Tarea 6: Haz que tu aplicación sea interactiva

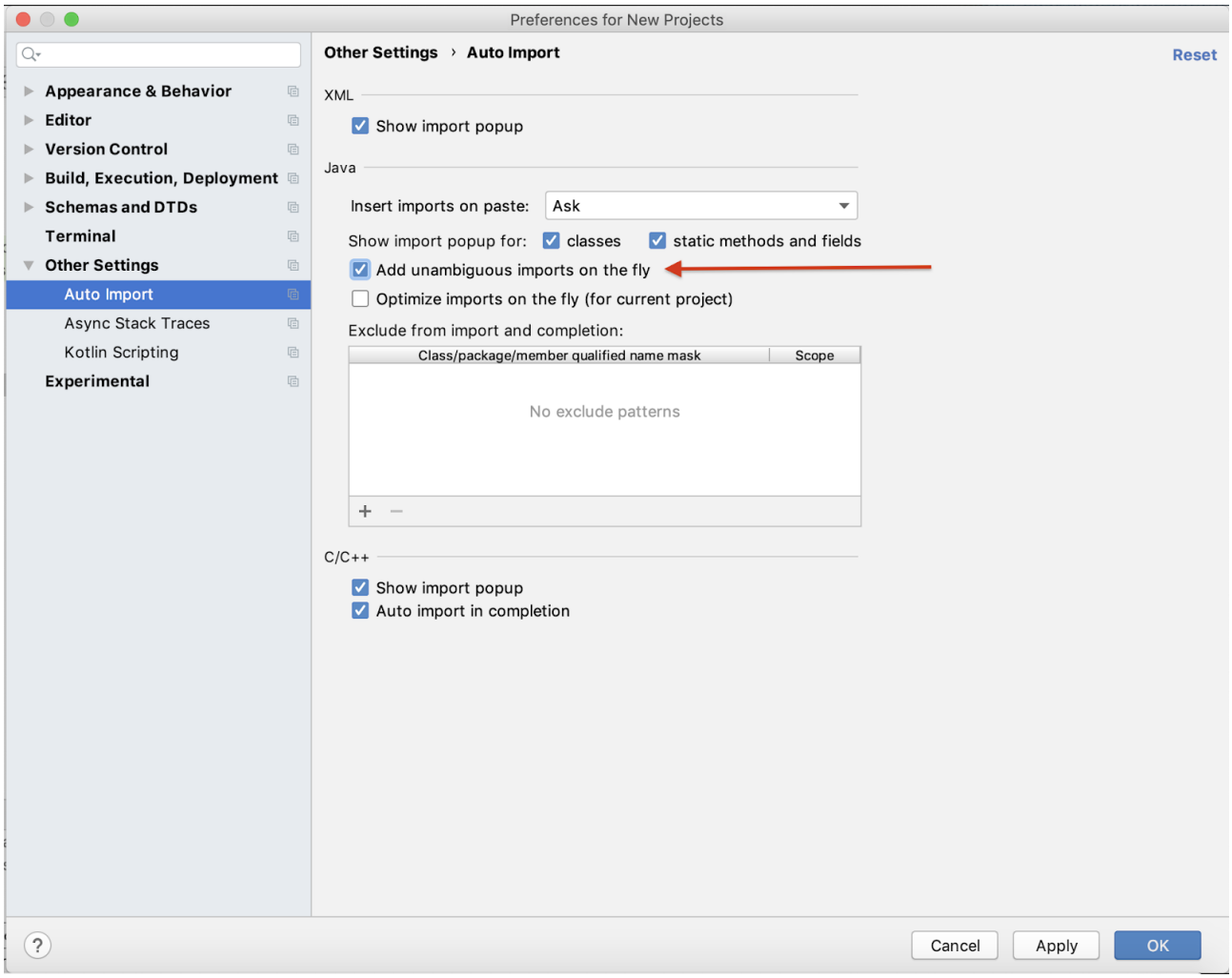
Ha agregado botones a la pantalla principal de su aplicación, pero actualmente los botones no hacen nada. En esta tarea, harás que tus botones respondan cuando el usuario los presione.

- Primero, hará que el **botón Toast** muestre un **mensaje emergente** llamado **toast**.
  A continuación, hará que el botón **Contar** actualice el número que se muestra en **TextView**.

Paso 1: habilite las importaciones automáticas

Para facilitarle la vida, puede habilitar las importaciones automáticas para que Android Studio importe automáticamente cualquier clase que necesite el código Java.

