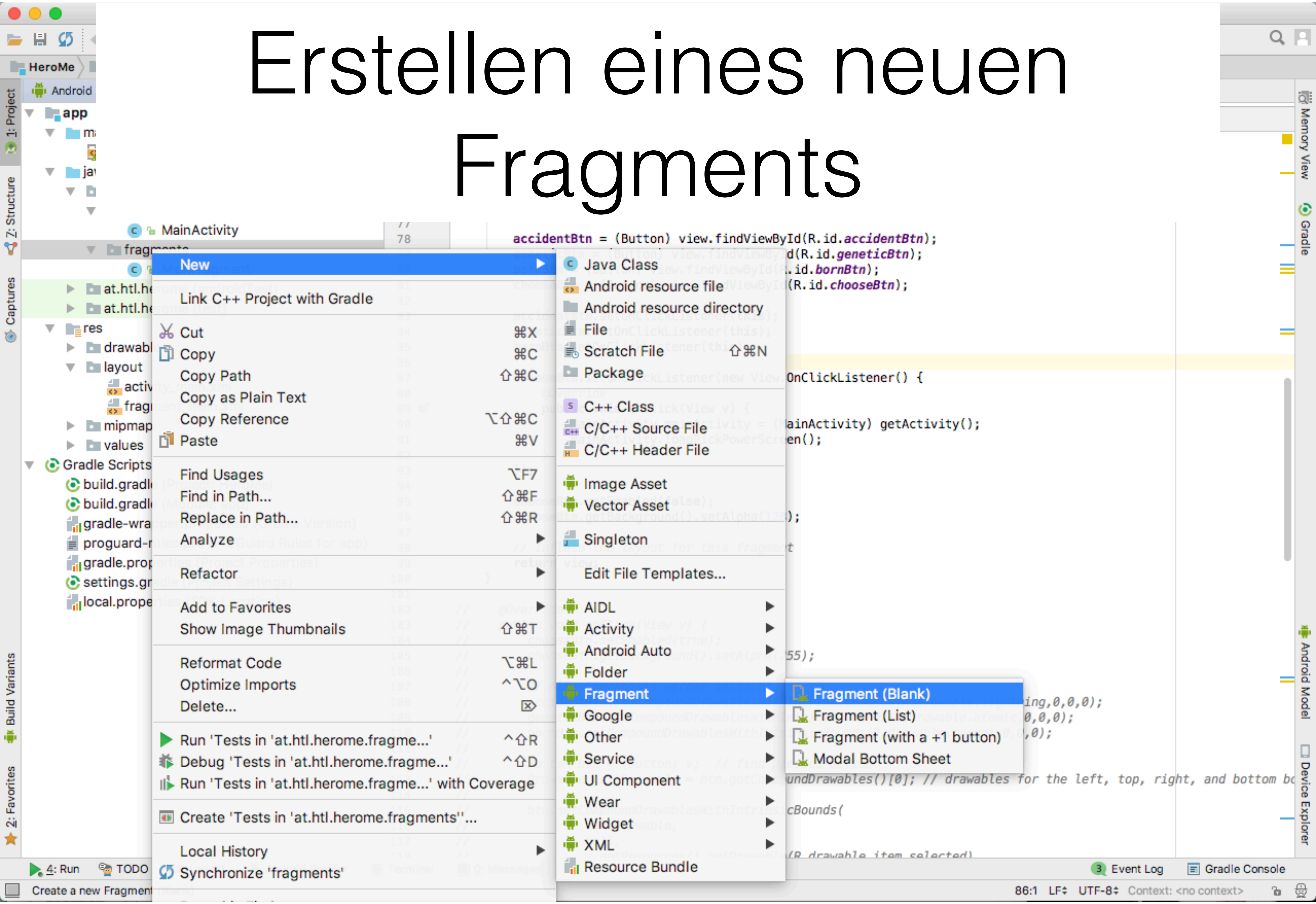
```
public class MainActivity extends AppCompatActivity
                          implements MainFragment.MainFragmentInteractionListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        FragmentManager manager = getSupportFragmentManager();
        Fragment fragment = manager.findFragmentById(R.id.fragment_container);

        if (fragment == null) {
            fragment = new MainFragment();
            manager.beginTransaction().add(R.id.fragment_container, fragment).commit();
        }
    }

    public void loadPickPowerScreen() {
        // to be continued ...
    }

    @Override
    public void onMainFragmentInteraction(Uri uri) {

    }
}
```

