**Билет 1**

1. **Типы данных, используемые в SQLite:**
   * **INTEGER** — целые числа.
   * **REAL** — числа с плавающей точкой.
   * **TEXT** — текстовые строки.
   * **BLOB** — бинарные данные.
   * **NULL** — значение NULL.
2. **Реализация Spinner:** Spinner в Android используется для отображения выпадающего списка, из которого пользователь может выбрать один вариант.

val spinner: Spinner = findViewById(R.id.spinner)

val items = listOf("Item 1", "Item 2", "Item 3")

val adapter = ArrayAdapter(this, android.R.layout.simple\_spinner\_item, items)

adapter.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item)

spinner.adapter = adapter

1. **Пример открытия базы данных SQLite в Android-приложении:**

val dbHelper = object : SQLiteOpenHelper(this, "MyDatabase.db", null, 1) {

override fun onCreate(db: SQLiteDatabase) {

db.execSQL("CREATE TABLE myTable (id INTEGER PRIMARY KEY, name TEXT)")

}

override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {}

}

val db = dbHelper.writableDatabase

**Билет 2**

1. **Операции с данными таблицы:**
   * **INSERT** — добавление новых данных.
   * **SELECT** — выборка данных.
   * **UPDATE** — обновление данных.
   * **DELETE** — удаление данных.
2. **Создание собственного контент-провайдера:** Контент-провайдер позволяет предоставлять данные из приложения другим приложениям.

class MyContentProvider : ContentProvider() {

override fun onCreate(): Boolean { return true }

override fun query(...) = null

override fun insert(...) = null

override fun update(...) = 0

override fun delete(...) = 0

override fun getType(...) = null

}

1. **Пример работы со списками:**

val listView: ListView = findViewById(R.id.listView)

val items = listOf("Item 1", "Item 2", "Item 3")

val adapter = ArrayAdapter(this, android.R.layout.simple\_list\_item\_1, items)

listView.adapter = adapter

### Билет 3

1. **Понятие LogCat:** LogCat — это инструмент Android Studio для просмотра журналов сообщений, выводимых системой и приложениями. С его помощью можно отслеживать ошибки и отладочные сообщения.
2. **Принцип работы с контактами:** Для работы с контактами в Android используется ContentProvider и URI для доступа к данным.

val contactsCursor = contentResolver.query(ContactsContract.Contacts.CONTENT\_URI, null, null, null, null)

if (contactsCursor != null && contactsCursor.moveToFirst()) {

val nameIndex = contactsCursor.getColumnIndex(ContactsContract.Contacts.DISPLAY\_NAME)

val contactName = contactsCursor.getString(nameIndex)

}

1. **Пример выполнения запроса к базе данных SQLite в Android-приложении:**

val cursor = db.rawQuery("SELECT \* FROM myTable WHERE id = ?", arrayOf("1"))

if (cursor.moveToFirst()) {

val name = cursor.getString(cursor.getColumnIndex("name"))

}

cursor.close()

### Билет 4

1. **Категории сообщений LogCat:**
   * **Verbose** — подробные сообщения.
   * **Debug** — отладочные сообщения.
   * **Info** — информационные сообщения.
   * **Warn** — предупреждения.
   * **Error** — ошибки.
   * **Assert** — критические ошибки.
2. **Механизм работы с файлами:** Работа с файлами в Android осуществляется через FileInputStream и FileOutputStream.

val file = File(filesDir, "myfile.txt")

file.writeText("Hello, world!")

val content = file.readText()

1. **Реализация собственного адаптера для списка:**

class CustomAdapter(context: Context, private val items: List<String>) : ArrayAdapter<String>(context, 0, items) {

override fun getView(position: Int, convertView: View?, parent: ViewGroup): View {

val view = convertView ?: LayoutInflater.from(context).inflate(android.R.layout.simple\_list\_item\_1, parent, false)

val item = items[position]

view.findViewById<TextView>(android.R.id.text1).text = item

return view

}

}

### Билет 5

1. **Понятие ArrayAdapter:** ArrayAdapter — это адаптер для отображения списков, который работает с массивами данных, преобразуя их в видимые элементы списка.
2. **Механизм запуска компонентов другого приложения:** Android поддерживает запуск Activity другого приложения через Intent.

