Mantener el estado en una función componible y elevar evento

En algunas situaciones una función componible puede mantener el estado sin requerir que otra función anterior mantenga dicho estado.

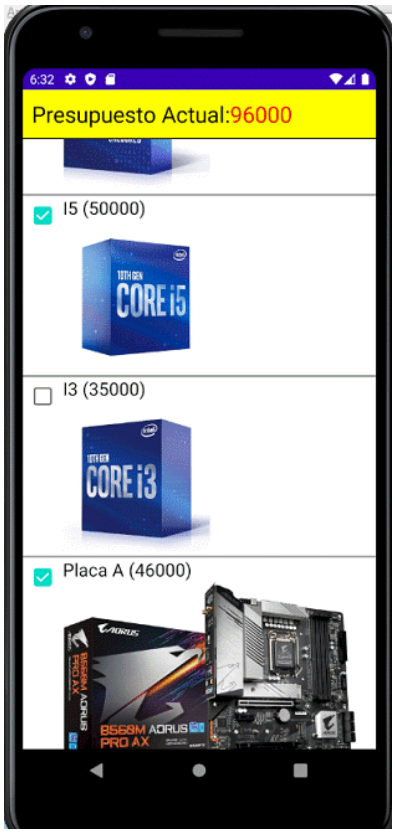
Problema

Desarrollar una aplicación que me permita generar el presupuesto de un equipo de computación. Mostrar una lista de componentes de computadora y mediante un Checkbox permitir su selección o deselección.

Implementar dos funciones componibles, la primera administra el estado del costo total según los componentes seleccionados, por otro lado cada componente debe administrar el estado si se encuentra seleccionado o no mediante un Checkbox.

Informar cada vez que se selecciona un componente a la función que calcula el presupuesto.

Crear el proyecto 'Compose9', el resultado final debe tener una apariencia similar a:



Primero no olvidar de agregar las imágenes de los componentes de computadoras en la carpeta drawable.

El código fuente es:

```
package com.tesji.compose9

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.material.Checkbox
import androidx.compose.material.Divider
import androidx.compose.material.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.*
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.saveable.rememberSaveable
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Presupuesto(productos)
        }
    }
}

@Composable
fun Presupuesto(productos: List<Producto>) {
    var monto by rememberSaveable { mutableStateOf(0) }
    Column() {
        Row(modifier = Modifier
            .fillMaxWidth()
            .background(Color.Yellow)
            .padding(10.dp)) {
            Text(text = "Presupuesto Actual:", fontSize = 25.sp)
            Text(text = "$monto", fontSize = 25.sp, color = Color.Red)
        }
        Divider(
            color = Color.Black,
            modifier = Modifier
                .fillMaxWidth()
                .width(5.dp)
        )
        LazyColumn() {
            items(productos) { producto ->
                ListadoProducto(producto, seleccionProducto = {
```

```
                    monto = monto + it
                })
            }
        }
    }
}

@Composable
fun ListadoProducto(producto: Producto, seleccionProducto: (Int) -> Unit) {
    var seleccionado by rememberSaveable { mutableStateOf(false) }
    Row() {
        Checkbox(
            checked = seleccionado,
            onCheckedChange = {
                seleccionado = it
                if (it)
                    seleccionProducto(producto.precio)
                else
                    seleccionProducto(-producto.precio)
            },
            modifier = Modifier.padding(10.dp),
        )
        Column() {
            Text(text = "${producto.nombre} (${producto.precio})", fontSize = 20.sp)
            Image(painter = painterResource(id = producto.imagen), contentDescription = null)
        }
    }
    Divider(
        color = Color.Black,
        modifier = Modifier
            .fillMaxWidth()
            .width(1.dp)
    )
}

data class Producto(val nombre: String, val precio: Int, val imagen: Int)

val productos = listOf<Producto>(
    Producto("I9", 70000, R.drawable.i7),
    Producto("I5", 50000, R.drawable.i5),
    Producto("I3", 35000, R.drawable.i3),
    Producto("Placa A", 46000, R.drawable.placa1),
    Producto("Placa B", 25000, R.drawable.placa2),
    Producto("Disco A", 15000, R.drawable.disco1),
    Producto("Disco B", 12000, R.drawable.disco2),
    Producto("Gabinete 1", 19000, R.drawable.gabinete1),
    Producto("Gabinete 2", 25000, R.drawable.gabinete2)
)
```

En la parte inferior del archivo hemos definido el data class Producto y una constante con una lista de 9 productos:

Para el problema hemos definido dos funciones, la primera define una variable llamada 'monto' que nos sirve para almacenar el presupuesto de todas las componentes del pc seleccionado, utilizamos la función rememberSaveable en lugar de remember para que el valor no se pierda si cambiamos la dirección del celular de horizontal a vertical o viceversa.