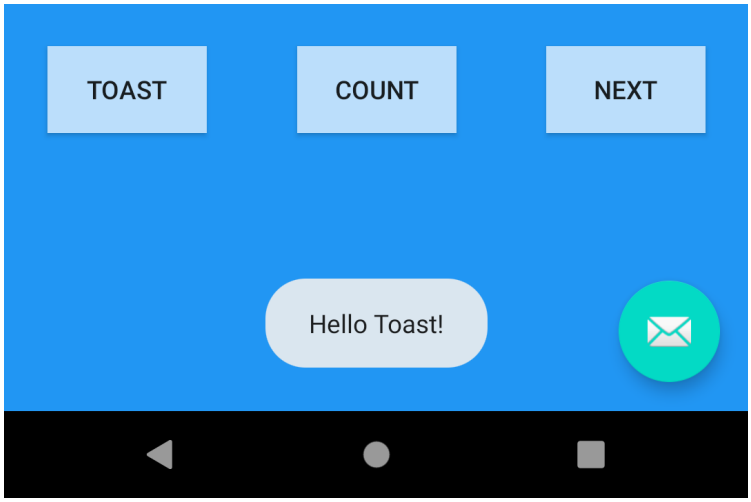
1. En Android Studio, abra el editor de configuraciones yendo a **Archivo > Otras configuraciones > Preferencias para nuevos proyectos.**
2. Seleccione **Importaciones automáticas**. En la sección **Java**, asegúrese de que esté marcada la opción **Agregar importaciones sin ambigüedades sobre la marcha**.
3. Cierre el editor de configuraciones presionando OK.



Paso 2: mostrar un toast

En este paso, adjuntará un método Java al **botón Toast** para mostrar un toast cuando el usuario presione el botón.

Un toast es un mensaje breve que aparece brevemente en la parte inferior de la pantalla.



1. Abra **FirstFragment.java** (**app** > **java** > **com.example.android.myfirstapp** > **FirstFragment**). Esta clase tiene solo dos métodos, **onCreateView()** y **onViewCreated()**. Estos métodos se ejecutan cuando se inicia el fragmento.

Como se mencionó anteriormente, la identificación de una vista lo ayuda a identificar esa vista claramente de otras vistas.
Usando el método **findViewByID()**, su código puede encontrar el botón aleatorio usando su **id**, **R.id.random_button**.

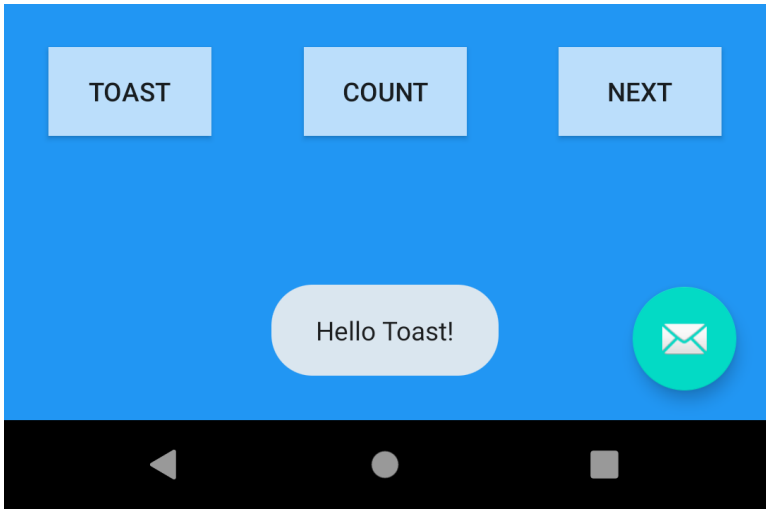
2. Eche un vistazo a **onViewCreated()**. Configura un **Listener (detector de clicks)** para el botón **Ramdom**, que se creó originalmente como el **botón Siguiente. (NEXT)**

```
view.findViewById(R.id.random_button).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        NavHostFragment.findNavController(FirstFragment.this)
                .navigate(R.id.action_FirstFragment_to_SecondFragment);
    }
});
```

Esto es lo que hace este código:
Use el método **findViewById()** con la identificación de la vista deseada como argumento, luego configure un detector de clics **(Listener**) en esa vista.
En el cuerpo del Listener, use una acción, que en este caso es para **navegar a otro fragmento**, y navegue hasta allí.
3. Justo debajo de ese Listener, agregue código para configurar un Listener para el botón de **Toast**, que crea y muestra un **toast**. Aquí está el código:

```
view.findViewById(R.id.toast_button).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Toast myToast = Toast.makeText(getActivity(), "Hello toast!",
Toast.LENGTH_SHORT);
        myToast.show();
    }
});
```

4. Ejecute la aplicación y presione el botón Toast. Ves el mensaje toast en la parte inferior de la pantalla?



4. Si lo desea, extraiga la cadena del mensaje en un recurso como lo hizo con las etiquetas de los botones.
Ha aprendido que para hacer que una vista sea interactiva, necesita configurar un Listener (detector de clics) para la vista **que dice qué hacer** cuando se hace clic en la vista (**botón**).
El Listener puede:

- Implementar una pequeña cantidad de código directamente.
- Llamar a un método que defina el comportamiento de clic deseado en la actividad.