val intent = Intent(Intent.ACTION\_VIEW, Uri.parse("http://www.example.com"))

startActivity(intent)

1. **Пример использования источника данных MediaStore:**

val cursor = contentResolver.query(MediaStore.Images.Media.EXTERNAL\_CONTENT\_URI, null, null, null, null)

while (cursor?.moveToNext() == true) {

val imagePath = cursor.getString(cursor.getColumnIndex(MediaStore.Images.Media.DATA))

}

cursor?.close()

**Билет 6**

1. **Понятие компонента в Android приложении:** Компоненты Android — это основные строительные блоки приложения:
   * **Activity** — экраны приложения.
   * **Service** — выполняет длительные операции в фоновом режиме.
   * **Broadcast Receiver** — позволяет приложению реагировать на системные события.
   * **Content Provider** — предоставляет данные одного приложения другим приложениям.
2. **Роль компонента ExpandableListView:** ExpandableListView — это виджет для отображения списка элементов, каждый из которых можно развернуть для показа подэлементов.

val expandableListView: ExpandableListView = findViewById(R.id.expandableListView)

val parentItems = listOf("Category 1", "Category 2")

val childItems = mapOf("Category 1" to listOf("Item 1", "Item 2"), "Category 2" to listOf("Item 3"))

val adapter = ExpandableListAdapter(this, parentItems, childItems)

expandableListView.setAdapter(adapter)

1. **Основные операции с контактами:**

* **Получение списка контактов:**

val cursor = contentResolver.query(ContactsContract.Contacts.CONTENT\_URI, null, null, null, null)

* **Добавление контакта:** Используется ContentResolver и ContentProviderOperation.
* **Удаление контакта:**

contentResolver.delete(ContactsContract.Contacts.CONTENT\_URI, "DISPLAY\_NAME=?", arrayOf("Contact Name"))

### Билет 7

1. **Понятие ContentProvider:** ContentProvider — компонент, обеспечивающий общий интерфейс для доступа к данным одного приложения из других приложений. Он работает через URI и разрешения.
2. **Реализация ExpandableListView:** Для реализации ExpandableListView нужно создать адаптер:

class ExpandableListAdapter(

private val context: Context,

private val parentItems: List<String>,

private val childItems: Map<String, List<String>>

) : BaseExpandableListAdapter() {

override fun getChild(groupPosition: Int, childPosition: Int) = childItems[parentItems[groupPosition]]!![childPosition]

override fun getGroup(groupPosition: Int) = parentItems[groupPosition]

override fun getGroupCount() = parentItems.size

override fun getChildrenCount(groupPosition: Int) = childItems[parentItems[groupPosition]]!!.size

override fun getGroupView(...) = // Настроить вид для родительского элемента

override fun getChildView(...) = // Настроить вид для дочернего элемента

override fun hasStableIds() = true

override fun isChildSelectable(groupPosition: Int, childPosition: Int) = true

}

1. **Пример создания базы данных SQLite:**

val dbHelper = object : SQLiteOpenHelper(this, "ExampleDB", null, 1) {

override fun onCreate(db: SQLiteDatabase) {

db.execSQL("CREATE TABLE Contacts (id INTEGER PRIMARY KEY, name TEXT)")

}

override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {}

}

val db = dbHelper.writableDatabase

### Билет 8

1. **Определение URL:** URL (Uniform Resource Locator) — это адрес ресурса в сети, включающий протокол, доменное имя и путь к ресурсу.
2. **Роль компонента Spinner:** Spinner — это виджет, который отображает выпадающий список с возможностью выбора одного элемента.

val spinner: Spinner = findViewById(R.id.spinner)

val items = listOf("Option 1", "Option 2")

val adapter = ArrayAdapter(this, android.R.layout.simple\_spinner\_item, items)

spinner.adapter = adapter

1. **Пример открытия базы данных SQLite:**

val dbHelper = SQLiteOpenHelper(this, "ExampleDB", null, 1)

val db = dbHelper.readableDatabase

**Билет 9**

1. **Методы для класса, унаследованного от ContentProvider:**
   * **onCreate()** — инициализация провайдера.
   * **query()** — запрос данных.
   * **insert()** — добавление данных.
   * **update()** — обновление данных.
   * **delete()** — удаление данных.
   * **getType()** — тип данных, возвращаемых query.
2. **Реализация Spinner:**

