● 뷰 이벤트 모델: 이벤트 소스와 이벤트 핸들러를 리스터(Listener)로 연결하여 처리하는 구조.
▶ 이벤트 소스(Event Source): 이벤트가 발생한 뷰 객체
▶ 이벤트 핸들러(Event Handler): 이벤트 처리 내용을 가지는 객체
▶ 리스터(Listener): 이벤트 소스와 이벤트 핸들러를 연결하는 작업

모든 뷰의 이벤트는 터치 이벤트로 처리할 수 있지만, 뷰 이벤트 모델로 처리하는 이유는 이벤트를 조금 더 명료하게 처리하기 위해서이다. 예를 들면 총 5개의 뷰에서 사용자 이벤트가 발생할 때, 모두 터치 이벤트로 처리하면 그 중 어느 뷰가 터치된 것인지 명료하지 않다. 모두 액티비티의 터치 이벤트로 처리하는 대신, 이벤트가 발생하는 뷰를 직접 지칭하여 각 이벤트의 성격별로 이벤트 이름을 다르게 처리하면 훨씬 더 명료해진다. 이처럼 뷰 이벤트 모델은 이벤트가 발생한 객체를 명료하게 지칭하고자 이벤트 소스를 사용하고, 이벤트 성격을 명료하게 지칭하고자 리스너를 사용한다. (ex switch CheckEvent, CheckBox CheckEvent, TextView ClickEvent).

```
vibrateCheckView.setOnCheckedChangeListener(new MyEventHandler());
```

"setOnXXXListener" 이 부분이 이벤트 소스와 이벤트 핸들러를 리스너로 연결하는 부분이다. vbrateCheckView 객체에서 CheckedChangeEvent가 발생하면, MyEventHandler라는 클래스 객체를 실행하여 이벤트를 처리하라는 의미이다. 이때 이벤트 핸들러 클래스는 꼭 지정된 인터페이스를 구현하는 것이어야 한다.

```
class MyEventHandler implemenets CompoundButton.OnCheckedChangeListener {
    @Override
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
    }
}
```

OnCheckedChangeListener 인터페이스를 구현하고, 추상 함수를 재정의(override)하여 이벤트 처리 로직을 작성해 놓으면 실제 이벤트가 발생할 때, 해당 함수가 실행된다. (객체에 이벤트 발생 -> Listener로 등록된 EventHandler의 함수 실행)

```
vibrateCheckView.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener()
    @Override
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
    }
});
```

※ EventHandler 클래스는 '이름 없는, 즉 익명의 하위 클래스(anonymous inner class)'의 형태로 작성한다

```
btn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
    }
});
```

```
btn.setOnClickListener(new MyHandler());

class MyHandler implements View.OnClickListener {
    @Override
    public void onClick(View v) {
    }
}
```

위의 코드는 아래 코드를 축약해서 작성한 것이다. new 다음에 나오는 게 클래스면 클래스를 상속받은 이름 없는 하위 클래스를 선언한 것이고, 인터페이스면 인터페이스를 구현한 이름 없는 클래스를 선언하고, 선언하자마자 객체를 생성한 코드인 것이다. 결국, OnClickListener를 생성한 게 아니라 OnClickListener 인터페이스를 구현한 이름 없는 하위 클래스 (즉, 익명 클래스)를 선언하고 생성해서 리스너로 연결한 코드인 것이다.

● 다양한 이벤트 처리

주요 Event	설명
OnClickListener	뷰 클릭 시 발생하는 이벤트
OnLongClickListener	뷰를 오래 클릭했을 때 발생하는 이벤트
OnCheckedChangeListener	CheckBox의 상태 변경 이벤트
OnDataSetListener	DataPicker의 날짜 선택 이벤트
OnTimeSetListener	TimePicker의 시간 선택 이벤트

이벤트를 처리하는 구조는 모두 같다. 이벤트 소스와 이벤트 핸들러를 setOnXXXListener() 함수로 연결하고, 이벤트 핸들러는 OnXXXListener을 구현해서 작성한다.

▶ OnClickListener와 onLongClickListener 이벤트는 모든 뷰에 적용할 수 있는 이벤트이다. 대표적인 뷰가 Button이며 이 외에 TextView, ImageView 등 모든 뷰에 등록할 수 있다.

```
btn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
    }
});
```

```
btn.setOnLongClickListener(new View.OnLongClickListener() {
    @Override
    public void onLongClick(View v) {
    }
});
```

▶ OnCheckedChangeListener: 체크박스나 라디오 버튼의 체크 상태를 변경할 때 발생하는 이벤트이다.

```
checkBox.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
    }
});
```