Schritt 2: Aufrufen aus Fragment

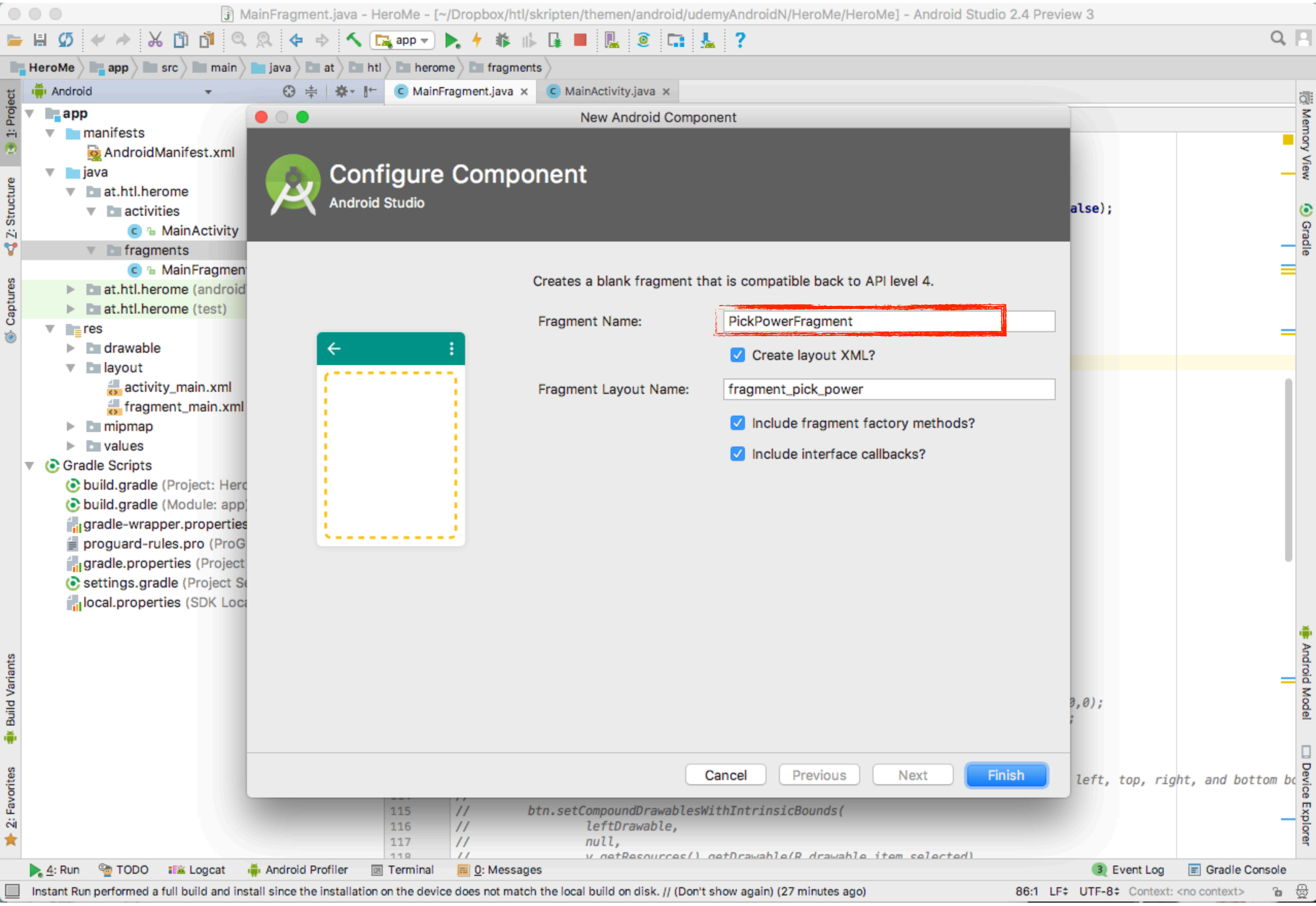
```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {

    ...

    accidentBtn.setOnClickListener(this);
    geneticBtn.setOnClickListener(this);
    bornBtn.setOnClickListener(this);

    chooseBtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            MainActivity mainActivity = (MainActivity) getActivity();
            mainActivity.loadPickPowerScreen();
        }
    });

    ...
    return view;
}
```

Vorteil: Sämtlicher Code zum Verwalten der Fragments ist zentral in der Activity-Klasse

Erstellen eines neuen Fragments



```
accidentBtn = (Button) view.findViewById(R.id.accidentBtn);
...d(R.id.geneticBtn);
....id.bornBtn);
...(R.id.chooseBtn);
...OnClickListener() {
...ainActivity) getActivity();
...en();
```



MainFragment.java - HeroMe - [~/Dropbox/htl/skripten/themen/android/udemyAndroidN/HeroMe/HeroMe] - Android Studio 2.4 Preview 3

New Android Component

Configure Component

Android Studio

Creates a blank fragment that is compatible back to API level 4.