val spinner: Spinner = findViewById(R.id.spinner)

val options = listOf("Item A", "Item B", "Item C")

val adapter = ArrayAdapter(this, android.R.layout.simple\_spinner\_item, options)

spinner.adapter = adapter

1. **Пример работы со списками:**

val listView: ListView = findViewById(R.id.listView)

val items = listOf("Apple", "Banana", "Cherry")

val adapter = ArrayAdapter(this, android.R.layout.simple\_list\_item\_1, items)

listView.adapter = adapter

**Билет 10**

1. **Примеры встроенных контент-провайдеров:**
   * **ContactsContract** — провайдер для доступа к контактам.
   * **MediaStore** — провайдер для доступа к мультимедиа-файлам.
   * **CalendarContract** — провайдер для доступа к событиям календаря.
2. **Роль компонента GridView:** GridView отображает элементы в виде сетки. Часто используется для галерей изображений.

val gridView: GridView = findViewById(R.id.gridView)

val items = listOf("A", "B", "C", "D")

val adapter = ArrayAdapter(this, android.R.layout.simple\_list\_item\_1, items)

gridView.adapter = adapter

1. **Пример выполнения запроса к базе данных SQLite:**

val cursor = db.rawQuery("SELECT \* FROM Contacts WHERE name = ?", arrayOf("John"))

if (cursor.moveToFirst()) {

val name = cursor.getString(cursor.getColumnIndex("name"))

}

cursor.close()

**Билет 11**

1. **Типы данных, используемые в SQLite:**
   * **INTEGER** — целые числа.
   * **REAL** — числа с плавающей точкой.
   * **TEXT** — текстовые строки.
   * **BLOB** — бинарные данные.
   * **NULL** — значение NULL.
2. **Реализация Spinner:** Spinner позволяет пользователю выбирать один элемент из списка.

val spinner: Spinner = findViewById(R.id.spinner)

val items = listOf("Option 1", "Option 2", "Option 3")

val adapter = ArrayAdapter(this, android.R.layout.simple\_spinner\_item, items)

adapter.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item)

spinner.adapter = adapter

1. **Реализация собственного адаптера для списка:**

class CustomAdapter(context: Context, private val items: List<String>) : ArrayAdapter<String>(context, 0, items) {

override fun getView(position: Int, convertView: View?, parent: ViewGroup): View {

val view = convertView ?: LayoutInflater.from(context).inflate(android.R.layout.simple\_list\_item\_1, parent, false)

view.findViewById<TextView>(android.R.id.text1).text = items[position]

return view

}

}

**Билет 12**

1. **Операции с данными таблицы:**
   * **INSERT** — добавление новых записей.
   * **SELECT** — выборка данных.
   * **UPDATE** — обновление данных.
   * **DELETE** — удаление данных.
2. **Механизм создания собственного контент-провайдера:** Для создания ContentProvider нужно унаследовать его и реализовать основные методы.

class MyContentProvider : ContentProvider() {

override fun onCreate(): Boolean { return true }

override fun query(...) = null

override fun insert(...) = null

override fun update(...) = 0

override fun delete(...) = 0

override fun getType(...) = null

}

1. **Пример обращения к источнику данных:** Обращение к контактам с помощью ContentResolver:

val cursor = contentResolver.query(ContactsContract.Contacts.CONTENT\_URI, null, null, null, null)

### Билет 13

1. **Понятие LogCat:** LogCat — это системный инструмент Android Studio для отслеживания логов, который выводит отладочные сообщения, ошибки и другую информацию, позволяющую анализировать состояние приложения.
2. **Принцип работы с контактами:** Чтобы работать с контактами, используется ContentProvider и ContentResolver:

val cursor = contentResolver.query(ContactsContract.Contacts.CONTENT\_URI, null, null, null, null)

if (cursor != null && cursor.moveToFirst()) {

val name = cursor.getString(cursor.getColumnIndex(ContactsContract.Contacts.DISPLAY\_NAME))

}

1. **Пример создания собственного источника данных:**

class MyDatabaseHelper(context: Context) : SQLiteOpenHelper(context, "MyDatabase.db", null, 1) {

override fun onCreate(db: SQLiteDatabase) {

db.execSQL("CREATE TABLE items (id INTEGER PRIMARY KEY, name TEXT)")