Paso 3: haga que el botón Contar actualice el número en la pantalla
El método que muestra el brindis es muy sencillo; no interactúa con ninguna otra vista en el diseño. En el siguiente paso, agrega comportamiento a su diseño para buscar y actualizar otras vistas.
Actualice el botón Contar para que cuando se presione, el número en la pantalla aumente en 1.

1.En el archivo **fragment_first.xml** note el **id** for the **TextView**:

```
<TextView
    android:id="@+id/textview_first"
```

2. En **FirstFragment.java**, agregue un detector de clics (**Listener**) para **count_button** debajo de los otros detectores de clics en on**ViewCreated()**. Debido a que tiene un poco más de trabajo por hacer, haga que llame a un nuevo método, **countMe().**

```
view.findViewById(R.id.count_button).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        countMe(view);
    }
});
```

3. En la clase **FirstFragment**, agregue el método **countMe()** que toma un solo argumento **View**. Este método se invocará cuando se haga clic en el **botón Contar** y se llame al Listener (detector de clics).

```
private void countMe(View view) {
}
```

4. Obtenga el valor de **showCountTextView**. Lo definirá en el siguiente paso.

```
    ...
     // Get the value of the text view
     String countString = showCountTextView.getText().toString();
```

5. Convierta el valor en un número e increméntelo.

```
    ...
     // Convert value to a number and increment it
     Integer count = Integer.parseInt(countString);
     count++;
```

6. Muestre el nuevo valor en **TextView** configurando mediante programación la propiedad de texto de TextView.

```
    ...
    // Display the new value in the text view.
    showCountTextView.setText(count.toString());
```

Aquí está todo el método:

```
private void countMe(View view) {
    // Get the value of the text view
    String countString = showCountTextView.getText().toString();
    // Convert value to a number and increment it
    Integer count = Integer.parseInt(countString);
    count++;
    // Display the new value in the text view.
    showCountTextView.setText(count.toString());
}
```

Paso 4: Guarde en caché el TextView para uso repetido

Puede llamar a **findViewById()** en **countMe()** para encontrar **showCountTextView**. Sin embargo, se llama a **countMe()** cada vez que se hace clic en el botón, y **findViewById()** es un método relativamente lento para llamar.
Por lo tanto, es mejor encontrar la vista una vez y almacenarla en caché.

1. En la clase **FirstFragment** antes de cualquier método, agregue una variable miembro para **showCountTextView** de tipo **TextView.**

```
TextView showCountTextView;
```

2. En **onCreateView()**, llamará a **findViewById()** para obtener el **TextView** que muestra el conteo. El método **findViewById()** debe llamarse en una vista donde debe comenzar la búsqueda de la ID solicitada, así que asigne la vista de diseño que se devuelve actualmente a una nueva variable, **fragmentFirstLayout**, en su lugar.

```
// Inflate the layout for this fragment
View fragmentFirstLayout = inflater.inflate(R.layout.fragment_first, container, false);
```

3. Llame a f**indViewById()** en **fragmentFirstLayout** y especifique el **id** de la vista que desea buscar, **textview_first**. Guarde en caché ese valor en **showCountTextView**.

```
    ...
     // Get the count text view
     showCountTextView = fragmentFirstLayout.findViewById(R.id.textview_first);
```

4. Devuelve **fragmentFirstLayout** desde **onCreateView()**.

```
return fragmentFirstLayout;
```

Aquí está el método completo y la declaración de showCountTextView:

```
TextView showCountTextView;
@Override
public View onCreateView(
        LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState
) {
    // Inflate the layout for this fragment
    View fragmentFirstLayout = inflater.inflate(R.layout.fragment_first, container, false);
    // Get the count text view
    showCountTextView = fragmentFirstLayout.findViewById(R.id.textview_first);
    return fragmentFirstLayout;
}
```

5. Ejecute su aplicación. Presione el botón Contar y observe cómo se actualiza el conteo.