Fragment Name: PickPowerFragment

[✓] Create layout XML?

Fragment Layout Name: fragment_pick_power

[✓] Include fragment factory methods?

[✓] Include interface callbacks?

```
//        btn.setCompoundDrawablesWithIntrinsicBounds(
//                leftDrawable,
//                null,
```

Instant Run performed a full build and install since the installation on the device does not match the local build on disk. // (Don't show again) (27 minutes ago)

Refactoring

```
public class PickPowerFragment extends Fragment {
   ...

    private PickPowerInteractionListener mListener;

    public PickPowerFragment() {
        // Required empty public constructor
    }

    ...

    /**
     * This interface must be implemented by activities that contain this
     * fragment to allow an interaction in this fragment to be communicated
     * to the activity and potentially other fragments contained in that
     * activity.
     * <p>
     * See the Android Training lesson <a href=
     * "http://developer.android.com/training/basics/fragments/communicating.html"
     * >Communicating with Other Fragments</a> for more information.
     */
    public interface PickPowerInteractionListener {
        // TODO: Update argument type and name
        void onPickPowerFragmentInteraction(Uri uri);
    }
}
```

<Shift>-F6

MainActivity.java

```
public class MainActivity extends AppCompatActivity
                           implements MainFragment.MainFragmentInteractionListener,
                                      PickPowerFragment.PickPowerInteractionListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        FragmentManager manager = getSupportFragmentManager();
        Fragment fragment = manager.findFragmentById(R.id.fragment_container);

        if (fragment == null) {
            fragment = new MainFragment();
            manager.beginTransaction().add(R.id.fragment_container, fragment).commit();
        }
    }

    public void loadPickPowerScreen() {
        // to be continued ...
    }

    @Override
    public void onMainFragmentInteraction(Uri uri) {

    }

    @Override
    public void onPickPowerFragmentInteraction(Uri uri) {

    }
}
```

