DUMPSARENA

Google Developers Certification - Associate Android Developer (Kotlin and Java Exam)

Google Associate-Android-Developer

Version Demo

Total Demo Questions: 10

Total Premium Questions: 128

Buy Premium PDF

https://dumpsarena.co

sales@dumpsarena.co

sales@dumpsarena.co
dumpsarena.co

Topic Break Down

Topic	No. of Questions
Topic 1, KOTLIN only	64
Topic 2, JAVA only	64
Total	128

QUESTION NO: 1

When your code execution reaches the breakpoint, Android Studio pauses execution of your app. You can then use the tools in the Debugger tab to identify the state of the app. With Evaluate Expression  you can

A. examine the object tree for a variable; expand it in the Variables view

B. evaluate an expression at the current execution point

C. advance to the next line in the code (without entering a method)

D. advance to the first line inside a method call

E. advance to the next line outside the current method

F. continue running the app normally

ANSWER: B

QUESTION NO: 2

By default, the notification's text content is truncated to fit one line. If you want your notification to be longer, for example, to create a larger text area, you can do it in this way:

A. var builder = NotificationCompat.Builder(this, CHANNEL_ID)
.setContentText("Much longer text that cannot fit one line...")
.setStyle(NotificationCompat.BigTextStyle()
.bigText("Much longer text that cannot fit one line...")) ...

B. var builder = NotificationCompat.Builder(this, CHANNEL_ID)
.setContentText("Much longer text that cannot fit one line...")
.setLongText("Much longer text that cannot fit one line..."))
...

C. var builder = NotificationCompat.Builder(this, CHANNEL_ID)
.setContentText("Much longer text that cannot fit one line...")
.setTheme(android.R.style.Theme_LongText); ...

ANSWER: A

Explanation:

Reference:

https://developer.android.com/training/notify-user/build-notification

QUESTION NO: 3

For example, our preferences.xml file was added by addPreferencesFromResource(R.xml.preferences). Our preferences.xml file contains such item:

android:entries="@array/sort_oder" android:entryValues="@array/sort_oder_value" android:defaultValue="@string/pref_default_sort_value" app:iconSpaceReserved="false" />

In our Fragment, we can dynamically get current notification preference value in this way:

A. val sortBy = PreferenceManager.getDefaultSharedPreferences(context).getString(context!!.getString(R.string.pref_sort_key), context!!.resources.getBoolean(R.bool.pref_default_sort_value))

B. val sortBy = PreferenceManager.getSharedPreferences(context).getString(context!!.getString(R.string.pref_default_sort_value), context!!.getString(R.string.pref_sort_key),)

C. val sortBy = PreferenceManager.getSharedPreferences(context).getBoolean(context!!.resources.getBoolean(R.bool.pref_default_sort_value), context!!.getString(R.string.pref_sort_key))

D. val sortBy = PreferenceManager.getDefaultSharedPreferences(context).getString(context!!.getString(R.string.pref_sort_key), context!!.getString(R.string.pref_default_sort_value))

ANSWER: D

QUESTION NO: 4

When scheduling unique work, you must tell WorkManager what action to take when there is a conflict. You do this by passing an enum when enquing the work. For one-time work, you provide an ExistingWorkPolicy, which supports some options for handling the conflict. (Choose four.)

A. REPLACE (existing work with the new work. This option cancels the existing work)

B. KEEP (existing work and ignore the new work)

C. APPEND (the new work to the end of the existing work. This policy will cause your new work to be chained to the existing work, running after the existing work finishes)

D. APPEND_OR_REPLACE (functions similarly to APPEND, except that it is not dependent on prerequisite work status. If the existing work is CANCELLED or FAILED, the new work still runs)

E. APPEND_OR_KEEP (functions similarly to APPEND, except that it is not dependent on prerequisite work status. If the existing work is CANCELLED or FAILED, the new work still not runs)

F. APPEND_AND_RUN (functions similarly to APPEND, except that it is not dependent on prerequisite work status. If the existing work is PAUSED, the new work still runs)

G. DESTROY (if any work exists, the new work will be ignored)

H. APPEND_OR_DESTROY (if no any work exists, the new work will be ignored)

ANSWER: A B C D

Explanation:

Videos:

- Working with WorkManager, from the 2018 Android Dev Summit
- WorkManager: Beyond the basics, from the 2019 Android Dev Summit

Reference: https://developer.android.com/reference/androidx/work/WorkManager?hl=en

QUESTION NO: 5

For example, we have a BufferedReader reader, associated with the json file through InputStreamReader. To get a file data we can do this:

A. var line: String? try {
while (reader.readLine().also { line = it } != null) { builder.append(line)
}
val json = JSONObject(builder.toString()) return json
} catch (exception: IOException) { exception.printStackTrace() } catch (exception: JSONException) {
exception.printStackTrace()
}