}

override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {}

}

**Билет 14**

1. **Категории сообщений LogCat:**
   * **Verbose** — подробные сообщения.
   * **Debug** — отладочные сообщения.
   * **Info** — информационные сообщения.
   * **Warn** — предупреждения.
   * **Error** — ошибки.
   * **Assert** — критические ошибки.
2. **Механизм работы с файлами:** Работа с файлами в Android осуществляется с помощью FileInputStream и FileOutputStream.

val file = File(filesDir, "sample.txt")

file.writeText("Hello, world!")

val content = file.readText()

1. **Приложение для вывода информации по изображениям в галерее:**

val cursor = contentResolver.query(MediaStore.Images.Media.EXTERNAL\_CONTENT\_URI, null, null, null, null)

while (cursor?.moveToNext() == true) {

val imagePath = cursor.getString(cursor.getColumnIndex(MediaStore.Images.Media.DATA))

Log.d("GalleryImage", "Image path: $imagePath")

}

cursor?.close()

### Билет 15

1. **Понятие ArrayAdapter:** ArrayAdapter — это класс, который связывает массив данных с представлением списка. Он автоматически преобразует каждый элемент массива в элемент списка.
2. **Механизм запуска компонентов другого приложения:** Android поддерживает вызов компонентов других приложений с использованием Intent.

val intent = Intent(Intent.ACTION\_VIEW, Uri.parse("http://www.example.com"))

startActivity(intent)

1. **Пример использования источника данных MediaStore:** Получение списка изображений из галереи с помощью MediaStore.

val cursor = contentResolver.query(MediaStore.Images.Media.EXTERNAL\_CONTENT\_URI, null, null, null, null)

while (cursor?.moveToNext() == true) {

val imagePath = cursor.getString(cursor.getColumnIndex(MediaStore.Images.Media.DATA))

Log.d("MediaStore", "Image path: $imagePath")

}

cursor?.close()

**Билет 16**

1. **Понятие компонента в Android приложении:** Компоненты Android — это основные модули, составляющие приложение. Ключевые компоненты:
   * **Activity** — экран пользовательского интерфейса.
   * **Service** — компонент для выполнения долгих операций в фоновом режиме.
   * **Broadcast Receiver** — компонент для перехвата и обработки широковещательных сообщений.
   * **Content Provider** — компонент для предоставления данных приложения другим приложениям.
2. **Роль компонента ExpandableListView:** ExpandableListView позволяет отображать иерархический список, где каждый элемент можно развернуть для отображения дочерних элементов.

val expandableListView: ExpandableListView = findViewById(R.id.expandableListView)

val parentItems = listOf("Parent 1", "Parent 2")

val childItems = mapOf("Parent 1" to listOf("Child 1.1", "Child 1.2"), "Parent 2" to listOf("Child 2.1"))

val adapter = ExpandableListAdapter(this, parentItems, childItems)

expandableListView.setAdapter(adapter)

1. **Основные операции с контактами**:

* Получение списка контактов:

val cursor = contentResolver.query(ContactsContract.Contacts.CONTENT\_URI, null, null, null, null)

* **Добавление нового контакта:** Используется ContentProviderOperation для добавления нового контакта.
* Удаление контакта:

contentResolver.delete(ContactsContract.Contacts.CONTENT\_URI, "DISPLAY\_NAME=?", arrayOf("Contact Name"))

### Билет 17

1. **Понятие ContentProvider:** ContentProvider предоставляет доступ к данным одного приложения другим приложениям через стандартизированный интерфейс. Это позволяет организовать доступ к данным (например, к контактам или медиафайлам) через URI.
2. **Реализация ExpandableListView:** Для использования ExpandableListView потребуется адаптер:

class CustomExpandableListAdapter(

private val context: Context,

private val parentItems: List<String>,

private val childItems: Map<String, List<String>>

) : BaseExpandableListAdapter() {

override fun getChild(groupPosition: Int, childPosition: Int) = childItems[parentItems[groupPosition]]!![childPosition]

override fun getGroup(groupPosition: Int) = parentItems[groupPosition]

override fun getGroupCount() = parentItems.size

override fun getChildrenCount(groupPosition: Int) = childItems[parentItems[groupPosition]]!!.size

override fun getGroupView(...) = // Настроить вид родителя

override fun getChildView(...) = // Настроить вид дочернего элемента

override fun hasStableIds() = true

override fun isChildSelectable(groupPosition: Int, childPosition: Int) = true

}

1. **Пример создания базы данных SQLite:**

val dbHelper = object : SQLiteOpenHelper(this, "MyDatabase.db", null, 1) {

override fun onCreate(db: SQLiteDatabase) {

db.execSQL("CREATE TABLE Contacts (id INTEGER PRIMARY KEY, name TEXT)")