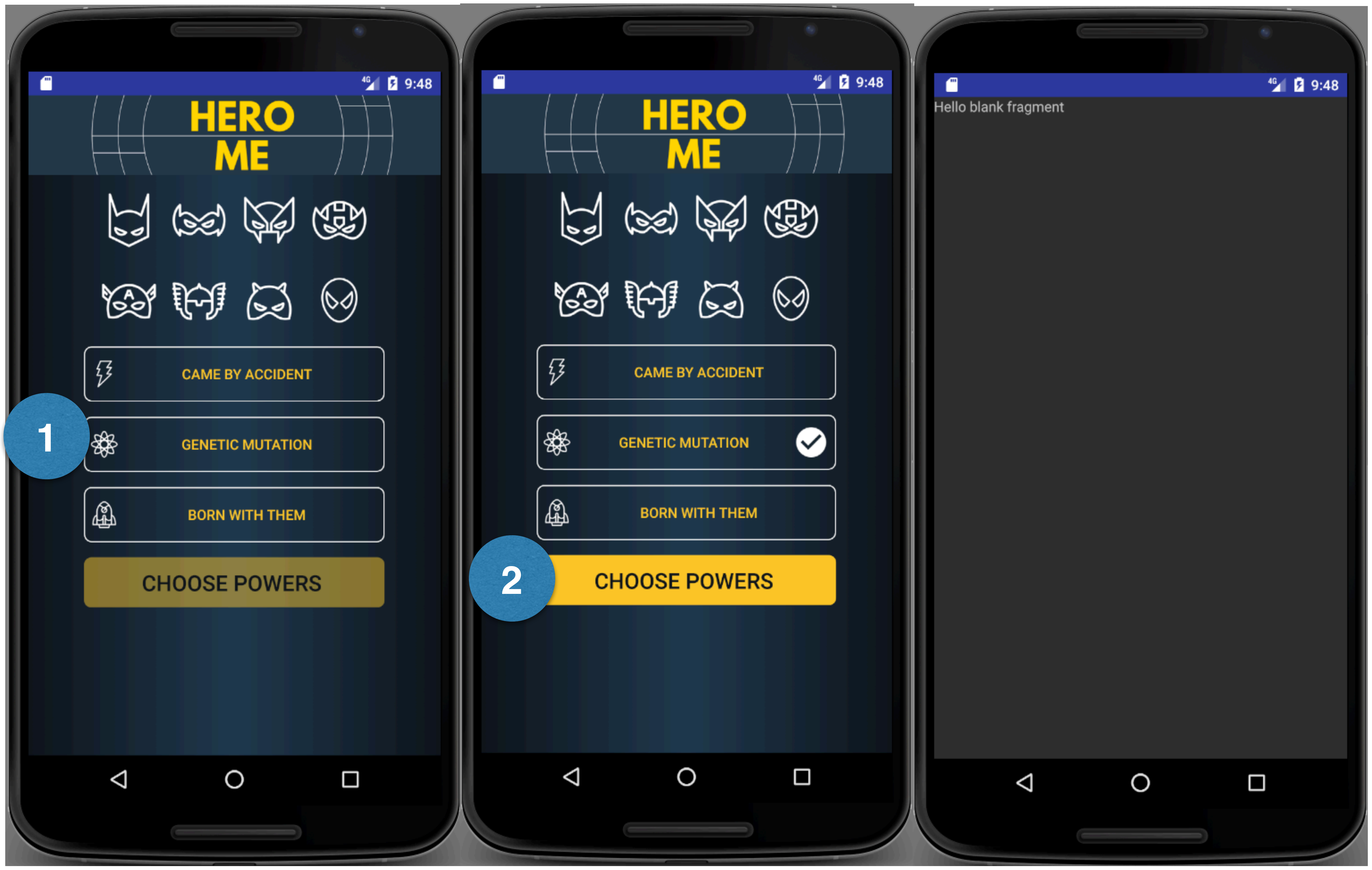
<alt>-Enter zum Implementieren der untenstehenden Methode

Die implementierte Methode onPickPowerFragmentInteraction(…) hat die gleiche Aufgabe wie loadPickPowerScreen(). Allerdings soll die zweite Variante mehr Möglichkeiten bieten.

MainActivity.java

```java
public class MainActivity extends AppCompatActivity
                          implements MainFragment.MainFragmentInteractionListener,
                                     PickPowerFragment.PickPowerInteractionListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...
    }

    public void loadPickPowerScreen() {
        PickPowerFragment pickPowerFragment = new PickPowerFragment();
        this.getSupportFragmentManager()
                .beginTransaction()
                .replace(R.id.fragment_container, pickPowerFragment)
                .addToBackStack(null)
                .commit();
    }

    @Override
    public void onMainFragmentInteraction(Uri uri) {

    }

    @Override
    public void onPickPowerFragmentInteraction(Uri uri) {

    }
}
```