B. var line: JSONObject ? try {
while (reader.readJSONObject ().also { line = it } != null) { builder.append(line)
}
val json = JSONObject(builder.toString()) return json
} catch (exception: IOException) { exception.printStackTrace() } catch (exception: JSONException) {
exception.printStackTrace()
}

C. var line: String?
try {
while (reader.readLine().also { line = it } != null) { builder.append(line)
}
val json = JSONObject(builder.toString()) return json
} catch (exception: RuntimeException) { exception.printStackTrace()
} catch (exception: ArrayIndexOutOfBoundsException) { exception.printStackTrace()
}

ANSWER: A

QUESTION NO: 6

If you want to access a specific UI component in an app, use the UiSelector class. This class represents a query for specific elements in the currently displayed UI. What is correct about it?

(Choose two.)

A. If more than one matching element is found, the first matching element in the layout hierarchy is returned as the target UiObject.

B. If no matching UI element is found, an IOException is thrown.

C. If more than one matching element is found, the last matching element in the layout hierarchy is returned as the target UiObject.

D. If no matching UI element is found, a UiAutomatorObjectNotFoundException is thrown.

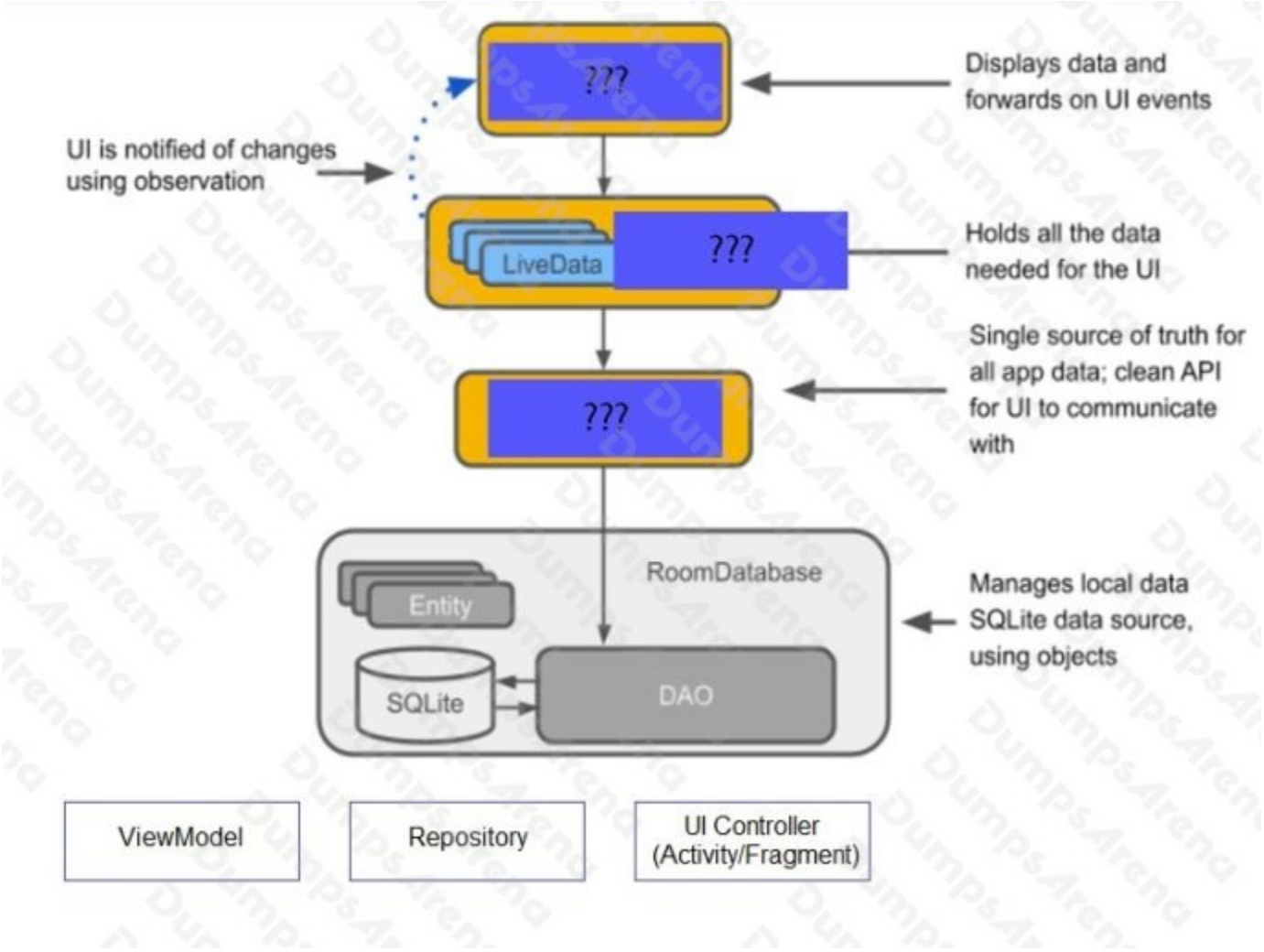
ANSWER: A D

QUESTION NO: 7 - (DRAG DROP)

