

**Associate Android Developer.VCEplus.premium.exam.107q**

Passing Score: 800  
Time Limit: 120 min  
File Version: 1.0



**Website:** <https://vceplus.com>

**VCE to PDF Converter:** <https://vceplus.com/vce-to-pdf/>

**Facebook:** <https://www.facebook.com/VCE.For.All.VN/>

**Twitter :** [https://twitter.com/VCE\\_Plus](https://twitter.com/VCE_Plus)

**Associate Android Developer**

**Version 1.0**



**Sections**

1. KOTLIN only
2. JAVA only

## Exam A

### QUESTION 1

What is a correct part of an Implicit Intent for sharing data implementation?

- A. 

```
val sendIntent = Intent(this, UploadService::class.java).apply { putExtra(Intent.EXTRA_TEXT, textMessage) ... }
```
- B. 

```
val sendIntent = Intent().apply { type = Intent.ACTION_SEND; ... }
```
- C. 

```
val sendIntent = Intent(this, UploadService::class.java).apply { data = Uri.parse(fileUrl) ... }
```
- D. 

```
val sendIntent = Intent().apply { action = Intent.ACTION_SEND ... }
```

**Correct Answer:** D

**Section:** KOTLIN only

**Explanation**

**Explanation/Reference:**

Explanation:

Create the text message with a string

```
val sendIntent = Intent().apply { action = Intent.ACTION_SEND putExtra(Intent.EXTRA_TEXT, textMessage) type = "text/plain" }
```

Reference: <https://developer.android.com/guide/components/fundamentals>



### QUESTION 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); ...
```

**Correct Answer:** A

**Section:** KOTLIN only

**Explanation**

**Explanation/Reference:**

Reference:

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

### QUESTION 3

Select correct demonstration of WorkRequest cancellation.

- A. `workManager.enqueue(OneTimeWorkRequest.Builder(FooWorker::class.java).build())`
- B. `val request: WorkRequest = OneTimeWorkRequest.Builder(FooWorker::class.java).build()`  
`workManager.enqueue(request)`  
`val status = workManager.getWorkInfoByIdLiveData(request.id)`  
`status.observe(...)`
- C. `val request: WorkRequest = OneTimeWorkRequest.Builder(FooWorker::class.java).build()`  
`workManager.enqueue(request) workManager.cancelWorkById(request.id)`
- D. `val request1: WorkRequest = OneTimeWorkRequest.Builder(FooWorker::class.java).build()`  
`val request2: WorkRequest = OneTimeWorkRequest.Builder(BarWorker::class.java).build()`  
`val request3: WorkRequest = OneTimeWorkRequest.Builder(BazWorker::class.java).build()`  
`workManager.beginWith(request1, request2).then(request3).enqueue()`
- E. `val request: WorkRequest = OneTimeWorkRequest.Builder(FooWorker::class.java).build()`  
`workManager.enqueue(request) workManager.cancelWork(request)`

**Correct Answer: C**

**Section: KOTLIN only**

**Explanation**

**Explanation/Reference:**

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 4

In general, you should send an `AccessibilityEvent` whenever the content of your custom view changes. For example, if you are implementing a custom slider bar that allows a user to select a numeric value by pressing the left or right arrows, your custom view should emit an event of type `TYPE_VIEW_TEXT_CHANGED` whenever the slider value changes. Which one of the following sample codes demonstrates the use of the `sendAccessibilityEvent()` method to report this event.

- A. 

```
override fun dispatchPopulateAccessibilityEvent(event:
AccessibilityEvent): Boolean {    return
super.dispatchPopulateAccessibilityEvent(event).let { completed ->
if (text?.isEmpty() == true) {          event.text.add(text)
true      } else {          completed
      }
    }
}
```
- B. 

```
override fun onKeyUp(keyCode: Int, event: KeyEvent): Boolean {
return when(keyCode) {
    KeyEvent.KEYCODE_DPAD_LEFT -> {
currentValue--
        sendAccessibilityEvent(AccessibilityEvent.TYPE_VIEW_TEXT_CHANGED)
true
    }
    ...
}
```
- C. 

```
override fun onKeyUp(keyCode: Int, event: KeyEvent): Boolean {
return when(keyCode) {
    KeyEvent.KEYCODE_ENTER -> {
currentValue--
        sendAccessibilityEvent(AccessibilityEvent.TYPE_VIEW_CONTEXT_CLICKED)
true
    }
    ...
}
```

```
}
```

**Correct Answer:** B

**Section:** KOTLIN only

**Explanation**

**Explanation/Reference:**

Reference: <https://developer.android.com/guide/topics/ui/accessibility/custom-views>

#### QUESTION 5

The easiest way of adding menu items (to specify the options menu for an activity) is inflating an XML file into the Menu via MenuInflater. With *menu\_main.xml* we can do it in this way:

```
A. override fun onCreateOptionsMenu(menu: Menu): Boolean
{
    menuInflater.inflate(R.menu.menu_main, menu)
    return true
}

B. override fun onOptionsItemSelected(item: MenuItem):
Boolean { menuInflater.inflate(R.menu.menu_main, menu)
    return super.onOptionsItemSelected(item)
}

C. override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.menu.menu_main) }
```

**Correct Answer:** A

**Section:** KOTLIN only

**Explanation**

**Explanation/Reference:**

Reference: <https://developer.android.com/guide/topics/ui/accessibility/custom-views>



#### QUESTION 6

Android Tests. You can use the `childSelector()` method to nest multiple `UiSelector` instances. For example, the following code example shows how your test might specify a search to find the first `ListView` in the currently displayed UI, then search within that `ListView` to find a UI element with the text property `Apps`. What is the correct sample?

```
A. val appItem: UiObject = device.findObject(
    UiSelector().className(ListView.class)
        .instance(1)
        .childSelector(
            UiSelector().text("Apps")
        )
)

B. val appItem: UiObject = device.findObject(
    UiSelector().className("android.widget.ListView")
        .instance(0)
        .childSelector(
            UiSelector().text("Apps")
        )
)

C. val appItem: UiObject = device.findObject(
    UiSelector().className("android.widget.ListView")
        .instance(
            UiSelector().text("Apps")
        )
)
```

**Correct Answer:** B

**Section: KOTLIN only**

**Explanation**

**Explanation/Reference:**

**QUESTION 7** The following code snippet shows an example of an Espresso test:

A. 

```
@Rule fun
greeterSaysHello() {
    onView(withId(R.id.name_field)).do(typeText("Steve"))
    onView(withId(R.id.greet_button)).do(click())
    onView(withText("Hello Steve!")).check(matches(isDisplayed()))
}
```

B. 

```
@Test fun
greeterSaysHello() {
    onView(withId(R.id.name_field)).perform(typeText("Steve"))
    onView(withId(R.id.greet_button)).perform(click())
    onView(withText("Hello Steve!")).check(matches(isDisplayed()))
}
```

C. 

```
@Test fun
greeterSaysHello() {
    onView(withId(R.id.name_field)).do(typeText("Steve"))
    onView(withId(R.id.greet_button)).do(click())
    onView(withText("Hello Steve!")).compare(matches(isDisplayed()))
}
```

**Correct Answer: B**

**Section: KOTLIN only**

**Explanation**

**Explanation/Reference:**



**QUESTION 8**

As an example. In an Activity we have our `TimerViewModel` object (extended `ViewModel`), named `mTimerViewModel`. `mTimerViewModel.timer` method returns a `LiveData<Long>` value. What can be a correct way to set an observer to change UI in case if data was changed?

A. `mTimerViewModel!!.timer.value.toString().observe`  
`(Observer { aLong -> callAnyChangeUIMethodHere(aLong!!) })`

B. `mTimerViewModel!!.timer.observe`  
`(this, Observer { aLong -> callAnyChangeUIMethodHere(aLong!!) })`

C. `mTimerViewModel.observe`  
`(Observer { aLong -> callAnyChangeUIMethodHere(aLong!!) })`

**Correct Answer: B**

**Section: KOTLIN only**

**Explanation**

**Explanation/Reference:**

**QUESTION 9**

`LiveData.postValue()` and `LiveData.setValue()` methods have some differences. So if you have a following code executed in the main thread:

```
liveData.postValue("a");
liveData.setValue("b");
```

What will be the correct statement?

- A. The value "b" would be set at first and later the main thread would override it with the value "a".
- B. The value "a" would be set at first and later the main thread would override it with the value "b".
- C. The value "b" would be set at first and would not be overridden with the value "a".
- D. The value "a" would be set at first and would not be overridden with the value "b".

**Correct Answer:** B

**Section:** KOTLIN only

**Explanation**

**Explanation/Reference:**

#### QUESTION 10

In our `TeaViewModel` class, that extends `ViewModel`, we have such property:

```
val tea: LiveData<Tea>
```

An observer in our Activity (type of `mViewModel` variable in example is `TeaViewModel`) is set in this way:

```
mViewModel!!.tea.observe(this, Observer { tea: Tea? -> displayTea(tea) })
```

What will be a correct `displayTea` method definition?

- A. `private fun displayTea()`
- B. `private fun displayTea(tea: Tea?)`
- C. `private fun displayTea(tea: LiveData<Tea>)`
- D. `private fun displayTea(tea: LiveData<T>)`

**Correct Answer:** B

**Section:** KOTLIN only

**Explanation**

**Explanation/Reference:**

#### QUESTION 11

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

```
<SwitchPreference    android:id="@+id/notification"
    android:key="@string/pref_notification_key"
    android:title="@string/pref_notification_title"
    android:summary="@string/pref_notification_summary"
    android:defaultValue="@bool/pref_notification_default_value"
    app:iconSpaceReserved="false"/>
```

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

- A. 

```
val isNotificationOn =
    PreferenceManager.getDefaultSharedPreferences(context).getBoolean(
        context!!.getString(R.string.pref_notification_key),
        context!!.resources.getBoolean(R.bool.pref_notification_default_value)
    )
```
- B. 

```
val isNotificationOn =
    PreferenceManager.getSharedPreferences(context).getBoolean(
        context!!.getString(R.string.pref_notification_default_value),
        context!!.getString(R.string.pref_notification_key), )
```

```
C. val isNotificationOn =
    PreferenceManager.getDefaultSharedPreferences(context).getBoolean(
        context!!.resources.getBoolean(R.bool.pref_notification_default_value),
        context!!.getString(R.string.pref_notification_key) )
```

**Correct Answer:** A

**Section:** KOTLIN only

**Explanation**

**Explanation/Reference:**

#### QUESTION 12

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

```
<ListPreference    android:id="@+id/order_by"
    android:key="@string/pref_sort_key"
    android:title="@string/pref_sort_title"
    android:summary="@string/pref_sort_summary"
    android:dialogTitle="@string/pref_sort_dialog_title"
    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) )
```



**Correct Answer:** D

**Section:** KOTLIN only

**Explanation**

**Explanation/Reference:**

#### QUESTION 13

For example, we have a file in our raw folder `app/src/main/res/raw/sample_treas.json`. To get an `InputStream` for reading it, from our `Context context`, we can do this:

- A. 

```
val input = context!!.openRawResource(R.raw.sample_treas)
```
- B. 

```
val input = context!!.getRawResource(R.raw.sample_treas)
```
- C. 

