Nama: Krishna Ksatria Putra

1. Apa perbedaan activity dengan fragment? (5)

Penggunaan Utama:

- Activity: Activity adalah komponen yang mewakili satu layar dengan antarmuka pengguna. Setiap layar dalam aplikasi Android direpresentasikan oleh satu Activity. Misalnya, tampilan login atau tampilan utama adalah contoh Activity.
- Fragment: Fragment adalah bagian dari antarmuka pengguna di dalam Activity. Fragment memungkinkan bagi tampilan dan perilaku suatu bagian dari antarmuka pengguna untuk dipecah menjadi bagian-bagian terpisah dan dapat digunakan kembali di berbagai Activity.

Pengelolaan UI:

- Activity: Activity memiliki tata letak sendiri dan mengelola UI secara utuh untuk satu layar.
- Fragment: Fragment dapat memiliki tata letak sendiri yang merupakan bagian dari UI Activity. Sebuah Activity dapat mengandung beberapa Fragment yang saling berinteraksi dalam satu antarmuka.

Kemampuan Reusabilitas:

- Activity: Meskipun Activity dapat digunakan kembali, namun tidak sefleksibel Fragment dalam hal penggunaan kembali komponen UI.
- Fragment: Fragment didesain untuk dapat digunakan kembali di berbagai Activity, sehingga memungkinkan reusabilitas dan modularitas yang lebih tinggi.

Siklus Hidup:

- Activity: Memiliki siklus hidupnya sendiri yang dimulai dari onCreate(), onStart(), onResume(), onPause(), onStop(), dan akhirnya onDestroy() saat Activity dihancurkan.
- Fragment: Memiliki siklus hidup yang mirip dengan Activity, yaitu onAttach(), onCreate(), onCreateView(), onActivityCreated(), onStart(), onResume(), onPause(), onStop(), onDestroyView(), dan onDetach().

Komunikasi Antarkomponen:

- Activity: Activity dapat berkomunikasi dengan Activity lainnya melalui intent dan bundel untuk mentransfer data.
- Fragment: Fragment dapat berkomunikasi dengan Activity yang mengandungnya dan dengan Fragment lain di dalam Activity tersebut melalui antarmuka dan metode yang ditentukan.

Dalam pengembangan aplikasi Android yang baik, kombinasi penggunaan Activity dan Fragment dengan bijak akan membantu membangun aplikasi yang modular, efisien, dan mudah dielola.

Berikan contoh use casenya?(10)

Contohnya seperti saat kita membuat onboarding atau menu pada home page, untuk slide onboarding atau mengubah page saat menu di tekan pada home,

kita memecah beberapa page menjadi fragment.

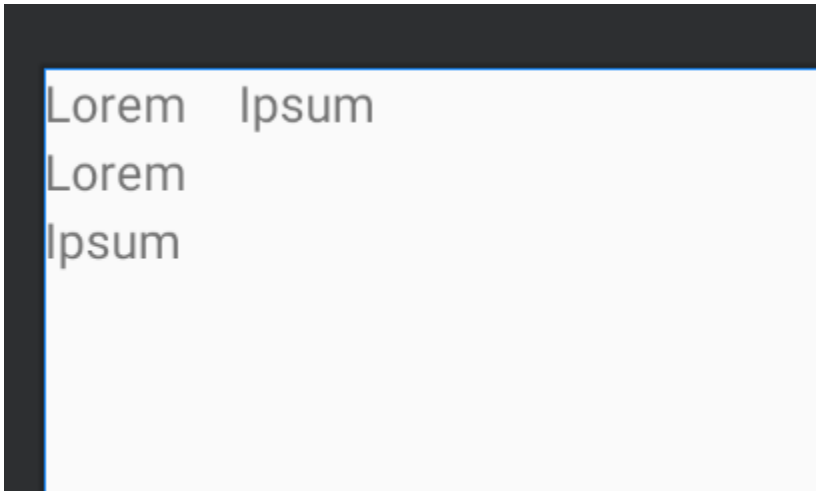
2. Apa perbedaan service dengan broadcast receiver?(5)
   - **Service:** Service adalah komponen yang digunakan untuk melakukan operasi latar belakang tanpa memerlukan interaksi langsung dengan pengguna. Contoh penggunaannya adalah pemutaran musik, pengunduhan file, pemrosesan data, dll.
   - **Broadcast Receiver:** Broadcast Receiver adalah komponen yang digunakan untuk menerima dan menanggapi pesan sistem atau pesan yang dikirim oleh aplikasi lain atau sistem, seperti notifikasi baterai rendah, SMS masuk, atau sinyal Wi-Fi yang berubah.