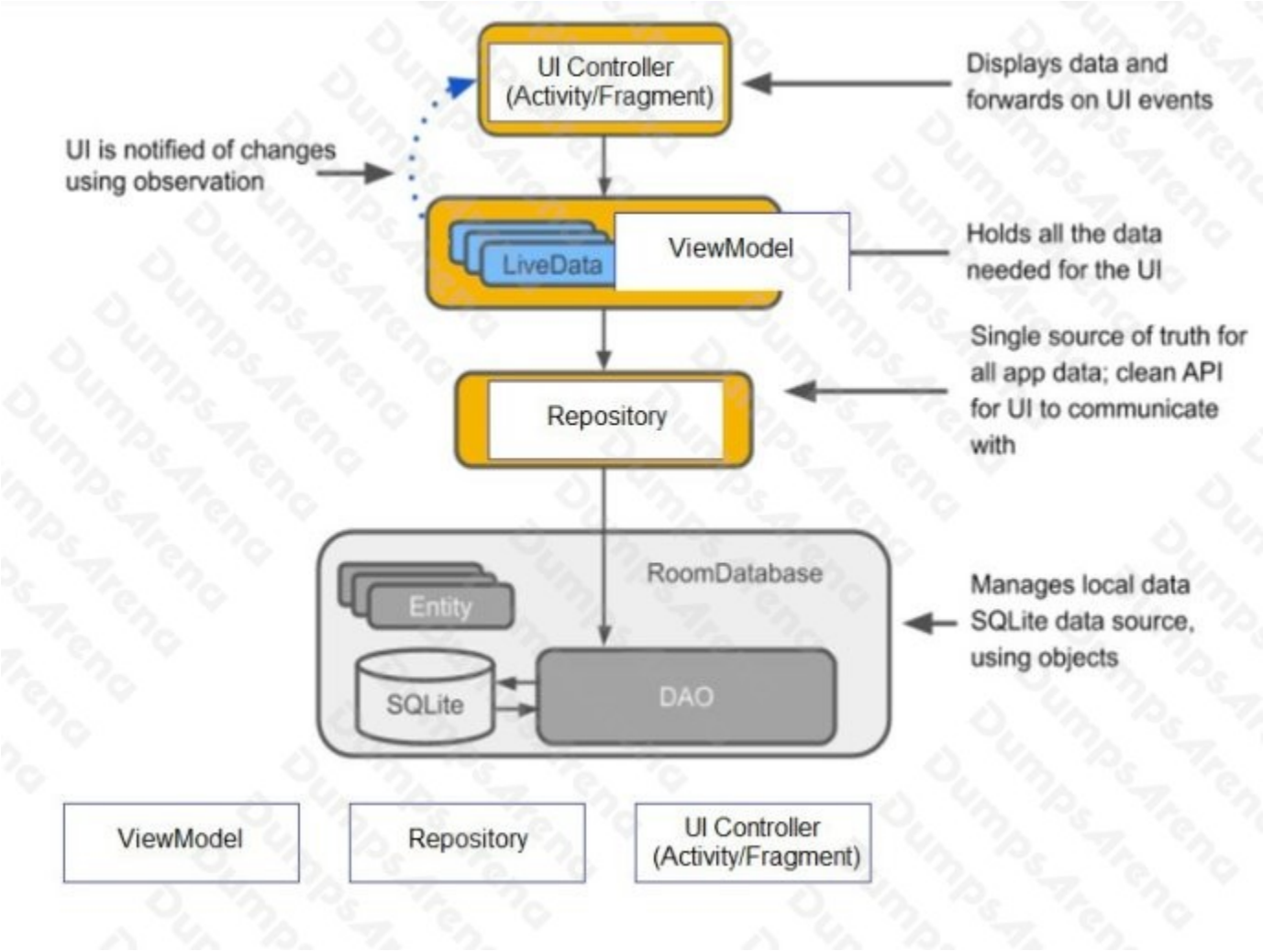
DRAG DROP

The diagram below shows a basic form of the recommended architecture for apps that use Architecture Components. The architecture consists of a UI controller, a ViewModel that serves LiveData, a Repository, and a Room database. Drag modules to correct places.