}

override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {}

}

val db = dbHelper.writableDatabase

### Билет 18

1. **Определение URI:** URI (Uniform Resource Identifier) — это строка, которая определяет ресурс в интернете или в системе. В Android URI используется для указания пути к данным, например, content://com.example.app.provider/table.
2. **Роль компонента Spinner:** Spinner — это виджет для выбора элемента из выпадающего списка.

val spinner: Spinner = findViewById(R.id.spinner)

val items = listOf("Item 1", "Item 2", "Item 3")

val adapter = ArrayAdapter(this, android.R.layout.simple\_spinner\_item, items)

spinner.adapter = adapter

1. **Пример открытия базы данных SQLite:**

val dbHelper = SQLiteOpenHelper(this, "ExampleDB", null, 1)

val db = dbHelper.readableDatabase

**Билет 19**

1. **Методы для класса, унаследованного от ContentProvider:**
   * **onCreate()** — инициализация провайдера.
   * **query()** — запрос данных.
   * **insert()** — добавление данных.
   * **update()** — обновление данных.
   * **delete()** — удаление данных.
   * **getType()** — возвращает тип данных, используемый query().
2. **Реализация Spinner:**

val spinner: Spinner = findViewById(R.id.spinner)

val options = listOf("Option A", "Option B", "Option C")

val adapter = ArrayAdapter(this, android.R.layout.simple\_spinner\_item, options)

adapter.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item)

spinner.adapter = adapter

1. **Пример работы со списками:**

val listView: ListView = findViewById(R.id.listView)

val items = listOf("Apple", "Banana", "Cherry")

val adapter = ArrayAdapter(this, android.R.layout.simple\_list\_item\_1, items)

listView.adapter = adapter

**Билет 20**

1. **Примеры встроенных контент-провайдеров:**
   * **ContactsContract** — провайдер для работы с контактами.
   * **MediaStore** — провайдер для работы с медиафайлами.
   * **CalendarContract** — провайдер для работы с календарными событиями.
2. **Роль компонента GridView:** GridView позволяет отображать элементы в виде сетки. Часто используется для отображения изображений.

val gridView: GridView = findViewById(R.id.gridView)

val items = listOf("A", "B", "C", "D")

val adapter = ArrayAdapter(this, android.R.layout.simple\_list\_item\_1, items)

gridView.adapter = adapter

1. **Пример выполнения запроса к базе данных SQLite:**

val cursor = db.rawQuery("SELECT \* FROM Contacts WHERE name = ?", arrayOf("John"))

if (cursor.moveToFirst()) {

val name = cursor.getString(cursor.getColumnIndex("name"))

}

cursor.close()

**Билет 21**

1. **Примеры встроенных контент-провайдеров:**
   * **ContactsContract** — для доступа к контактам.
   * **MediaStore** — для доступа к медиафайлам.
   * **CalendarContract** — для работы с событиями календаря.
2. **Реализация Spinner:**

val spinner: Spinner = findViewById(R.id.spinner)

val items = listOf("Choice 1", "Choice 2", "Choice 3")

val adapter = ArrayAdapter(this, android.R.layout.simple\_spinner\_item, items)

spinner.adapter = adapter

1. **Реализация собственного адаптера для списка:**

class CustomListAdapter(context: Context, private val data: List<String>) : ArrayAdapter<String>(context, 0, data) {

override fun getView(position: Int, convertView: View?, parent: ViewGroup): View {

val view = convertView ?: LayoutInflater.from(context).inflate(android.R.layout.simple\_list\_item\_1, parent, false)

view.findViewById<TextView>(android.R.id.text1).text = data[position]

return view

}

}