```
val input = context!!.resources.openRawResource(R.raw.sample_treas)
```

**Correct Answer:** C

**Section:** KOTLIN only

**Explanation**

**Explanation/Reference:****QUESTION 14**

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()
}
```



**Correct Answer: A**

**Section: KOTLIN only**

**Explanation**

**Explanation/Reference:****QUESTION 15**

For example, we have a file in our assets folder `app/src/main/assets/sample_tas.json`. To get an `InputStream` for reading it, from out `Context` context, we can try do this:

A. `val input = context!!.resources.openRawResource(R.raw.sample_tas)`

B. `val input = context!!.assets.open("sample_tas.json")`

C. `val input = context!!.resources.assets.open("sample_tas.json")`

**Correct Answer: B**



## Section: KOTLIN only

### Explanation

#### Explanation/Reference:

### QUESTION 16

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 <T : ViewModel?> create(modelClass: Class<T>): 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)`

**Correct Answer: C**

## Section: KOTLIN only

### Explanation

#### Explanation/Reference:

**QUESTION 17** What is demonstrated by the code below?

```
// RawDao.kt @Dao interface RawDao {    @RawQuery    fun
getUserViaQuery(query: SupportSQLiteQuery?): User? }

// Usage of RawDao
...
val query =
    SimpleSQLiteQuery("SELECT * FROM User WHERE id = ? LIMIT 1",
arrayOf<Any>(sortBy)) val user = rawDao.getUserViaQuery(query)
...
```

- A. A method in a Dao annotated class as a raw query method where you can pass the query as a `SupportSQLiteQuery`.
- B. A method in a Dao annotated class as a query method.
- C. A method in a `RoomDatabase` class as a query method.

**Correct Answer: A**

## Section: KOTLIN only

### Explanation

**Explanation/Reference:**

**QUESTION 18** What happens when you create a DAO method and annotate it with @Insert?

Example:

```
@Dao
interface MyDao {
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    fun insertUsers(vararg users: User) }
```

- A. Room generates an implementation that inserts all parameters into the database in a single transaction.
- B. Room modifies a set of entities, given as parameters, in the database. It uses a query that matches against the primary key of each entity.
- C. Room removes a set of entities, given as parameters, from the database. It uses the primary keys to find the entities to delete.

**Correct Answer:** A

**Section:** KOTLIN only

**Explanation**

**Explanation/Reference:**

**QUESTION 19**

What do you want from Room when you create a DAO method and annotate it with @Update?

Example:

```
@Dao interface
MyDao {
    @Update
    fun updateUsers(vararg users: User)
}
```



- A. Room generates an implementation that inserts all parameters into the database in a single transaction.
- B. Room modifies a set of entities, given as parameters, in the database. It uses a query that matches against the primary key of each entity.
- C. Room removes a set of entities, given as parameters, from the database. It uses the primary keys to find the entities to delete.

**Correct Answer:** B

**Section:** KOTLIN only

**Explanation**

**Explanation/Reference:**

**QUESTION 20**

What do you want from Room when you create a DAO method and annotate it with @Delete?

Example:

```
@Dao interface
MyDao {
    @Delete
    fun deleteUsers(vararg users: User)
}
```

- A. Room generates an implementation that inserts all parameters into the database in a single transaction.
- B. Room modifies a set of entities, given as parameters, in the database. It uses a query that matches against the primary key of each entity.
- C. Room removes a set of entities, given as parameters, from the database. It uses the primary keys to find the entities to delete.

**Correct Answer:** C  
**Section:** KOTLIN only  
**Explanation**

**Explanation/Reference:**

#### QUESTION 21

To automate UI tests with Android Studio, you implement your test code in a separate Android test folder. Folder could be named:

- A. app/androidTest/java
- B. app/src/androidTest/java
- C. app/java/androidTest

**Correct Answer:** B  
**Section:** KOTLIN only  
**Explanation**

**Explanation/Reference:**

#### QUESTION 22

Once your test has obtained a UiObject object, you can call the methods in the UiObject class to perform user interactions on the UI component represented by that object. You can specify such actions as: (Choose four.)

- A. `click()` : Clicks the center of the visible bounds of the UI element.
- B. `touch()` : Touch the center of the visible bounds of the UI element.
- C. `dragTo()` : Drags this object to arbitrary coordinates.
- D. `moveTo()` : Move this object to arbitrary coordinates.
- E. `setText()` : Sets the text in an editable field, after clearing the field's content. Conversely, the `clearTextField()` method clears the existing text in an editable field.
- F. `swipeUp()` : Performs the swipe up action on the UiObject. Similarly, the `swipeDown()`, `swipeLeft()`, and `swipeRight()` methods perform corresponding actions.

**Correct Answer:** ACEF  
**Section:** KOTLIN only  
**Explanation**

**Explanation/Reference:**

#### QUESTION 23

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.

**Correct Answer:** AD  
**Section:** KOTLIN only  
**Explanation**

**Explanation/Reference:**

#### QUESTION 24

Each time your test invokes `onView()`, Espresso waits to perform the corresponding UI action or assertion until the following synchronization conditions are met: (Choose three.)

- A. The message queue is empty.

- B. The message queue is not empty.
- C. There are some instances of AsyncTask currently executing a task.
- D. There are no instances of AsyncTask currently executing a task.
- E. Some developer-defined idling resources are not idle.
- F. All developer-defined idling res






**Correct Answer:** ADF

**Section:** KOTLIN only

**Explanation**

**Explanation/Reference:**

**QUESTION 25** To run your local unit tests, follow these steps:

1. Be sure your project is synchronized with Gradle by clicking **Sync Project**  in the toolbar.
2. Run your test in one of the following ways (select possible): (Choose three.)
  - A.  To run a single test, open the Project window, and then right-click a test and click Run .
  - B. To test all methods in a class, right-click a class or method in the test file and click Run.
  - C.  To run all tests in a directory, right-click on the directory and select Run tests.
  - D.  To run all tests in Project, open the Project window, and then right-click a test and click Run.

**Correct Answer:** ABC

**Section:** KOTLIN only

**Explanation**

**Explanation/Reference:**

**QUESTION 26**

To create a basic JUnit 4 test class, create a class that contains one or more test methods. A test method begins with the specific annotation and contains the code to exercise and verify a single functionality in the component that you want to test. What is the annotation?

- A. @RunWith
- B. @LargeTest
- C. @Rule
- D. @Test

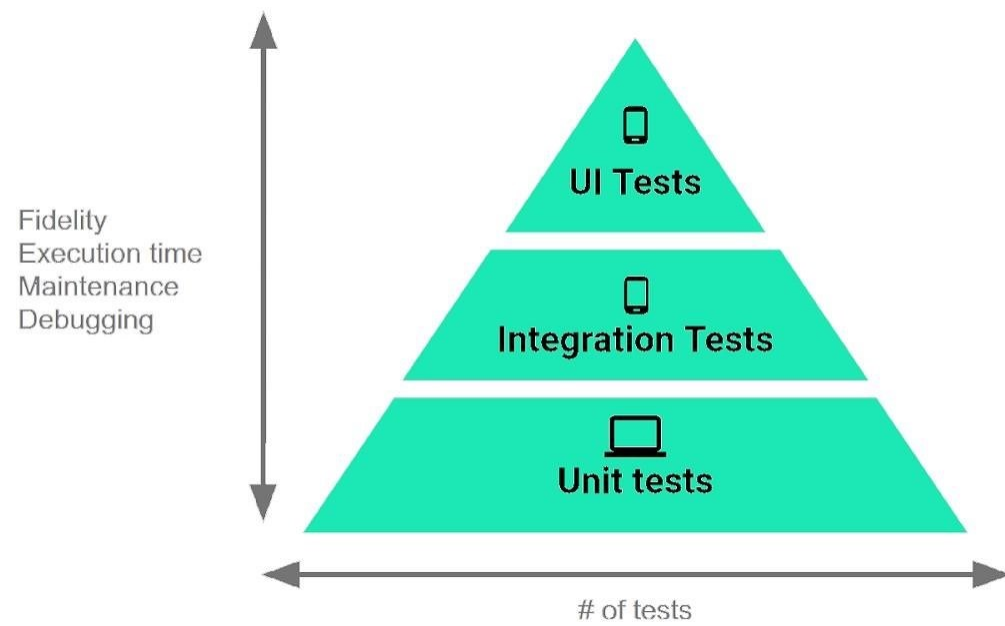
**Correct Answer:** D

**Section:** KOTLIN only

**Explanation**

**Explanation/Reference:**

**QUESTION 27**



The Testing Pyramid, shown in the Figure, illustrates how your app should include the three categories of tests: small, medium, and large. Medium tests are integration tests that:

- A. validate your app's behavior one class at a time.
- B. validate either interactions between levels of the stack within a module, or interactions between related modules.
- C. validate user journeys spanning multiple modules of your app.

**Correct Answer:** B

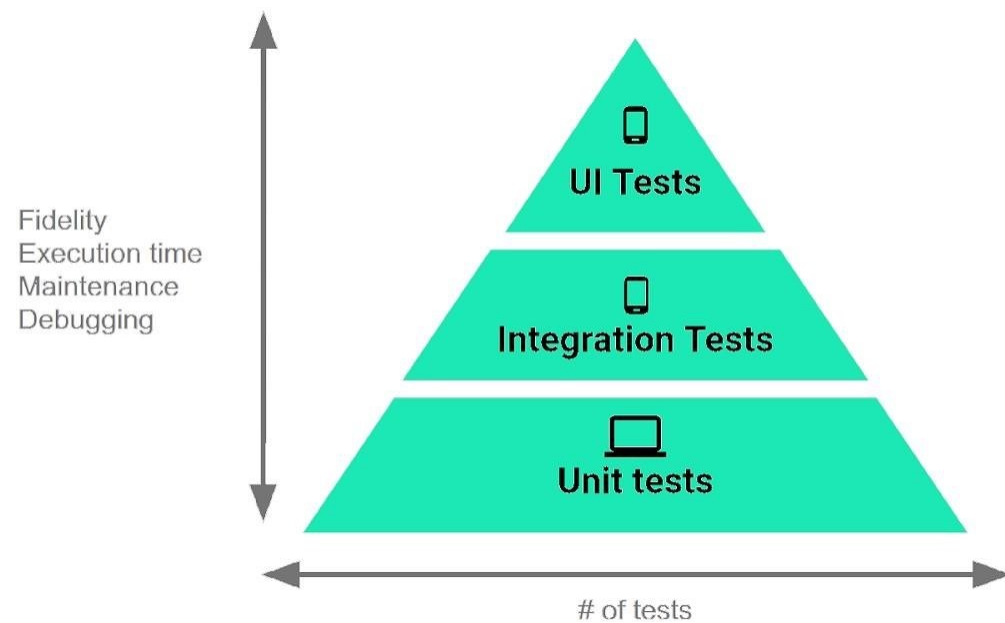
**Section:** KOTLIN only

**Explanation**

**Explanation/Reference:**



#### QUESTION 28



The Testing Pyramid, shown in the Figure, illustrates how your app should include the three categories of tests: small, medium, and large. Small tests are unit tests that :

- A. validate your app's behavior one class at a time.

- B. validate either interactions between levels of the stack within a module, or interactions between related modules.
- C. validate user journeys spanning multiple modules of your app.