3. Bagaimana cara membawa informasi dari satu activity ke activity lainnya?(5)

   Jawaban: Menggunakan Intent untuk berpindah activity dan menggunakan put extra untuk membawa informasi ke activity selanjutnya.

   - Berikan contoh jika payload yang dibawa “isCard” dengan value true(5)
     ```
     val intent = Intent(this, NextActivity::class.java)
     intent.putExtra("isCard", true)
     startActivity(intent)
     ```
   - Berikan contoh jika payload yang dibawa adalah model yang mempunyai attribute id String, isValue Boolean, optionalJson JsonObject(10)
     ```
     data class TestingModel(
         val id: String?,
         val isValue: Boolean?,
         val optionalJson: JsonObject?
     )
     val model = TestingModel()
     val intent = Intent(this, NextActivity::class.java)
     intent.putExtra("testingModel", model)
     startActivity(intent)
     ```

4. Bagaimana cara agar setiap textview menjadi seperti gambar dibawah ini?(10)




```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="_____"> (1)
        <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:orientation="_____"> (2)
                <TextView
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:layout_marginEnd="15dp"
                        android:text="Lorem"/>
                <TextView
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:text="Ipsum"/>
        </LinearLayout>
        <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:orientation="_____"> (3)
                <TextView
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:layout_marginEnd="15dp"
                        android:text="Lorem"/>

                <TextView
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:text="Ipsum"/>
```

```
        </LinearLayout>
</LinearLayout>
```

Jawaban:
1. Vertical
2. Horizontal
3. Vertical

Merchant	
Id	Int
Name	String
Address	String

5. Lihat table diatas
   - Bagaimana cara create table diatas?(10)

```
data class Merchant(
        val id: Int = “”,
        val name: String = “”,
        val address: String = “”
) {
        fun getId(): Int = Id
        fun getName(): String = name
        fun getAddress(): String = address

        fun setId(newId: Int) { id = newId }
        fun setName(newname: String) { name = newName }
        fun setAddress(newAddress: String) { address = newAddress }
}
```

   - Bagaimana cara read dari table tersebut?(5)

```
val merchant: Merchant = Merchant()
merchant.getId()
```

   - Bagaimana cara insert ke table tersebut?(5)

```
val merchant: Merchant = Merchant()
merchant.setId(“testingId”)
```

   - Bagaimana cara update suatu row dengan kondisi nama = “PCS” dan address yang 3 kata yang diawali “PCS”(5)

6. Apa perbedaan sqlite vs sharedpreference?(5)

   Tipe Data:

- **SQLite:** SQLite adalah sistem manajemen basis data relasional (RDBMS) yang digunakan untuk menyimpan dan mengelola data terstruktur dalam bentuk tabel dengan skema yang telah ditentukan.
- **SharedPreferences:** SharedPreferences adalah penyimpanan data sederhana yang digunakan untuk menyimpan data primitif (seperti string, integer, boolean) dalam bentuk pasangan kunci-nilai.

Penggunaan:

- **SQLite:** Digunakan untuk menyimpan dan mengelola data yang kompleks, terstruktur, dan memerlukan kueri SQL untuk mengambil atau memodifikasi data.
- **SharedPreferences:** Digunakan untuk menyimpan data kecil, sederhana, dan primitif seperti pengaturan aplikasi, preferensi pengguna, atau informasi sesi.