Disponemos una columna y una primer fila (Row) donde se muestra el presupuesto actual almacenado en la variable 'monto'.

El monto se actualiza cada vez que en el operador marca o desmarca un Checkbox creado en la otra función (gracias a la función lambda que le enviamos en el parámetro 'seleccionProducto'):

```
@Composable
fun Presupuesto(productos: List<Producto>) {
    var monto by rememberSaveable { mutableStateOf(0) }
    Column() {
        Row(modifier = Modifier
            .fillMaxWidth()
            .background(Color.Yellow)
            .padding(10.dp)) {
            Text(text = "Presupuesto Actual:", fontSize = 25.sp)
            Text(text = "$monto", fontSize = 25.sp, color = Color.Red)
        }
        Divider(
            color = Color.Black,
            modifier = Modifier
                .fillMaxWidth()
                .width(5.dp)
        )
        LazyColumn() {
            items(productos) { producto ->
                ListadoProducto(producto, seleccionProducto = {
                    monto = monto + it
                })
            }
        }
    }
}
```

La segundo función muestra cada producto que llega como parámetro y almacena el estado del Checkbox, por defecto false o desmarcado.

Para el correcto funcionamiento del Checkbox debemos pasar al parámetro 'checked' la variable que almacena el estado y al parámetro onCheckedChange una función lambda que actualiza el estado de la variable 'seleccionado' y dispara el evento a ser capturado por la otra función para que actualize el presupuesto (si está seleccionado el Checkbox pasa el precio del producto, en caso contrario pasa el valor negativo del producto para que lo descuente):

```
@Composable
fun ListadoProducto(producto: Producto, seleccionProducto: (Int) -> Unit) {
    var seleccionado by rememberSaveable { mutableStateOf(false) }
    Row() {
        Checkbox(
            checked = seleccionado,
            onCheckedChange = {
                seleccionado = it
                if (it)
                    seleccionProducto(producto.precio)
                else
                    seleccionProducto(-producto.precio)
            },
            modifier = Modifier.padding(10.dp),
        )
        Column() {
            Text(text = "${producto.nombre} (${producto.precio})", fontSize = 20.sp)
            Image(painter = painterResource(id = producto.imagen), contentDescription = null)
        }
    }
    Divider(
        color = Color.Black,
        modifier = Modifier
            .fillMaxWidth()
            .width(1.dp)
    )
}
```

Acotaciones

En el ejemplo anterior para seleccionar un producto debemos hacer clic solo en el Checkbox, si queremos que se puede seleccionar o deseleccionar haciendo click en el texto o la imágen, es decir en cualquier parte de la fila, solo debemos hacer este cambio:

```
@Composable
fun ListadoProducto(producto: Producto, seleccionProducto: (Int) -> Unit) {
    var seleccionado by rememberSaveable { mutableStateOf(false) }
    Row(modifier = Modifier.clickable {
        seleccionado=!seleccionado
        if (seleccionado)
            seleccionProducto(producto.precio)
        else
            seleccionProducto(-producto.precio)
    }) {
        Checkbox(
            checked = seleccionado,
            onCheckedChange = {
                seleccionado = it
                if (it)
                    seleccionProducto(producto.precio)
                else
                    seleccionProducto(-producto.precio)
            },
            modifier = Modifier.padding(10.dp),
        )
        Column() {
            Text(text = "${producto.nombre} (${producto.precio})", fontSize = 20.sp)
            Image(painter = painterResource(id = producto.imagen), contentDescription = null)
        }
    }
    Divider(
        color = Color.Black,
        modifier = Modifier
            .fillMaxWidth()
            .width(1.dp)
    )
}
```

Cuando se haga clic en cualquier parte de la fila se procede a cambiar el valor de la variable 'seleccionado', esto hace que cambie el estado del Checkbox y se dispare el evento de cambio de selección (onCheckedChange)