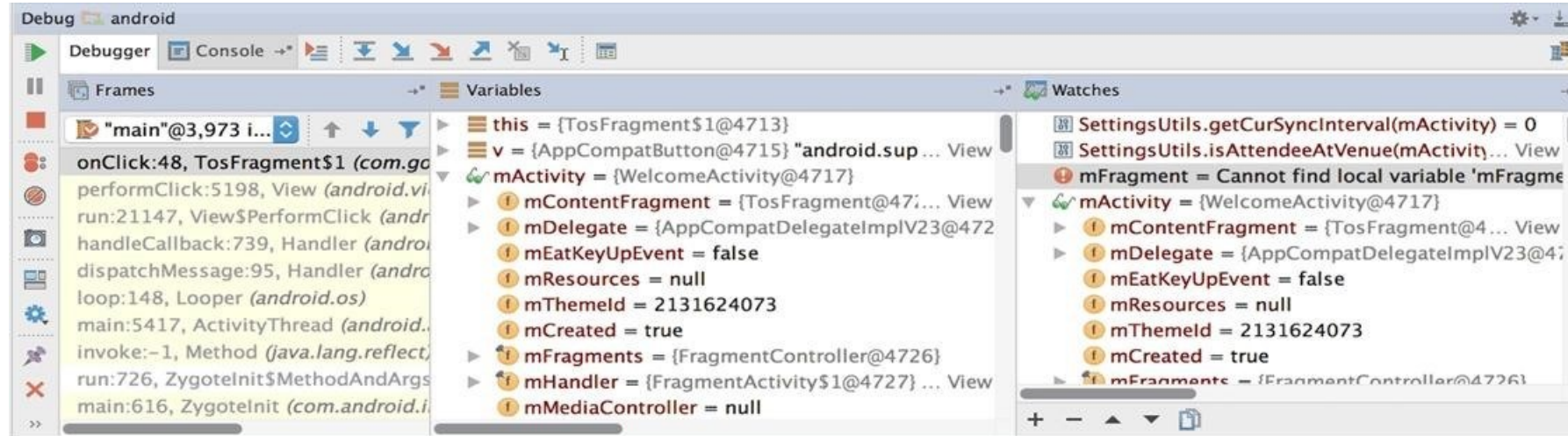
**Correct Answer: A**

**Section: KOTLIN only**

**Explanation**

**Explanation/Reference:**

#### QUESTION 29



What is illustrated in the picture?

- A. Logcat window with filter settings
- B. Debugging native code using LLDB
- C. The Variables and Watches panes in the Debugger window
- D. The Breakpoints window lists all the current breakpoints and includes behavior settings for each
- E. Adding a watchpoint to a variable in memory



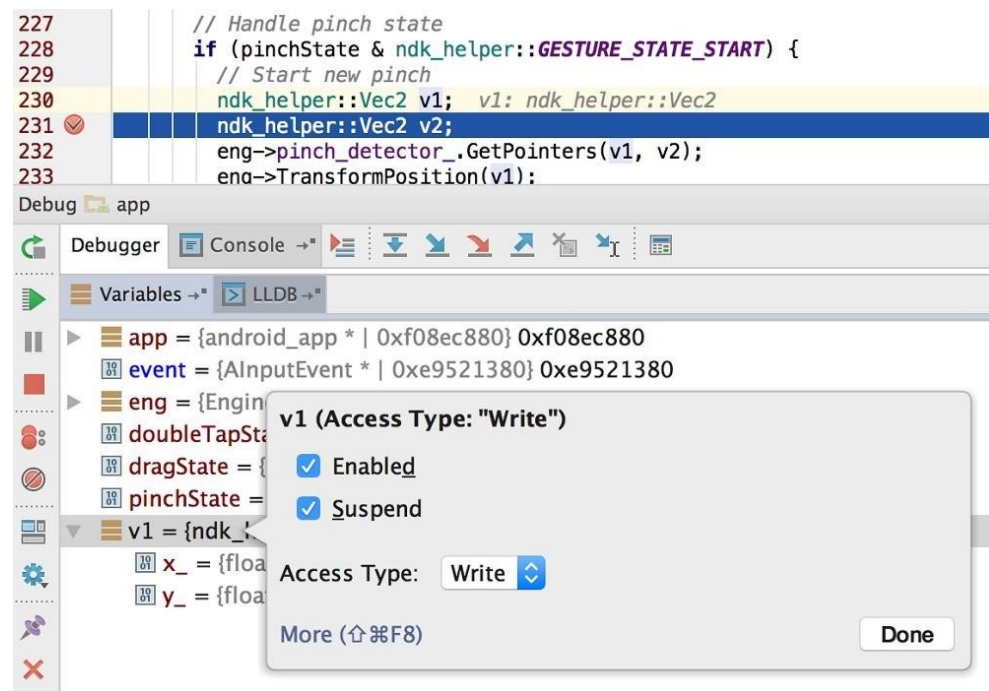
**Correct Answer: C**

**Section: KOTLIN only**

**Explanation**

**Explanation/Reference:**

#### QUESTION 30



What is illustrated in the picture?

- A. Logcat window with filter settings
- B. Debugging native code using LLDB
- C. The Variables and Watches panes in the Debugger window
- D. The Breakpoints window lists all the current breakpoints and includes behavior settings for each
- E. Adding a watchpoint to a variable in memory


**Correct Answer:** E

**Section:** KOTLIN only

**Explanation**

**Explanation/Reference:**

### QUESTION 31

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 Step Into  you can

- A. examine the object tree for a variable, expand it in the Variables view. If the Variables view is not visible
- 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


**Correct Answer:** D

**Section:** KOTLIN only

**Explanation**

**Explanation/Reference:**

### QUESTION 32

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


**Correct Answer:** B

**Section:** KOTLIN only

**Explanation**

**Explanation/Reference:**

### QUESTION 33

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 Step Over  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

**Correct Answer:** C


**Section:** KOTLIN only

**Explanation**

**Explanation/Reference:**



### QUESTION 34

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 Step Out  you can

- A. examine the object tree for a variable; expand it in the Variables view. If the Variables view is not visible
- 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

**Correct Answer:** E

**Section:** KOTLIN only

**Explanation**

**Explanation/Reference:**

### QUESTION 35

The Log class allows you to create log messages that appear in logcat. Generally, you could use the following log methods: (Choose five.)

- A. `Log.e(String, String)` (error)
- B. `Log.a(String, String)` (all outputs) C. `Log.w(String, String)` (warning)
- D. `Log.i(String, String)` (information)
- E. `Log.q(String, String)` (questions)
- F. `Log.d(String, String)` (debug)
- G. `Log.v(String, String)` (verbose)



**Correct Answer:** ACDFG

**Section:** KOTLIN only

**Explanation**

**Explanation/Reference:**

**QUESTION 36**

Filter logcat messages. If in the filter menu, a filter option “Show only selected application”? means:

- A. Display the messages produced by the app code only (the default). Logcat filters the log messages using the PID of the active app.
- B. Apply no filters. Logcat displays all log messages from the device, regardless of which process you selected.
- C. Create or modify a custom filter. For example, you could create a filter to view log messages from two apps at the same time.

**Correct Answer:** A

**Section:** KOTLIN only

**Explanation**

**Explanation/Reference:**

**QUESTION 37** Filter logcat messages. If in the filter menu, a filter option “Edit Filter Configuration”? means:

- A. Display the messages produced by the app code only (the default). Logcat filters the log messages using the PID of the active app.
- B. Apply no filters. Logcat displays all log messages from the device, regardless of which process you selected.
- C. Create or modify a custom filter. For example, you could create a filter to view log messages from two apps at the same time.

**Correct Answer:** C

**Section:** KOTLIN only

**Explanation**



**Explanation/Reference:**

**QUESTION 38**

The Layout Inspector in Android Studio allows you to compare your app layout with design mockups, display a magnified or 3D view of your app, and examine details of its layout at runtime. When this is especially useful?

- A. when your layout is built entirely in XML rather than runtime and the layout is behaving expectedly.
- B. when your layout is built at runtime rather than entirely in XML and the layout is behaving unexpectedly.

**Correct Answer:** B

**Section:** KOTLIN only

**Explanation**

**Explanation/Reference:**

**QUESTION 39**

If you want the Database Inspector to automatically update the data it presents as you interact with your running app, check the Live updates checkbox at the top of the inspector window. While live updates are enabled, what happens with the table in the inspector window?

- A. It is still editable. You can modify data in a table by double-clicking a cell, typing a new value, and pressing Enter.
- B. It becomes read-only and you cannot modify its values.
- C. It becomes read-only, but you cannot see its updated values before updating the data by clicking the Refresh table button at the top of the inspector window.

**Correct Answer:** B

**Section:** KOTLIN only

**Explanation**

**Explanation/Reference:**

**QUESTION 40**

Enable debugging on your device: If you are using the emulator, this is enabled by default. But for a connected device, you need to

- A. enable transfer data from the device in usb connection options.
- B. enable debugging in the device developer options.
- C. enable connection in bluetooth options.

**Correct Answer:** B

**Section:** KOTLIN only

**Explanation**

**Explanation/Reference:**

**QUESTION 41** To run a debuggable build variant you must use a build variant that includes

- A. minifyEnabled false in the build configuration
- B. debuggable true or debuggable false in the build configuration
- C. debuggable true in the build configuration

**Correct Answer:** C

**Section:** KOTLIN only

**Explanation**

**Explanation/Reference:**



**QUESTION 42**

About running a debuggable build variant. Usually, you can just select the default "debug" variant that's included in every Android Studio project (even though it's not visible in the build.gradle file). But if you define new build types that should be debuggable, you must add 'debuggable true' to the build type. Is that mostly true?

- A. Yes.
- B. No, if you define new build types that should be debuggable, you must add 'debuggable false'
- C. No, the debug variant should be visible in the build.gradle file anyway.

**Correct Answer:** A

**Section:** KOTLIN only

**Explanation**

**Explanation/Reference:**

**QUESTION 43** With a room database. When performing queries, you'll often want your app's UI to update automatically when the data changes. Can you use a return value of type LiveData in your query method description to achieve this?

- A. Yes
- B. No

**Correct Answer:** A

**Section:** KOTLIN only

**Explanation**

**Explanation/Reference:**

Explanation:

Room generates all necessary code to update the LiveData when the database is updated.

**QUESTION 44**

As an example. Our `MutableLiveData<Long>` object, named `mLapseTime`, is not connected to a Room database, etc. How can we change the value in `mLapseTime`?

- A. `mLapseTime.postValue("new String")`
- B. `mLapseTime.setValue(10001)`
- C. `mLapseTime.changeValue(10001)`

**Correct Answer:** B

**Section:** KOTLIN only

**Explanation**

**Explanation/Reference:**

**QUESTION 45** Interface for a callback to be invoked when a shared preference is changed.

Interface is named:

- A. `android.content.SyncStatusObserver`
- B. `android.content.SharedPreferences.Editor`
- C. `android.content.SharedPreferences.OnSharedPreferenceChangeListener`
- D. `android.content.SharedPreferences`

**Correct Answer:** C

**Section:** KOTLIN only

**Explanation**

**Explanation/Reference:**

**QUESTION 46**

With our Context we can get `SharedPreferences` with a method, named: `getSharedPreferences(String name, int mode)`. What value can we transfer in a "mode" parameter?