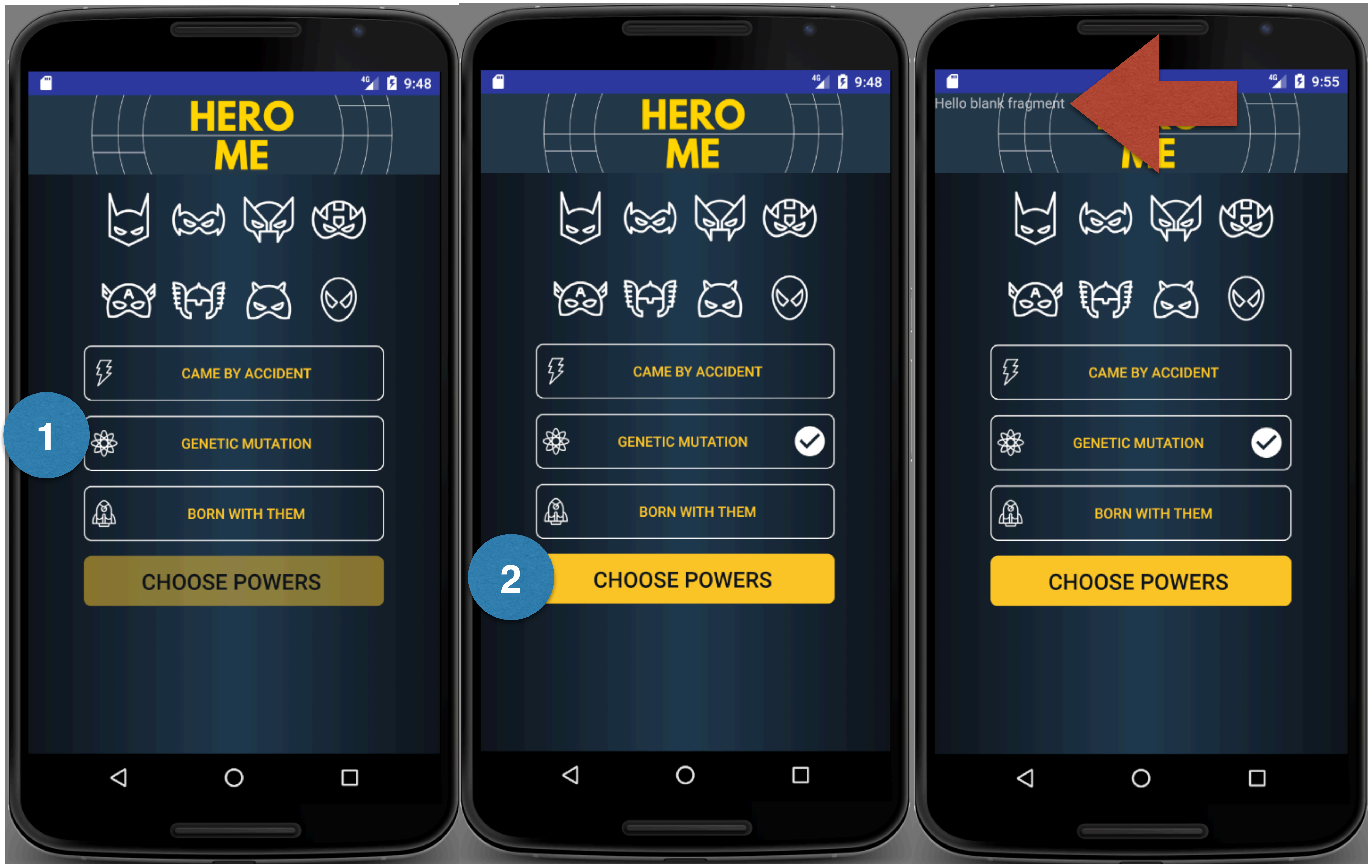
Dadurch kehrt man mit dem Back-Button zum vorherigen Fragment zurück


9:48
HERO ME
CAME BY ACCIDENT
1
GENETIC MUTATION
BORN WITH THEM
CHOOSE POWERS
9:48
HERO ME
CAME BY ACCIDENT
GENETIC MUTATION
BORN WITH THEM
2
CHOOSE POWERS
9:48
Hello blank fragment

MainActivity.java

```java
public class MainActivity extends AppCompatActivity
                         implements MainFragment.MainFragmentInteractionListener,
                                    PickPowerFragment.PickPowerInteractionListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...
    }

    public void loadPickPowerScreen() {
        PickPowerFragment pickPowerFragment = new PickPowerFragment();
        getSupportFragmentManager()
        .beginTransaction()
        .add(R.id.fragment_container, pickPowerFragment)
        .commit();
    }

    @Override
    public void onMainFragmentInteraction(Uri uri) {

    }

    @Override
    public void onPickPowerFragmentInteraction(Uri uri) {

    }
}
```