Kinerja:

- **SQLite:** Lebih cocok untuk operasi yang melibatkan jumlah data besar atau kompleks. Memiliki kinerja yang baik untuk query yang kompleks dan transaksi.
- **SharedPreferences:** Kinerjanya baik untuk data kecil dan operasi sederhana, tetapi tidak disarankan untuk menyimpan data yang besar atau kompleks.

Query dan Pencarian:

- **SQLite:** Memungkinkan query yang kompleks menggunakan bahasa SQL untuk melakukan pencarian, pengurutan, dan manipulasi data dengan fleksibilitas tinggi.
- **SharedPreferences:** Tidak memungkinkan query atau pencarian data secara langsung. Data diakses berdasarkan kunci yang telah ditetapkan sebelumnya.

Security dan Keamanan:

- **SQLite:** Dapat menyediakan tingkat keamanan tambahan karena dapat mengimplementasikan kontrol akses dan enkripsi data di dalam database.
- **SharedPreferences:** Tidak disarankan untuk menyimpan data sensitif karena tidak menyediakan tingkat keamanan yang tinggi dan dapat diakses dengan mudah jika perangkat sudah di-root atau dalam keadaan tidak aman.

Jangkauan Data:

- **SQLite:** Data di SQLite bersifat persisten, artinya data akan tetap ada bahkan setelah aplikasi ditutup atau perangkat di-restart.
- **SharedPreferences:** Data SharedPreferences hanya bersifat per-session dan akan terhapus jika aplikasi ditutup atau perangkat di-restart.

Pengelolaan Data:

- **SQLite:** Memerlukan logika untuk membuat, mengelola, dan memperbarui struktur tabel serta menjalankan query SQL.
- **SharedPreferences:** Mudah dikelola dan diakses dengan API SharedPreferences bawaan Android.

Kesimpulannya, SQLite cocok untuk penyimpanan data yang kompleks, terstruktur, dan membutuhkan query SQL, sedangkan SharedPreferences lebih cocok untuk penyimpanan data sederhana, primitif, dan pengaturan aplikasi. Pilihan antara keduanya tergantung pada kebutuhan aplikasi dan jenis data yang ingin disimpan.

Tuliskan contoh read dan write di shared preference(10)

val preferences = applicationContext().getSharedPreferences("",
Context.*MODE_PRIVATE*)
val editor = preferences.edit()

Read:
preferences.getString(“token”, ””)

Write:
editor.putString(“token”, token)
editor.commit()

7. Buatlah function untuk mengirim data ke server dengan payload username dan password(15)

8. [a,b,c,d], [f,g,a,b], [a,b,f,g]
   dari array tersebut hitung jumlah setiap huruf muncul contoh a muncul 3 kali b muncul 3kali dst(10)

   Jawaban:
   a muncul 3 kali
   b muncul 3 kali
   c muncul 1 kali
   d muncul 1 kali
   f muncul 2 kali
   g muncul 2 kali

9. [1,2,3,4]
   [5,6,7,8]
   [9,10,11,12]
   Buatlah program yang mempunyai output : 1,2,3,4,8,12,11,10,9,5,6,7 (10)

```
fun main() {
   val array1 = intArrayOf(1, 2, 3, 4)
   val array2 = intArrayOf(5, 6, 7, 8)
   val array3 = intArrayOf(9, 10, 11, 12)
```

```
    val combinedArray = combineArrays(array1, array2, array3)

    println("Output: ${combinedArray.joinToString(", ")}")
}

fun combineArrays(vararg arrays: IntArray): IntArray {
    val combinedArray = mutableListOf<Int>()

    // Add array1
    combinedArray.addAll(arrays[0].toList())

   // Add first index of array2
   combinedArray.add(arrays[1].last())

    // Add array3 in reverse order
    combinedArray.addAll(arrays[2].asList().reversed())

    // Add array3 in reverse order
    combinedArray.addAll(arrays[1].asList())

    // Remove last index
    combinedArray.removeAt(combinedArray.lastIndex)

    return combinedArray.toIntArray()
}
```