Summary

We just covered a really powerful concept - *Interfaces* - and saw a specific use case - the *OnClickListener Interface* - which helps us listen for when a View is clicked on and then trigger an action.

```
public interface OnClickListener
void onClick()
```

Interfaces Recap

Interfaces are similar to classes we've used before; however, they have some large differences. Interfaces don't contain states and methods that are fully implemented and ready for us to use. Instead, interfaces, contain NO states and only method signatures (which specifies the name, return value, and input parameters). Thus, interfaces cannot be instantiated; in order to use them, any class that implements the interface must implement the methods (e.g. build out the behavior) declared in that interface.

OnClickListener Interface

In our example in order to use the **OnClickListener Interface**, we'll need to implement the **onClick** method.
Why use an interface in this case?
The Android team knows that many developers will probably want to customize the click behavior, so it wouldn't make sense to provide default behavior for what happens when a button is clicked. Despite this, the Android team did want to standardize what the method call would be and that the class type would be an OnClickListener. Then they leave the logic inside the onClick method for the developer to write themselves.

Further Readings

Interfaces are an important fundamental Java concept. Take some time to solidify the concepts with these resources:

- **Beginners Book - Interface in Java with example programs**
- **Java Tutorials - Interfaces and Inheritance**