verwendet man add() anstelle von replace() ergibt dies ein anderes Ergebnis

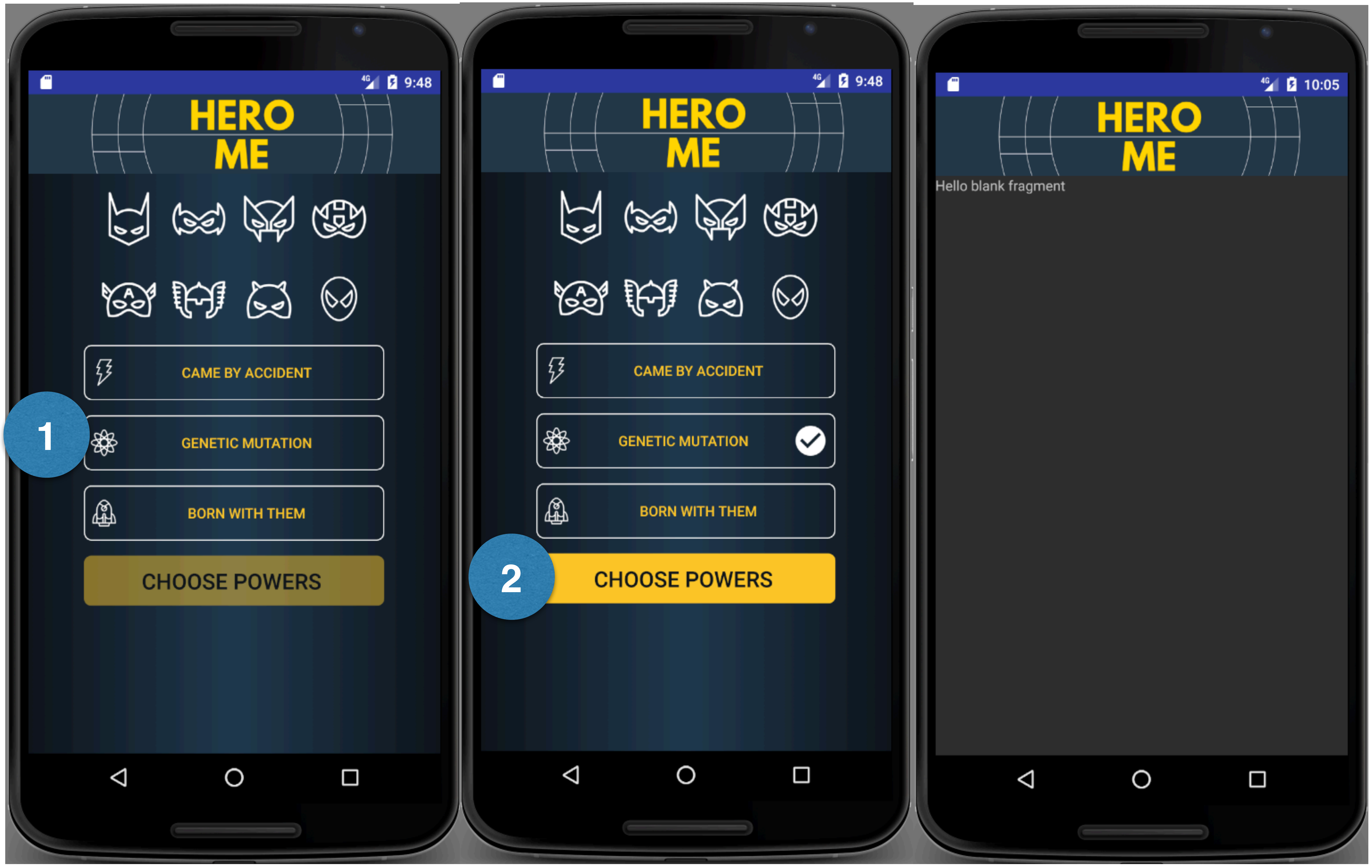

9:48
HERO ME
CAME BY ACCIDENT
GENETIC MUTATION
BORN WITH THEM
CHOOSE POWERS
1
2
9:55
Hello blank fragment

Toolbar auf jedem Screen

- Wir wollen nun die Toolbar auf jedem Screen (Fragment) erscheinen lassen
- Wir könnten die Toolbar ins andere Fragment kopieren
- Dies würde das DRY-Prinzip verletzen (don’t repeat yourself)
- Daher verwenden wir das activity_main.xml für die Toolbar

activity_main.xml

```xml
<?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:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="at.ntl.nerome.activities.MainActivity">

    <android.support.v7.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#253748"
        android:minHeight="?attr/actionBarSize"
        android:theme="?attr/actionBarTheme">

        <ImageView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:src="@drawable/top_banner_bar" />
    </android.support.v7.widget.Toolbar>

    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/fragment_container"
        tools:context=".activities.MainActivity">
    </FrameLayout>

</LinearLayout>
```



9:48
HERO ME
CAME BY ACCIDENT
1
GENETIC MUTATION
BORN WITH THEM
CHOOSE POWERS
9:48
HERO ME
CAME BY ACCIDENT
GENETIC MUTATION
BORN WITH THEM
2
CHOOSE POWERS
10:05
HERO ME
Hello blank fragment

Noch Fragen?