- A. `MODE_PRIVATE` or `MODE_PUBLIC`
- B. combination of `MODE_WORLD_READABLE` and `MODE_WORLD_WRITEABLE`
- C. Value is either 0 or a combination of `MODE_PRIVATE`, `MODE_WORLD_READABLE`, `MODE_WORLD_WRITEABLE`, and `MODE_MULTI_PROCESS`

**Correct Answer:** C

**Section:** KOTLIN only

**Explanation**

**Explanation/Reference:**

**QUESTION 47**

What statements about `InputStreamReader` (`java.io.InputStreamReader`) are correct? (Choose two.)

- A. An `InputStreamReader` is a bridge from byte streams to character streams: It reads bytes and decodes them into characters using a specified charset. The charset that it uses may be specified by name or may be given explicitly, or the platform's default charset may be accepted.
- B. An `InputStreamReader` is a bridge from character streams to byte streams: It reads characters using a specified charset and encodes them into bytes. The charset that it uses may be specified by name or may be given explicitly, or the platform's default charset may be accepted.
- C. Each invocation of one of an `InputStreamReader`'s `read()` methods may cause one or more bytes to be read from the underlying byte-input stream. To enable the efficient conversion of bytes to characters, more bytes may be read ahead from the underlying stream than are necessary to satisfy the current read operation.
- D. No any invocation of one of an `InputStreamReader`'s `read()` methods can cause some bytes to be read from the underlying byte-input stream.

**Correct Answer:** AC

**Section: KOTLIN only**

**Explanation**

**Explanation/Reference:**

**QUESTION 48** In a class extended `PreferenceFragmentCompat`. What method is used to inflate the given XML resource and add the preference hierarchy to the current preference hierarchy?

- A. `findPreference`
- B. `getPreferenceManager`
- C. `addPreferencesFromResource`
- D. `setPreferenceScreen`

**Correct Answer: C**

**Section: KOTLIN only**

**Explanation**

**Explanation/Reference:**

**QUESTION 49**

In a class `PreferenceFragmentCompat`. What method is called during `onCreate(Bundle)` to supply the preferences for this fragment. And where subclasses are expected to call `setPreferenceScreen(PreferenceScreen)` either directly or via helper methods such as `addPreferencesFromResource(int)`?

- A. `onCreateLayoutManager`
- B. `onCreatePreferences`
- C. `onCreateRecyclerView`
- D. `onCreateView`



**Correct Answer: B**

**Section: KOTLIN only**

**Explanation**

**Explanation/Reference:**

**QUESTION 50**

In a class `PreferenceFragmentCompat`. As a convenience, this fragment implements a click listener for any preference in the current hierarchy. So, in what overridden method we can handle that a preference in the tree rooted at this `PreferenceScreen` has been clicked?

- A. `onCreateLayoutManager`
- B. `onCreatePreferences`
- C. `onCreateRecyclerView`
- D. `onPreferenceTreeClick`

**Correct Answer: D**

**Section: KOTLIN only**

**Explanation**

**Explanation/Reference:**

**QUESTION 51**

`SharedPreferences.Editor` is an interface used for modifying values in a `SharedPreferences` object. All changes you make in an editor are batched, and not copied back to the original `SharedPreferences` until you call:

- A. `commit()`

B. `apply()`  
C. `commit()` or `apply()`

**Correct Answer: C**

**Section: KOTLIN only**

**Explanation**

**Explanation/Reference:**

#### QUESTION 52

`SharedPreferences.Editor` is an interface used for modifying values in a `SharedPreferences` object. To mark in the editor that a preference value should be removed, which will be done in the actual preferences once `commit()` or `apply()` is called, what method in `SharedPreferences.Editor` should we use?

A. `delete(String key)`  
B. `clear()`  
C. `remove(String key)`  
D. `removeAll()`

**Correct Answer: B**

**Section: KOTLIN only**

**Explanation**

**Explanation/Reference:**

Explanation:

`clear()` method marks in the editor to remove ALL values from the preferences. Once `commit` is called, the only remaining preferences will be any that you have defined in this editor.

And no `delete` and `removeAll` method exists in `SharedPreferences.Editor`

#### QUESTION 53 What is the incorrect statement about Data Access Object

(`androidx.room.Dao`)?



A. Data Access Objects are the main classes where you define your database interactions. They can include a variety of query methods.  
B. The class marked with `@Dao` should either be an interface or an abstract class. At compile time, Room will generate an implementation of this class when it is referenced by a Database.  
C. An abstract `@Dao` class can optionally have a constructor that takes a Database as its only parameter.  
D. It is recommended to have only one Dao class in your codebase for all tables.

**Correct Answer: D**

**Section: KOTLIN only**

**Explanation**

**Explanation/Reference:**

#### QUESTION 54

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

**Correct Answer: BC**

**Section: KOTLIN only**

**Explanation**

**Explanation/Reference:**

**QUESTION 55**

What is a correct part of an Implicit Intent for sharing data implementation?

A. `Intent sendIntent = new Intent(this, UploadService.class)`  
`sendIntent.putExtra(Intent.EXTRA_TEXT, textMessage);` B.  
`Intent sendIntent = new Intent();`  
`sendIntent.setType(Intent.ACTION_SEND);`  
C. `Intent sendIntent = new Intent(this, UploadService.class)`  
`sendIntent.setData(Uri.parse(fileUrl));` D. `Intent`  
`sendIntent = new Intent();`  
`sendIntent.setAction(Intent.ACTION_SEND);`

**Correct Answer:** D

**Section:** JAVA only

**Explanation**

**Explanation/Reference:**

Explanation:

Create the text message with a string

```
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, textMessage);
sendIntent.setType("text/plain");
```

Reference: <https://developer.android.com/guide/components/fundamentals>

**QUESTION 56**

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

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

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

**Correct Answer:** A

**Section:** JAVA only

**Explanation**

**Explanation/Reference:**

Reference:

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

**QUESTION 57** “workManager” is an instance of WorkManager. Select correct demonstration of WorkRequest cancellation:

A. `workManager.enqueue(new OneTimeWorkRequest.Builder(FooWorker.class).build());`

B. `WorkRequest request = new OneTimeWorkRequest.Builder(FooWorker.class).build();`  
`workManager.enqueue(request);`  
`LiveData<WorkInfo> status = workManager.getWorkInfoByIdLiveData(request.getId());`  
`status.observe(...);`

```
C. WorkRequest request = new OneTimeWorkRequest.Builder(FooWorker.class).build();
   workManager.enqueue(request); workManager.cancelWorkById(request.getId());

D. WorkRequest request1 = new OneTimeWorkRequest.Builder(FooWorker.class).build();
   WorkRequest request2 = new OneTimeWorkRequest.Builder(BarWorker.class).build();
   WorkRequest request3 = new OneTimeWorkRequest.Builder(BazWorker.class).build();
   workManager.beginWith(request1, request2).then(request3).enqueue();

E. WorkRequest request = new OneTimeWorkRequest.Builder(FooWorker.class).build();
   workManager.enqueue(request); workManager.cancelWork(request);
```

**Correct Answer: C**

**Section: JAVA only**

**Explanation**

**Explanation/Reference:**

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 58

In general, you should send an AccessibilityEvent whenever the content of your custom view changes. For example, if you are implementing a custom slider bar that allows a user to select a numeric value by pressing the left or right arrows, your custom view should emit an event of type TYPE\_VIEW\_TEXT\_CHANGED whenever the slider value changes. Which one of the following sample codes demonstrates the use of the sendAccessibilityEvent() method to report this event.

```
A. @Override public boolean
dispatchPopulateAccessibilityEvent(AccessibilityEvent
event) { boolean completed =
super.dispatchPopulateAccessibilityEvent(event);
CharSequence text = getText(); if
(!TextUtils.isEmpty(text)) {
event.getText().add(text); return true;
}
return completed;
}

B. @Override public boolean onKeyUp (int keyCode, KeyEvent
event) { if (keyCode == KeyEvent.KEYCODE_DPAD_LEFT)
{
currentValue--;
sendAccessibilityEvent(AccessibilityEvent.TYPE_VIEW_TEXT_CHANGED);
return true;
}
...
}

C. @Override public boolean onKeyUp (int keyCode, KeyEvent
event) { if (keyCode == KeyEvent.KEYCODE_ENTER) {
currentValue--;
sendAccessibilityEvent(AccessibilityEvent.TYPE_VIEW_CONTEXT_CLICKED);
return true;
}
...
}
```



**Correct Answer: B**

**Section: JAVA only**

**Explanation**

**Explanation/Reference:**

Reference:

<https://developer.android.com/guide/topics/ui/accessibility/custom-views>

#### QUESTION 59

The easiest way of adding menu items (to specify the options menu for an activity) is inflating an XML file into the Menu via MenuInflater. With *menu\_main.xml* we can do it in this way:

- A. 

```
@Override public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}
```
- B. 

```
@Override public boolean onOptionsItemSelected(MenuItem item) {
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return super.onOptionsItemSelected(item);
}
```
- C. 

```
@Override protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.menu.menu_main);
}
```

**Correct Answer: A**

**Section: JAVA only**

**Explanation**

**Explanation/Reference:**

Reference:

<https://developer.android.com/guide/topics/ui/menus>

#### QUESTION 60

Android Tests. You can use the `childSelector()` method to nest multiple `UiSelector` instances. For example, the following code example shows how your test might specify a search to find the first `ListView` in the currently displayed UI, then search within that `ListView` to find a UI element with the text property Apps. What is the correct sample?

- A. 

```
UiObject appItem = device.findObject(new UiSelector()
    .className(ListView.class)
    .instance(1)
    .childSelector(new UiSelector()
    .text("Apps")));
```
- B. 

```
UiObject appItem = device.findObject(new UiSelector()
    .className("android.widget.ListView")
    .instance(0)
    .childSelector(new UiSelector()
    .text("Apps")));
```
- C. 

```
UiObject appItem = device.findObject(new UiSelector()
    .className("android.widget.ListView")
    .instance(new UiSelector()
    .text("Apps")));
```



**Correct Answer: B**

**Section: JAVA only**

**Explanation**

**Explanation/Reference:**

**QUESTION 61** The following code snippet shows an example of an Espresso test:

- A. 

```
@Rule public void greeterSaysHello() {
    onView(withId(R.id.name_field)).do(typeText("Steve"));
    onView(withId(R.id.greet_button)).do(click());
}
```



```
        onView(withText("Hello Steve!")).check(matches(isDisplayed()));
    }
B. @Test public void
greeterSaysHello() {
    onView(withId(R.id.name_field)).perform(typeText("Steve"));
    onView(withId(R.id.greet_button)).perform(click());
    onView(withText("Hello Steve!")).check(matches(isDisplayed()));
}
C. @Test public void
greeterSaysHello() {
    onView(withId(R.id.name_field)).do(typeText("Steve"));
    onView(withId(R.id.greet_button)).do(click());
    onView(withText("Hello Steve!")).compare(matches(isDisplayed()));
}
```

**Correct Answer: B**

**Section: JAVA only**

**Explanation**

**Explanation/Reference:**

#### QUESTION 62

As an example. In an Activity we have our `TimerViewModel` object (extended `ViewModel`), named `mTimerViewModel`. `mTimerViewModel.getTimer()` method returns a `LiveData<Long>` value. What can be a correct way to set an observer to change UI in case if data was changed?

```
A. mTimerViewModel.getTimer().getValue().toString().observe(new
Observer<Long>() {    @Override
    public void onChanged(Long aLong) {
        callAnyChangeUIMethodHere(aLong)
    }
});
B. mTimerViewModel.getTimer().observe(this, new Observer<Long>() {
    @Override
    public void onChanged(Long aLong) {
        callAnyChangeUIMethodHere(aLong)
    }
});
C. mTimerViewModel.observe(new Observer<Long>() {    @Override
    public void onChanged(Long aLong) {
        callAnyChangeUIMethodHere(aLong)
    }
});
```



**Correct Answer: B**

**Section: JAVA only**

**Explanation**

**Explanation/Reference:**

#### QUESTION 63

`LiveData.postValue()` and `LiveData.setValue()` methods have some differences. So if you have a following code executed in the main thread:

```
liveData.postValue("a");
liveData.setValue("b");
```

What will be the correct statement?

