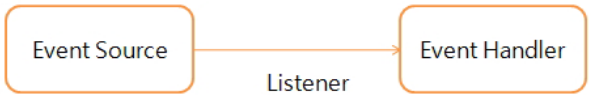
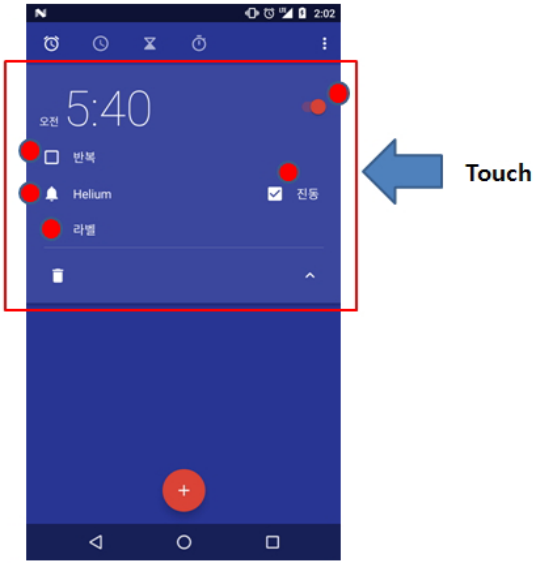
6장. 사용자 이벤트 처리

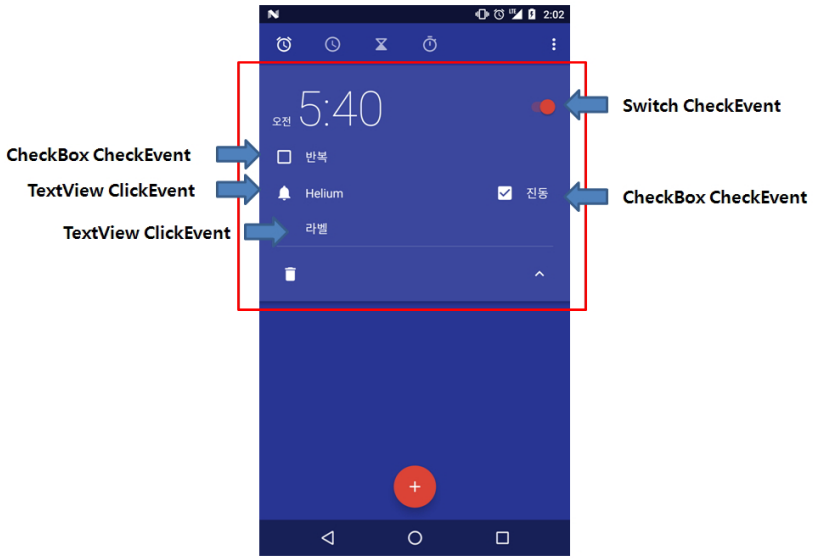
6.1 Delegation Event Model

6.1.1. 이벤트 프로그램 구조



- 이벤트 소스(Event Source): 이벤트가 발생한 뷰 객체
- 이벤트 핸들러(Event Handler): 이벤트 처리 내용을 가지는 객체
- 리스너(Listener): 이벤트 소스와 이벤트 핸들러를 연결하는 작업





6.1 Delegation Event Model


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

```
class MyEventHandler implements CompoundButton.OnCheckedChangeListener{
   @Override
   public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
      // 이벤트 처리 로직 작성
   }
}
```

1. 객체에 Event 발생

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

2. Listener로 등록된 EventHandler 의 함수 실행

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

    }
}
```

6.1 Delegation Event Model

6.1.2. 다양한 이벤트 처리

표 6-1 주요 이벤트

Event	설명
OnClickListener	뷰 클릭 시 발생하는 이벤트
OnLongClickListener	뷰를 오래 클릭했을 때 발생하는 이벤트
OnCheckedChangeListener	CheckBox의 상태 변경 이벤트
OnItemClickListener	ListView의 항목 선택 이벤트
OnDateSetListener	DatePicker의 날짜 선택 이벤트
OnTimeSetListener	TimePicker의 시간 선택 이벤트

OnClickListener

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

    }
});
```