Select and Place:


???
Displays data and forwards on UI events
UI is notified of changes using observation
LiveData
???
Holds all the data needed for the UI
Single source of truth for all app data; clean API for UI to communicate with
???
RoomDatabase
Entity
SQLite
DAO
Manages local data SQLite data source, using objects
ViewModel
Repository
UI Controller (Activity/Fragment)

ANSWER:



Explanation:

QUESTION NO: 8

An example. In our ViewModelFactory (that implements ViewModelProvider.Factory) we have an instance of our Repository, named mRepository. Our ViewModel has such constructor:

class MyViewModel(private val mRepository: MyRepository) : ViewModel() ...

Next, in our ViewModelFactory create ViewModel method (overriden) looks like this: override fun create(modelClass: Class): T { return try {

//MISSED RETURN VALUE HERE”

} catch (e: InstantiationException) {

throw RuntimeException("Cannot create an instance of $modelClass", e)

} catch (e: IllegalAccessException) {

throw RuntimeException("Cannot create an instance of $modelClass", e)

} catch (e: NoSuchMethodException) {

throw RuntimeException("Cannot create an instance of $modelClass", e)

} catch (e: InvocationTargetException) {

throw RuntimeException("Cannot create an instance of $modelClass", e)

} }

What should we write instead of "//MISSED RETURN VALUE HERE"?

A. modelClass.getConstructor()
.newInstance(mRepository)

B. modelClass.getConstructor(MyRepository::class.java) .newInstance()

C. modelClass.getConstructor(MyRepository::class.java)
.newInstance(mRepository)

ANSWER: C

QUESTION NO: 9

By adding a RoomDatabase.Callback to the room database builder RoomDatabase.Builder (method addCallback(RoomDatabase.Callback callback)), we can: (Choose two.)

A. set the database factory

B. handle database first time creation

C. handle database opening

D. disable the main thread query check for Room

ANSWER: B C

QUESTION NO: 10

If you added to your build.gradle file a room.schemaLocation:

android { defaultConfig { javaCompileOptions { annotationProcessorOptions {

arguments = ["room.schemaLocation": "$projectDir/schemas".toString()]

}

}

}

}

Then, you build your app or module.

As a result you got a json file, with such path to it:

app/schemas/your_app_package/db_package/DbClass/DB_VERSION.json

What are the correct statements about this file? (Choose all that apply.)

A. It's a file with Room-exported schema

B. Main JSONObject in this file usually should contain a number "formatVersion" and a JSONObject "database"

C. The JSONObject "database" in this file usually should contain such objects, like "entities", "views", "setupQueries", ets.

ANSWER: A B C

Explanation:

Exported schema file example:

{

"formatVersion": 1,

"database": {

"version": 1,

"identityHash": "d90c93040756d2b94a178d5555555555",

"entities": [

{

"tableName": "tea_table",

"createSql": "CREATE TABLE IF NOT EXISTS `${TABLE_NAME}` (`id` INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, `name` TEXT, `type` TEXT,

`origin` TEXT, `steep_times` INTEGER, `Description` TEXT, `ingredients` TEXT, `cafeinLevel` TEXT, `favorite` INTEGER)",

"fields": [

{

"fieldPath": "mId",

"columnName": "id",

"affinity": "INTEGER",

"notNull": true

},

{

"fieldPath": "mName",

"columnName": "name",

"affinity": "TEXT",

"notNull": false

},

{

"fieldPath": "mType",

"columnName": "type",

"affinity": "TEXT",

"notNull": false

},

{

"fieldPath": "mOrigin",

"columnName": "origin",

"affinity": "TEXT",

"notNull": false

},

{

"fieldPath": "mSteepTimeMs",

"columnName": "steep_times",

"affinity": "INTEGER",

"notNull": false

},

{

"fieldPath": "mDescription",

"columnName": "Description",

"affinity": "TEXT",

"notNull": false

},

{

"fieldPath": "mIngredients",
"columnName": "ingredients",
"affinity": "TEXT",
"notNull": false
},
{
"fieldPath": "mCaffeineLevel",
"columnName": "cafeinLevel",
"affinity": "TEXT",
"notNull": false
},
{
"fieldPath": "mFavorite",
"columnName": "favorite",
"affinity": "INTEGER",
"notNull": false
}
],
"primaryKey": {
"columnNames": [
"id"
],
"autoGenerate": true
},
"indices": [],
"foreignKeys": []
}
],
"views": [],
"setupQueries": [

"CREATE TABLE IF NOT EXISTS room_master_table (id INTEGER PRIMARY KEY,identity_hash TEXT)",

"INSERT OR REPLACE INTO room_master_table (id,identity_hash) VALUES(42,
'd90c93040756d2b94a178d5555555555')"

]

}

}