- A. The value "b" would be set at first and later the main thread would override it with the value "a".
- B. The value "a" would be set at first and later the main thread would override it with the value "b".
- C. The value "b" would be set at first and would not be overridden with the value "a".
- D. The value "a" would be set at first and would not be overridden with the value "b".

**Correct Answer:** B

**Section:** JAVA only

**Explanation**

**Explanation/Reference:**

#### QUESTION 64

In our `TeaViewModel` class, that extends `ViewModel`, we have such method:

```
public LiveData<Tea> getTea() {  
    return mTea; }  

```

An observer in our Activity (type of `mViewModel` variable in example is `TeaViewModel`) is set in this way:

```
mViewModel.getTea().observe(this, this::displayTea);  

```

What will be a correct `displayTea` method definition?

- A. `private void displayTea()`
- B. `private void displayTea(Tea tea)`
- C. `private void displayTea(LiveData<Tea>)`
- D. `private void displayTea(LiveData<T>)`

**Correct Answer:** B

**Section:** JAVA only

**Explanation**

**Explanation/Reference:**

#### QUESTION 65

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

```
<SwitchPreference    android:id="@+id/notification"  
    android:key="@string/pref_notification_key"  
    android:title="@string/pref_notification_title"  
    android:summary="@string/pref_notification_summary"  
    android:defaultValue="@bool/pref_notification_default_value"  
    app:iconSpaceReserved="false"/>  

```

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

- A. `boolean isNotificationOn = PreferenceManager.getDefaultSharedPreferences(getContext()).getBoolean(getContext().getString(R.string.pref_notification_key), getContext().getResources().getBoolean(R.bool.pref_notification_default_value));`
- B. `boolean isNotificationOn = PreferenceManager.getSharedPreferences(getContext()).getBoolean(getContext().getString(R.string.pref_notification_default_value), getContext().getString(R.string.pref_notification_key));`

```
C. boolean isNotificationOn = PreferenceManager.getSharedPreferences(getContext()).getBoolean(
    getContext().getResources().getBoolean(R.bool.pref_notification_default_value) ,
    getContext().getString(R.string.pref_notification_key) );
```

**Correct Answer: A**  
**Section: JAVA only**  
**Explanation**

**Explanation/Reference:**  
**QUESTION 66**

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

```
<ListPreference    android:id="@+id/order_by"
    android:key="@string/pref_sort_key"
    android:title="@string/pref_sort_title"
    android:summary="@string/pref_sort_summary"
    android:dialogTitle="@string/pref_sort_dialog_title"
    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. String sortBy =  
 PreferenceManager.*getDefaultSharedPreferences*(getContext()).getString(  
 getContext().getString(R.string.pref\_sort\_key) ,  
 getContext().getResources().getBoolean(R.bool.pref\_default\_sort\_value)  
 );
- B. String sortBy = PreferenceManager.*getSharedPreferences*(getContext()).getString(  
 getContext().getString(R.string.pref\_default\_sort\_value) ,  
 getContext().getString(R.string.pref\_sort\_key) );
- C. boolean sortBy = PreferenceManager.*getSharedPreferences*(getContext()).getBoolean(  
 getContext().getResources().getBoolean(R.bool.pref\_default\_sort\_value) ,  
 getContext().getString(R.string.pref\_sort\_key) );
- D. String sortBy =  
 PreferenceManager.*getDefaultSharedPreferences*(getContext()).getString(  
 getContext().getString(R.string.pref\_sort\_key) ,  
 getContext().getString(R.string.pref\_default\_sort\_value) )

**Correct Answer: D**  
**Section: JAVA only**  
**Explanation**

**Explanation/Reference:**

**QUESTION 67**

For example, we have a file in our raw folder app/src/main/res/raw/sample\_treas.json. To get an InputStream for reading it, from our Context context, we can do this:

- A. InputStream input = context.openRawResource(R.raw.*sample\_treas*) ;
- B. InputStream input = context.getRawResource(R.raw.*sample\_treas*) ;
- C. InputStream input = context.getResources().openRawResource(R.raw.*sample\_treas*) ;

**Correct Answer: C**  
**Section: JAVA only**  
**Explanation**

**Explanation/Reference:****QUESTION 68**

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

A. `String line;`

```
try {
    while ((line = reader.readLine()) != null) {
        builder.append(line);
    }
    JSONObject json = new JSONObject(builder.toString());
    return json;
} catch (IOException | JSONException exception) {
    exception.printStackTrace();
}
```

B. `JSONObject`

```
line; try {
    while ((line = reader.readJSONObject ()) != null) {
        builder.append(line);
    }
    JSONObject json = new JSONObject(builder.toString());
    return json;
} catch (IOException | JSONException exception) {
    exception.printStackTrace();
}
```

C. `String line;`

```
try {
    while ((line = reader.readLine()) != null) {
        builder.append(line);
    }
    JSONObject json = new JSONObject(builder.toString());
    return json;
} catch (RuntimeException|ArrayIndexOutOfBoundsException exception) {
    exception.printStackTrace();
}
```



**Correct Answer: A**

**Section: JAVA only**

**Explanation**

**Explanation/Reference:****QUESTION 69**

For example, we have a file in our assets folder `app/src/main/assets/sample_tas.json`. To get an `InputStream` for reading it, from our `Context context`, we can try doing this:

A. `InputStream input = context.getResources().openRawResource(R.raw.sample_tas);`

B. `InputStream input = context.getAssets().open("sample_tas.json");`

C. `InputStream input = context.getResources().getAssets().open("sample_tas.json");`

**Correct Answer: B**

**Section: JAVA only**

**Explanation**

**Explanation/Reference:****QUESTION 70**

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

```
public MyViewModel(MyRepository myRepository)...
```

Next, in our `ViewModelFactory` create `ViewModel` method (overriden) looks like this:

```
@NonNull @Override
```

```
public <T extends ViewModel> T create(@NonNull Class<T> modelClass) {
```

```
try {
```

```
    //MISSED RETURN VALUE HERE
```

```
    } catch (InstantiationException | IllegalAccessException |
```

```
NoSuchMethodException | InvocationTargetException e) {
```

```
    throw new RuntimeException("Cannot create an instance of " + modelClass, e);
```

```
    }
```

```
}
```

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

A. `return modelClass.getConstructor().`

`.newInstance(mRepository);`

B. `return modelClass.getConstructor(MyRepository.class)`

`.newInstance();`

C. `return modelClass.getConstructor(MyRepository.class)`

`.newInstance(mRepository);`

**Correct Answer: C**

**Section: JAVA only**

**Explanation**

**Explanation/Reference:**



**QUESTION 71** What is demonstrated by the code below?

```
// RawDao.java
```

```
@Dao interface
```

```
RawDao {
```

```
    @RawQuery
```

```
    User getUserViaQuery(SupportSQLiteQuery query);
```

```
}
```

```
// Usage of RawDao
```

```
...
```

```
SimpleSQLiteQuery query =
```

```
    new SimpleSQLiteQuery("SELECT * FROM User WHERE id = ? LIMIT 1",
```

```
    new Object[]{userId});
```

```
User user = rawDao.getUserViaQuery(query);
```

```
...
```

A. A method in a Dao annotated class as a raw query method where you can pass the query as a `SupportSQLiteQuery`.

B. A method in a Dao annotated class as a query method.

C. A method in a `RoomDatabase` class as a query method.

**Correct Answer: A**

**Section: JAVA only**

**Explanation**

**Explanation/Reference:**

**QUESTION 72** What happens when you create a DAO method and annotate it with `@Insert`?

Example:

```
@Dao
public interface MyDao {
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    public void insertUsers(User... users); }
```

- A. Room generates an implementation that inserts all parameters into the database in a single transaction.
- B. Room modifies a set of entities, given as parameters, in the database. It uses a query that matches against the primary key of each entity.
- C. Room removes a set of entities, given as parameters, from the database. It uses the primary keys to find the entities to delete.

**Correct Answer:** A

**Section:** JAVA only

**Explanation**

**Explanation/Reference:**

### QUESTION 73

What do you want from Room when you create a DAO method and annotate it with `@Update`?

Example:

```
@Dao
public interface MyDao {
    @Update
    public void updateUsers(User... users);
}
```

- A. Room generates an implementation that inserts all parameters into the database in a single transaction.
- B. Room modifies a set of entities, given as parameters, in the database. It uses a query that matches against the primary key of each entity.
- C. Room removes a set of entities, given as parameters, from the database. It uses the primary keys to find the entities to delete.

**Correct Answer:** B

**Section:** JAVA only

**Explanation**

**Explanation/Reference:**

**QUESTION 74** What do you want from Room when you create a DAO method and annotate it with `@Delete`?

Example:

```
@Dao
public interface MyDao {
    @Delete
    public void deleteUsers(User... users);
}
```

- A. Room generates an implementation that inserts all parameters into the database in a single transaction.
- B. Room modifies a set of entities, given as parameters, in the database. It uses a query that matches against the primary key of each entity.
- C. Room removes a set of entities, given as parameters, from the database. It uses the primary keys to find the entities to delete.

**Correct Answer:** C

**Section:** JAVA only

**Explanation**

**Explanation/Reference:**

**QUESTION 75** In Android 8.0, API level 26, some APIs regarding notification behaviors were moved from Notification to NotificationChannel. For example, what should we use instead of NotificationCompat.Builder.setPriority() for Android 8.0 and higher?

- A. NotificationChannel.setPriority()
- B. NotificationChannel.setImportance()
- C. NotificationCompat.Builder.setImportance()

**Correct Answer:** B

**Section:** JAVA only

**Explanation**

**Explanation/Reference:**

Reference:

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

**QUESTION 76** What method should we use with Notification.Builder to supply a PendingIntent to be sent when the notification is clicked?

- A. setContentInfo
- B. setContentIntent
- C. setDeleteIntent

**Correct Answer:** B

**Section:** JAVA only

**Explanation**

**Explanation/Reference:**

Reference:

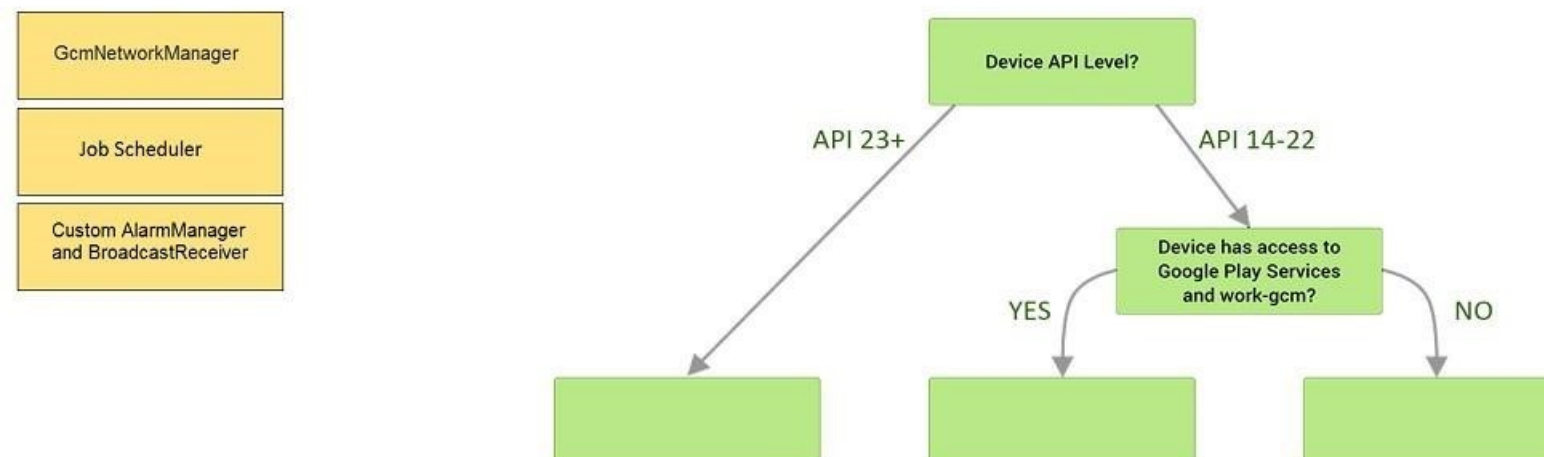
<https://developer.android.com/training/notify-user/build-notification>

**QUESTION 77**

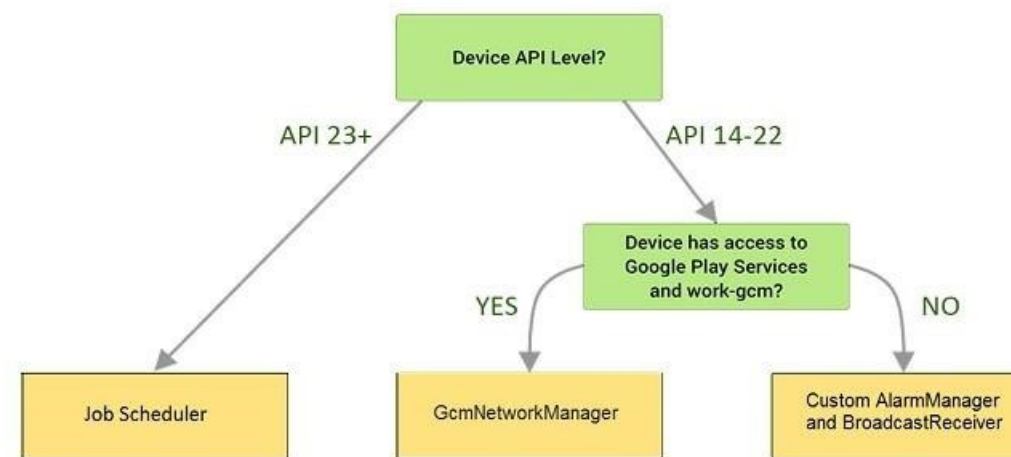
DRAG DROP

Under the hood WorkManager uses an underlying job dispatching service based on the following criteria. You need to move services to the correct places.

**Select and Place:**



**Correct Answer:**



#### Section: JAVA only

#### Explanation

#### Explanation/Reference:

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 78

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 enqueuing 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)

**Correct Answer:** ABCD

#### Section: JAVA only

#### Explanation

#### Explanation/Reference:

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 79

If you are working with a Builder that creates a `PeriodicWorkRequest` to run periodically once within the flex period of every interval period. What statement is correct?



- A. The repeat interval must be greater than PeriodicWorkRequest.MIN\_PERIODIC\_INTERVAL\_MILLIS and the flex interval must be greater than PeriodicWorkRequest.MIN\_PERIODIC\_FLEX\_MILLIS.
- B. The repeat interval must be lower than or equal to PeriodicWorkRequest.MIN\_PERIODIC\_INTERVAL\_MILLIS and the flex interval must be lower than or equal to PeriodicWorkRequest.MIN\_PERIODIC\_FLEX\_MILLIS.
- C. The repeat interval must be greater than or equal to PeriodicWorkRequest.MIN\_PERIODIC\_INTERVAL\_MILLIS and the flex interval can be anything in relation to PeriodicWorkRequest.MIN\_PERIODIC\_FLEX\_MILLIS.
- D. The repeat interval must be greater than or equal to PeriodicWorkRequest.MIN\_PERIODIC\_INTERVAL\_MILLIS and the flex interval must be greater than or equal to PeriodicWorkRequest.MIN\_PERIODIC\_FLEX\_MILLIS.

**Correct Answer:** D  
**Section:** JAVA only  
**Explanation**

**Explanation/Reference:**

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 80**

Custom duration in milliseconds as a parameter for the setDuration method is available when you are working with:

- A. Toast
- B. Snackbar
- C. for none of them
- D. for both of them

**Correct Answer:** B  
**Section:** JAVA only  
**Explanation**



**Explanation/Reference:**

Reference:

<https://developer.android.com/guide/topics/ui/notifiers/toasts>

<https://developer.android.com/training/snackbar/action>

**QUESTION 81**

If constant LENGTH\_INDEFINITE is used as a parameter for the setDuration method in Snackbar, what will happen?

- A. The Snackbar will be displayed for a short period of time.
- B. The Snackbar will be displayed for a long period of time.
- C. The Snackbar will be displayed for a very long period of time.
- D. The Snackbar will be displayed from the time that is shown until either it is dismissed, or another Snackbar is shown.
- E. The constant LENGTH\_INDEFINITE is impossible parameter for the setDuration method in Snackbar

**Correct Answer:** D  
**Section:** JAVA only  
**Explanation**

**Explanation/Reference:**

Reference:

[https://developer.android.com/reference/com/google/android/material/snackbar/BaseTransientBottomBar#LENGTH\\_INDEFINITE](https://developer.android.com/reference/com/google/android/material/snackbar/BaseTransientBottomBar#LENGTH_INDEFINITE)

<https://developer.android.com/guide/topics/ui/notifiers/toasts> <https://developer.android.com/training/snackbar/action>

**QUESTION 82** What public methods are there in android.widget.Toast.Callback? (Choose two.)

- A. onDismissed()
- B. onToastHidden()
- C. onShown()

- D. onToastShown()
- E. onToastCancelled()

**Correct Answer:** BD  
**Section:** JAVA only  
**Explanation**

**Explanation/Reference:**

Reference:

<https://developer.android.com/guide/topics/ui/notifiers/toasts>

<https://developer.android.com/training/snackbar/action>

**QUESTION 83** Which build options in the Build menu to choose to delete all intermediate/cached build files.

- A. Make Module
- B. Generate Signed Bundle / APK
- C. Rebuild Project
- D. Clean Project
- E. Make Project

**Correct Answer:** D  
**Section:** JAVA only  
**Explanation**

**Explanation/Reference:**

Reference:

<https://developer.android.com/studio/run>

**QUESTION 84** If you want get a debuggable APK that people can install without adb, in Android Studio you can:

- A. Select your debug variant and click Build Bundle(s) / APK(s) > Build APK(s).
- B. Click the Run button from toolbar
- C. Select your debug variant and click Analyze APK.

**Correct Answer:** A  
**Section:** JAVA only  
**Explanation**

**Explanation/Reference:**

Explanation:

The Run button builds an APK with testOnly="true", which means the APK can only be installed via adb (which Android Studio uses). If you want a debuggable APK that people can install without adb, select your debug variant and click Build Bundle(s) / APK(s) > Build APK(s). Reference:

<https://developer.android.com/studio/run>

**QUESTION 85**

To build a debug APK, you can open a command line and navigate to the root of your project directory. To initiate a debug build, invoke the `assembleDebug` task:

