```
//William Murray
//017540586
//CECS 453
package com.example.digitalartspace

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.foundation.Image
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.material3.Button
import androidx.compose.runtime.MutableState
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.text.SpanStyle
import androidx.compose.ui.text.buildAnnotatedString
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.style.TextOverflow
import androidx.compose.ui.text.withStyle
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.unit.dp
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.TextUnit
import androidx.compose.ui.unit.TextUnitType

import com.example.digitalartspace.ui.theme.DigitalArtSpaceTheme
import com.example.digitalartspace.models.Artwork
import com.example.digitalartspace.data.Database


class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            DigitalArtSpaceTheme {
                // A surface container using the 'background' color
from the theme
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
                ) {
                    val database = Database()
                    val artworks = database.loadArtwork()

                    // State for keeping track of the current artwork
index
                    var artworkIndex : MutableState<Int> = remember {
                        mutableStateOf(0)
                    }

                    // Get the current artwork from the list
                    var currentArtwork: MutableState<Artwork> =
remember {
                      mutableStateOf(artworks[artworkIndex.value])
                    }

                    ArtworkViewer(
                        artworkName =
getString(currentArtwork.value.nameResId),
                        artworkYear =
resources.getInteger(currentArtwork.value.yearResId),
                        artworkCreator =
getString(currentArtwork.value.creatorResId),
                        artwork = currentArtwork.value.imageResId,
                        artworkDesc =
getString(currentArtwork.value.descriptionResId),
                        onPrevious = {
                            // Update the index for the previous
artwork, cycling back if necessary
                            artworkIndex.value = (artworkIndex.value -
1 + artworks.size) % artworks.size
                            currentArtwork.value =
artworks[artworkIndex.value]
                            println("hello")
                        },
                        onNext = {
                            // Update the index for the next artwork,
cycling back if necessary
                            artworkIndex.value = (artworkIndex.value +
1) % artworks.size
                            currentArtwork.value =
artworks[artworkIndex.value]
                        },
                        modifier = Modifier.padding(horizontal =
15.dp))

                }
            }
        }
    }
```

```
}
@Composable
fun ArtworkViewer(artworkName: String, artworkYear: Int,
artworkCreator: String,
                  artwork: Int, artworkDesc: String,  onPrevious: ()
-> Unit, // Callback for when "Previous" is clicked
                  onNext: () -> Unit, modifier: Modifier = Modifier) {


    Column(
        modifier = modifier.fillMaxSize(),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Bottom
    ) {
        Surface (
            shadowElevation = 10.dp,
            color = Color.White
        ) {
            Box(modifier = Modifier.padding(30.dp).fillMaxHeight(.6f)
                .fillMaxWidth(),
                contentAlignment = Alignment.Center) {


                Image(

                    painter = painterResource(id = artwork),
                    contentDescription = "Still Life of Blue Rose and
Other Flowers"
                )
            }
        }

        Spacer(modifier = Modifier.height(48.dp))

        ArtworkDescriptor(
            artworkName = artworkName,
            artworkCreator = artworkCreator,
            artworkYear = artworkYear,
            modifier = modifier
        )

        Spacer(modifier = Modifier.height(32.dp))

        DisplayController(onNext = onNext, onPrevious =
onPrevious ,modifier = modifier)

    }
}

@Composable
```

```
fun ArtworkDescriptor(artworkName: String, artworkCreator: String,
artworkYear: Int, modifier: Modifier = Modifier) {
    val combinedText = buildAnnotatedString {
        withStyle(style = SpanStyle(fontWeight = FontWeight.Bold)) {
            append("$artworkCreator ")
        }
        append("($artworkYear)")

        // Apply a bold style to the following text

    }
    Box (
        modifier = modifier
            .background(color = Color(0xFFADD8E6))
            .padding(15.dp)
            .fillMaxWidth(1f)
    ) {
        Column (
            horizontalAlignment = Alignment.Start
        ) {
            Text(
                text = artworkName,
                fontSize = TextUnit(22.0f, TextUnitType.Sp),
                textAlign = TextAlign.Start,
                overflow = TextOverflow.Ellipsis
            )
            Text(
                text = combinedText,
                textAlign = TextAlign.Center
            )
        }
    }



}

@Composable
fun DisplayController(    onPrevious: () -> Unit,
                          onNext: () -> Unit,
                          modifier: Modifier = Modifier) {
    Row (
        verticalAlignment = Alignment.CenterVertically,
        modifier = modifier.padding(bottom = 15.dp)
    ) {
        Button(
            onClick = onPrevious,
            modifier = modifier
                .width(120.dp)
                .height(40.dp),
            ) {
            Text("Previous")
        }
        Spacer(modifier = Modifier.weight(1f))
        Button(
            onClick = onNext,
            modifier = modifier
                .width(120.dp)
                .height(40.dp)
            ) {
            Text("Next")
        }
    }
}

//@Composable
//fun Greeting(name: String, modifier: Modifier = Modifier) {
//    Text(
//        text = "Hello $name!",
//        modifier = modifier
//    )
//}
//
//@Preview(showBackground = true)
//@Composable
//fun GreetingPreview() {
//    DigitalArtSpaceTheme {
//        Greeting("Android")
//    }
//}
```