컴포즈 스터디

@Composable

- 프로그램이 어떤 방법으로 해야 하는지를 나타내기보다 무엇과 같은지를 설명하는 경우에 `"선언형"`이라고 한다.
- 기존에는 findByViewId() 같은 걸로 복잡하게 접근하여 사고가 날 가능성이 있었음

기존 방식

```
b.setColor(red)
b.clearChildren()
ViewC c3 = new ViewC(...)
b.add(c3)
```

선언형 UI 사용 시

```
return ViewB(
  color: red,
  child: const ViewC(),
);
```

> **명령형 방식** - OO역의 북쪽 출구로 나와 왼쪽으로 5분 가면 ㅁㅁ학교가 나오는데 거기서 XX방면으로 10분동안 직진하면 됩니다.

> **선언형 방식** - 주소는 OO시 ㅁㅁ구 XX로 입니다.

- 이 함수가 데이터를 UI로 변환하기 위한 함수라는 것을 Compose 컴파일러에 알립니다.

Surface

- Surface는 안드로이드 컴포즈에서 화면에 그리는 기본적인 단위
- Surface는 렌더링 가능한 영역을 나타냄
- 내부 컴포넌트에 margin을 부여하는게 아니라 surface에 padding을 부여하는게 성능적으로 더 좋다네요

```
  Surface {
      TopAppBar(
          modifier = modifier
              .statusBarsPadding()
              .background(color = MaterialTheme.colorScheme.surface),
```

```
            title = {
                Row {
                    IconButton(
                        onBackClick,
                        Modifier.align(Alignment.CenterVertically)
                    ) {
                        Icon(
                            Icons.AutoMirrored.Filled.ArrowBack,
                            contentDescription = stringResource(id = R.string.a
                        )
                    }
                    Text(
                        text = plantName,
                        style = MaterialTheme.typography.titleLarge,
                        // As title in TopAppBar has extra inset on the left, n
                        modifier = Modifier
                            .weight(1f)
                            .fillMaxSize()
                            .wrapContentSize(Alignment.Center)
                    )
                    val shareContentDescription =
                        stringResource(R.string.menu_item_share_plant)
                    IconButton(
                        onShareClick,
                        Modifier
                            .align(Alignment.CenterVertically)
                            // Semantics in parent due to https://issuetracker.
                            .semantics { contentDescription = shareContentDescr
                    ) {
                        Icon(
                            Icons.Filled.Share,
                            contentDescription = null
                        )
                    }
                }
            }
        )
    }
```

Slot Api

- 컴포넌트에 특정 자리를 비워 두고 사용자 정의를 할 수 있게 함

```
@Composable
fun CustomSlot(content: @Composable () -> Unit) {
    Card(modifier = Modifier.padding(16.dp)) {
        Box(modifier = Modifier.padding(16.dp)) {
```

```
            content() // 여기에 배치된다
        }
    }
}

@Composable
fun TestSlot() {
    CustomSlot {
        Text("Hello!")
    }
}
```

- 컴포저블 람다(`content: @Composable () -> Unit`)를 사용
- React에서의 prop과 유사한 부분이나, UI 단위로 집어넣을 수 있다는 것이 차이로 보임