```
gradlew assembleDebug
```

This creates an APK named [module\_name]-debug.apk in [project\_name]/[module\_name]/build/outputs/apk/

Select correct statements about generated file. (Choose all that apply.)

- A. The file is already signed with the debug key
- B. The file is already aligned with zipalign
- C. You can immediately install this file on a device.



**Correct Answer:** ABC

**Section:** JAVA only

**Explanation**

**Explanation/Reference:**

Reference:

<https://developer.android.com/studio/run>

**QUESTION 86**

Building your app from the command line, if you have a "demo" product flavor, then you can build the debug version with the command:

- A. gradlew assembleDemoDebug
- B. gradlew installDemoDebug
- C. both variants are correct.

**Correct Answer:** C

**Section:** JAVA only

**Explanation**

**Explanation/Reference:**

Explanation:

Before immediately install build on a running emulator or connected device, installDemoDebug cause an APK building. Reference:

<https://developer.android.com/studio/run>

**QUESTION 87** If no any folder like *res/anim-<qualifiers>*, *res/drawable-<qualifiers>*, *res/layout-<qualifiers>*, *res/raw-<qualifiers>*, *res/xml-<qualifiers>* exist in the project. Which folders are required in the project anyway? (Choose two.)

- A. res/anim/
- B. res/drawable/
- C. res/layout/
- D. res/raw/
- E. res/xml/



**Correct Answer:** BC

**Section:** JAVA only

**Explanation**

**Explanation/Reference:**

Reference:

<https://developer.android.com/guide/topics/resources/localization>

**QUESTION 88**

Assume that an app includes a default set of graphics and two other sets of graphics, each optimized for a different device setup:

- res/drawable/  
Contains default graphics.
- res/drawable-small-land-stylus/  
Contains graphics optimized for use with a device that expects input from a stylus and has a QVGA low-density screen in landscape orientation.
- res/drawable-ja/  
Contains graphics optimized for use with Japanese.

What happens if the app runs on a device that is configured to use Japanese and, at the same time, the device happens to be one that expects input from a stylus and has a QVGA low-density screen in landscape orientation?

- A. Android loads graphics from res/drawable/
- B. Android loads graphics from res/drawable-small-land-stylus/
- C. Android loads graphics from res/drawable-ja/

**Correct Answer:** C

**Section:** JAVA only

**Explanation**

**Explanation/Reference:**

Reference:

<https://developer.android.com/guide/topics/resources/localization>

**QUESTION 89**

Assume that you have the following situation: •

The app code calls for R.string.text\_a

- Three relevant resource files are available:
  - res/values/strings.xml, which includes text\_a in the app's default language, in this case English.
  - res/values-mcc404/strings.xml, which includes text\_a in the app's default language, in this case English.-
  - res/values-hi/strings.xml, which includes text\_a in Hindi.
- The app is running on a device that has the following configuration:
  - The SIM card is connected to a mobile network in India (MCC 404). - The language is set to Hindi (hi).

Which is the correct statement below?

- A. Android loads text\_a from res/values/strings.xml (in English)
- B. Android loads text\_a from res/values-mcc404/strings.xml (in English)
- C. Android loads text\_a from res/values-hi/strings.xml (in Hindi)

**Correct Answer:** B

**Section:** JAVA only

**Explanation**

**Explanation/Reference:**

Explanation:

Android loads text\_a from res/values-mcc404/strings.xml (in English), even if the device is configured for Hindi. That is because in the resource-selection process, Android prefers an MCC match over a language match (as a priority Exception).

Reference:

<https://developer.android.com/guide/topics/resources/localization>

**QUESTION 90** What is the placeholder tag

<xliff:g> used for?

- A. To mark text that should not be translated.
- B. To raise a translation priority to a higher level
- C. To raise a quantity of translations for the string
- D. To pick up and move sting translation from a different resource file

**Correct Answer:** A

**Section:** JAVA only

**Explanation**

**Explanation/Reference:**

Reference:

<https://developer.android.com/guide/topics/resources/localization>

**QUESTION 91** Choose the most correct statement.

- A. Android is a closed source, Linux-based software stack created for a wide array of devices and form factors.
- B. Android is a closed source, Windows-based software stack created for a wide array of devices and form factors.
- C. Android is an open source, Linux-based software stack created for a wide array of devices and form factors.
- D. Android is an open source software stack created for a highly limited array of devices and form factors.

**Correct Answer:** C

**Section:** JAVA only

**Explanation**

**Explanation/Reference:**

Reference:

<https://developer.android.com/guide/platform>

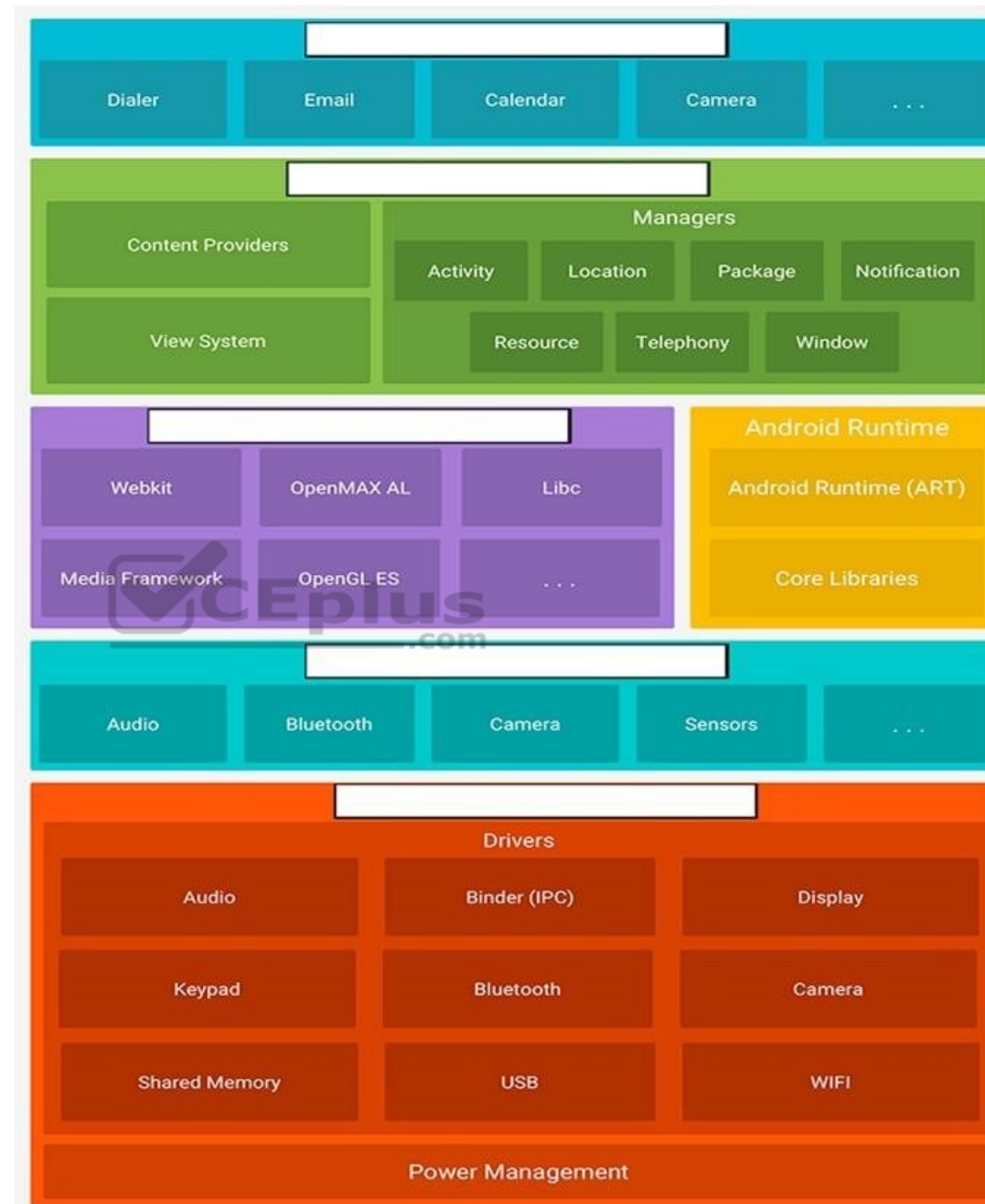
**QUESTION 92**

DRAG DROP

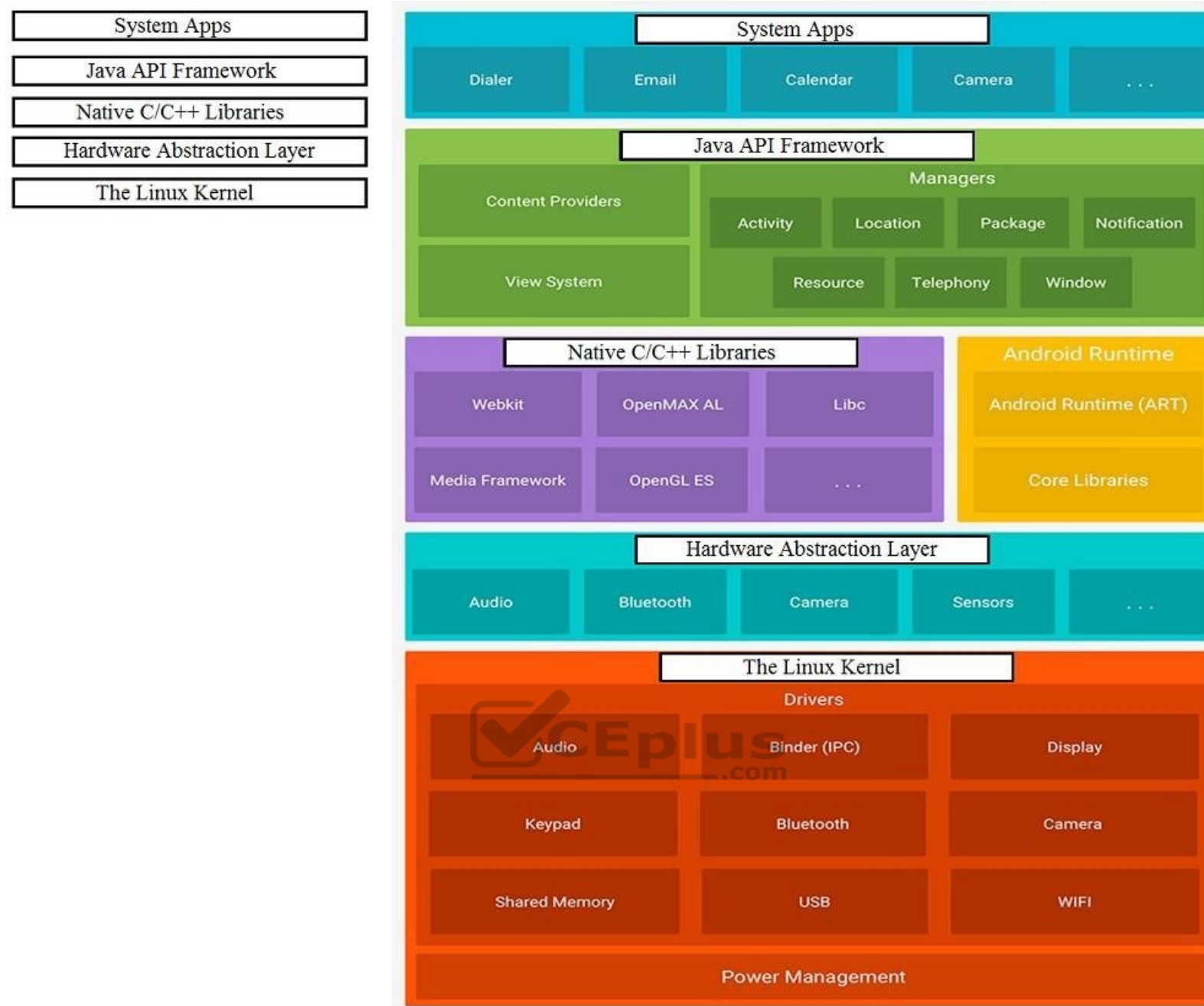
Move the major components of the Android platform to correct places in diagram.

**Select and Place:**

System Apps
Java API Framework
Native C/C++ Libraries
Hardware Abstraction Layer
The Linux Kernel



**Correct Answer:**



### Section: JAVA only

#### Explanation

#### Explanation/Reference:

Reference:

<https://developer.android.com/guide/platform>

**QUESTION 93** Select correct statements about Hardware Abstraction Layer (HAL). (Choose two.)

- A. The HAL provides standard interfaces that expose device hardware capabilities to the higher-level Java API framework.
- B. The HAL function both as apps for users and to provide key capabilities that developers can access from their own app. For example, if your app would like to deliver an SMS message, you don't need to build that functionality yourself – you can instead invoke whichever SMS app is already installed to deliver a message to the recipient you specify
- C. The HAL consists of multiple library modules, each of which implements an interface for a specific type of hardware component, such as the camera or bluetooth module. When a framework API makes a call to access device hardware, the Android system loads the library module for that hardware component.
- D. Using a HAL, not using a Linux kernel, allows Android to take advantage of key security features and allows device manufacturers to develop hardware drivers for a well-known kernel.

**Correct Answer:** AC

**Section: JAVA only****Explanation****Explanation/Reference:**

Explanation:

The system apps function both as apps for users and to provide key capabilities that developers can access from their own app. For example, if your app would like to deliver an SMS message, you don't need to build that functionality yourself — you can instead invoke whichever SMS app is already installed to deliver a message to the recipient you specify

Using a Linux kernel allows Android to take advantage of key security features and allows device manufacturers to develop hardware drivers for a well-known kernel. Reference:

<https://developer.android.com/guide/platform>

**QUESTION 94**

Custom views and directional controller clicks. On most devices, clicking a view using a directional controller sends (to the view currently in focus) a KeyEvent with:

- A. KEYCODE\_DPAD\_CENTER
- B. KEYCODE\_BUTTON\_START