6.1 Delegation Event Model

- OnLongClickListener

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

- OnCheckedChangeListener

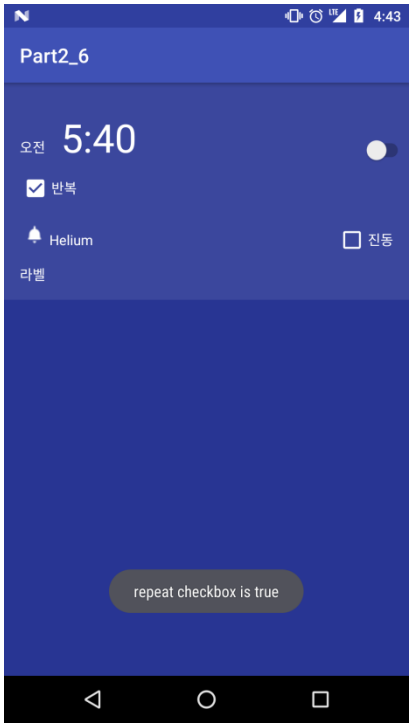
```
checkBox.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {

    }
});
```

Step by Step 6-1 – Delegation Event

•Alarm App의 신규 등록 화면을 가정해서 이벤트를 등록시키는 실습
•이벤트 처리 로직은 단순 Toast 문자열로 이벤트를 확인하는 정도로 처리

1. Module 생성
2. 파일 복사
3. Event Listener interface 선언
4. onClick 함수 구현
5. onCheckedChanged 함수 구현
6. 실행



6.2 Hierarchy Event Model

6.2.1. 터치 이벤트

```
public boolean onTouchEvent(MotionEvent event) {
   return super.onTouchEvent(event);
}
```

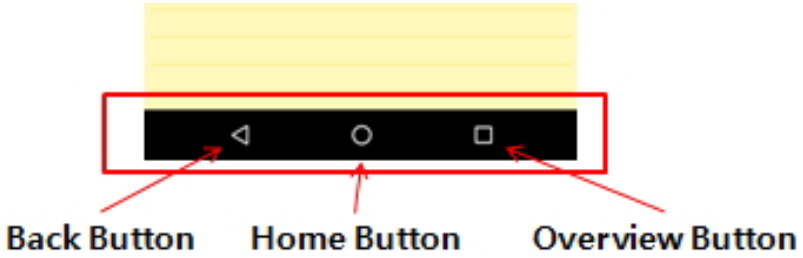
- ACTION_DOWN: 화면에 터치된 순간의 이벤트
- ACTION_UP: 터치를 떼는 순간의 이벤트
- ACTION_MOVE: 터치한 후 이동하는 순간의 이벤트

- getX()
- getY()
- getRawX()
- getRawY()

```
public boolean onTouchEvent(MotionEvent event) {
    if(event.getAction()==MotionEvent.ACTION_DOWN){
        initX=event.getRawX();
    }
    return true;
}
```

6.2 Hierarchy Event Model

6.2.2. 키 이벤트



- onKeyDown: 키가 눌린 순간의 이벤트
- onKeyUp: 키를 떼는 순간의 이벤트
- onKeyLongPress: 키를 오래 누르는 순간의 이벤트

```
public boolean onKeyDown(int keyCode, KeyEvent event) {
    if(keyCode==KeyEvent.KEYCODE_BACK){

    }
    return super.onKeyDown(keyCode, event);
}
```

```
public void onBackPressed() {
    super.onBackPressed();
}
```

Step by Step 6-2 – 터치, 키 이벤트

•App 에서 제공되는 종료하실려면 한번더 누르세요 메시지를 Toast로 뿌려주는 실습을 진행

•화면을 유저가 오른쪽으로 밀었는지 왼쪽으로 밀었는지를 판단해 간단하여 Toast로 출력하는 기능을 추가

•이번 실습은 새로운 Activity를 만들지 않고 6-1 실습시 이용한 MainActivity에 추가해 개발

1. onTouchEvent 함수 추가
2. onKeyDown 함수 추가
3. 결과 실행

화면을 왼쪽으로 민경우

Back Button 누른 경우