- C. KEYCODE\_CALL
- D. KEYCODE\_BUTTON\_SELECT

**Correct Answer: A**

**Section: JAVA only****Explanation****Explanation/Reference:**

Reference:

<https://developer.android.com/guide/topics/ui/accessibility/custom-views>

**QUESTION 95**

Custom views and directional controller clicks. In general, you should send an AccessibilityEvent whenever the content of your custom view changes. For example, if a text value was changed in your custom view, you should emit an event of this type:

- A. TYPE\_WINDOWS\_CHANGED
- B. TYPE\_VIEW\_CONTEXT\_CLICKED
- C. TYPE\_WINDOWS\_CHANGED
- D. TYPE\_VIEW\_TEXT\_CHANGED

**Correct Answer: D**

**Section: JAVA only****Explanation****Explanation/Reference:**

Reference:

<https://developer.android.com/guide/topics/ui/accessibility/custom-views>

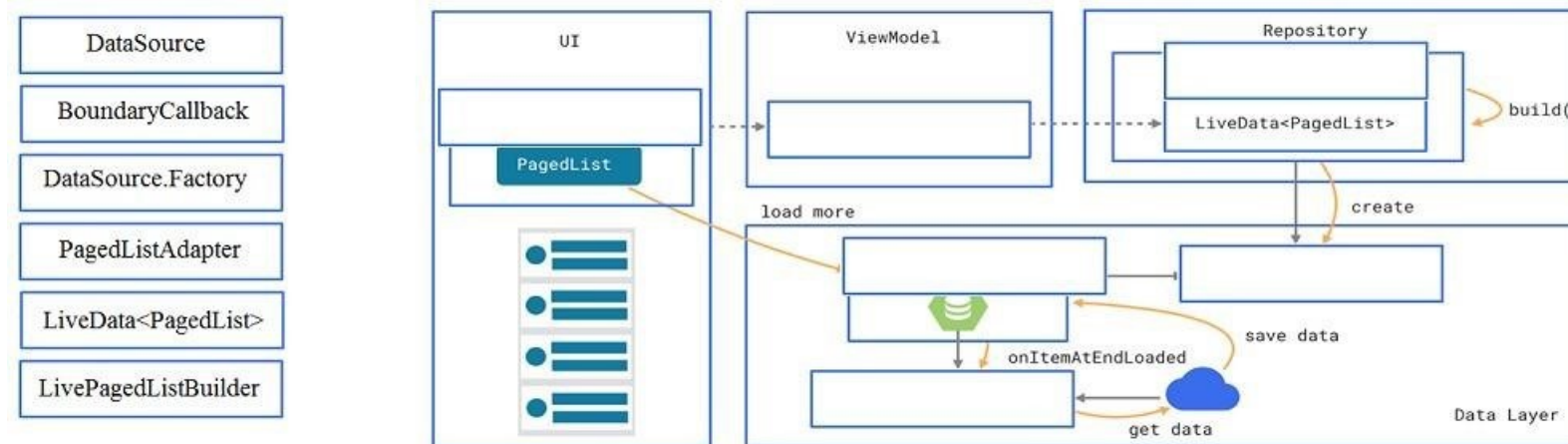
**QUESTION 96**

DRAG DROP

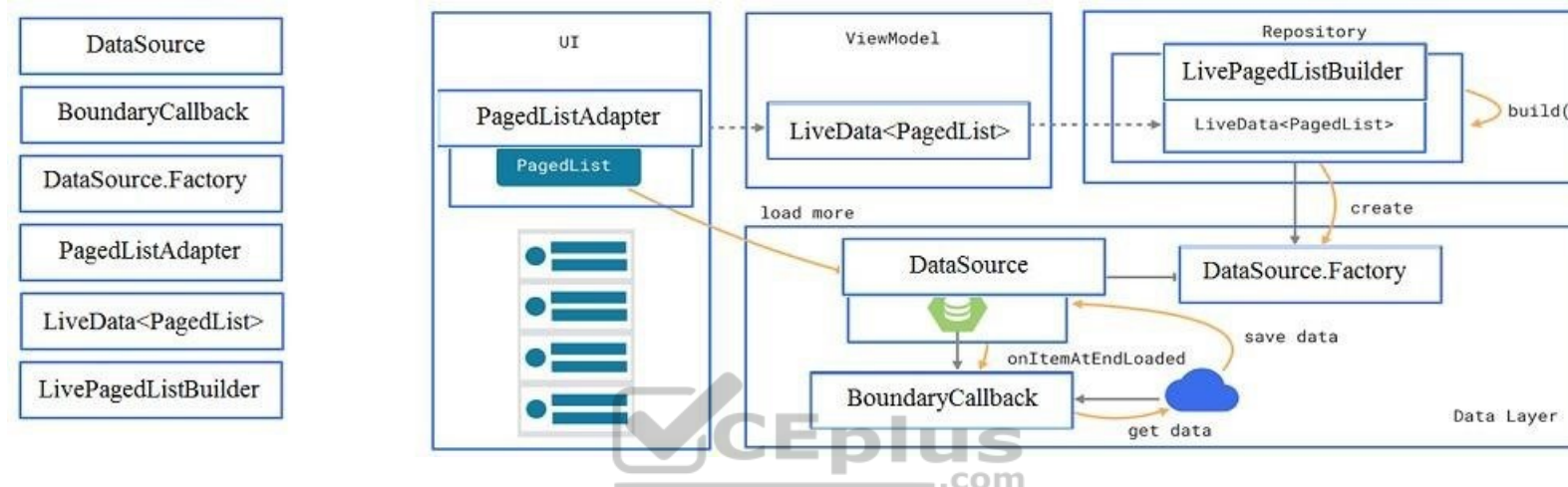
In a common Paging Library architecture scheme, move instances to the correct positions.

**Select and Place:**





**Correct Answer:**



**Section: JAVA only**

**Explanation**

**Explanation/Reference:**

Reference:

<https://developer.android.com/topic/libraries/architecture/paging/ui>

**QUESTION 97** Select a correct statement about PagedList.

- A. PagedList is content-mutable. This means that new content can be loaded into an instance of PagedList and the loaded items themselves can change once loaded.
- B. PagedList is content-immutable. This means that, although new content can be loaded into an instance of PagedList, the loaded items themselves cannot change once loaded.
- C. PagedList is content-accidental. This means that new content can be loaded into an instance of PagedList and the loaded items themselves can be changed to accidental values randomly.

**Correct Answer: B**

**Section: JAVA only**

**Explanation**

**Explanation/Reference:**

Reference:

<https://developer.android.com/topic/libraries/architecture/paging/ui>

**QUESTION 98** If content in a PagedList updates, the PagedListAdapter object receives:

- A. only one item from PagedList that contains the updated information.
- B. one or more items from PagedList that contains the updated information.



C. a completely new PagedList that contains the updated information.

**Correct Answer:** C

**Section:** JAVA only

**Explanation**

**Explanation/Reference:**

Reference:

<https://developer.android.com/topic/libraries/architecture/paging/ui>

#### QUESTION 99

Relative positioning is one of the basic building blocks of creating layouts in ConstraintLayout. Constraints allow you to position a given widget relative to another one. What constraints do not exist?

- A. layout\_constraintBottom\_toBottomOf
- B. layout\_constraintBaseline\_toBaselineOf
- C. layout\_constraintBaseline\_toStartOf
- D. layout\_constraintStart\_toEndOf

**Correct Answer:** C

**Section:** JAVA only

**Explanation**

**Explanation/Reference:**

Reference:

<https://developer.android.com/reference/androidx/constraintlayout/widget/ConstraintLayout>

**QUESTION 100** Which statement is most true about layout\_constraintLeft\_toRightOf and layout\_constraintStart\_toEndOf constraints ?

- A. layout\_constraintLeft\_toRightOf is equal to layout\_constraintStart\_toEndOf in any case
- B. layout\_constraintLeft\_toRightOf is equal to layout\_constraintStart\_toEndOf in case if user choose a language that uses right-to-left (RTL) scripts, such as Arabic or Hebrew, for their UI locale
- C. layout\_constraintLeft\_toRightOf is equal to layout\_constraintStart\_toEndOf in case if user choose a language that uses left-to-right (LTR) scripts, such as English or French, for their UI locale
- D. layout\_constraintLeft\_toRightOf works with horizontal axes and layout\_constraintStart\_toEndOf works with vertical axes

**Correct Answer:** C

**Section:** JAVA only

**Explanation**

**Explanation/Reference:**

Reference:

<https://developer.android.com/training/basics/supporting-devices/languages>

**QUESTION 101** In application theme style, flag windowNoTitle (<item name="windowNoTitle">) indicates:

- A. whether this window should have an Action Bar in place of the usual title bar.
- B. whether there should be no title on this window.
- C. that this window should not be displayed at all.
- D. whether this is a floating window.
- E. whether this Window is responsible for drawing the background for the system bars.

**Correct Answer:** B

**Section:** JAVA only

**Explanation**

**Explanation/Reference:**

Reference:

<https://developer.android.com/guide/topics/ui/look-and-feel/themes> <https://developer.android.com/reference/android/R.styleable.html>

**QUESTION 102** "Set the activity content to an explicit view. This view is placed directly into the activity's view hierarchy. It can itself be a complex view hierarchy." This can be done by calling method:

- A. findViewById
- B. setContentView
- C. setActionBar
- D. setContentTransitionManager
- E. setTheme

**Correct Answer:** B

**Section:** JAVA only

**Explanation**

**Explanation/Reference:**

Reference:

<https://developer.android.com/training/basics/firstapp/building-ui>

<https://developer.android.com/reference/android/app/Activity>

**QUESTION 103**

A content label sometimes depends on information only available at runtime, or the meaning of a View might change over time. For example, a Play button might change to a Pause button during music playback. In these cases, to update the content label at the appropriate time, we can use:

- A. View#setContentDescription(int contentDescriptionResId)
- B. View#setContentLabel(int contentDescriptionResId)
- C. View#setContentDescription(CharSequence contentDescription)
- D. View#setContentLabel(CharSequence contentDescription)

**Correct Answer:** C

**Section:** JAVA only

**Explanation**



**Explanation/Reference:**

Reference:

<https://support.google.com/accessibility/android/answer/7158690?hl=en>

**QUESTION 104**

When using an ImageView, ImageButton, CheckBox, or other View that conveys information graphically. What attribute to use to provide a content label for that View?

- A. android:contentDescription
- B. android:hint
- C. android:labelFor

**Correct Answer:** A

**Section:** JAVA only

**Explanation**

**Explanation/Reference:**

Reference:

<https://support.google.com/accessibility/android/answer/7158690?hl=en>

**QUESTION 105**

When using an EditText or editable TextViews, or other editable View. What attribute to use to provide a content label for that View?

- A. android:contentDescription
- B. android:hint
- C. android:labelFor

**Correct Answer:** B  
**Section:** JAVA only  
**Explanation**

**Explanation/Reference:**

Reference:

<https://support.google.com/accessibility/android/answer/7158690?hl=en>

**QUESTION 106** Content labels. What attribute to use to indicate that a View should act as a content label for another View?

- A. android:contentDescription
- B. android:hint
- C. android:labelFor

**Correct Answer:** C  
**Section:** JAVA only  
**Explanation**

**Explanation/Reference:**

Reference:

<https://support.google.com/accessibility/android/answer/7158690?hl=en>

**QUESTION 107** In application theme style, flag windowActionBar (<item name="windowActionBar">) indicates:

- A. whether the given application component is available to other applications.
- B. whether action modes should overlay window content when there is not reserved space for their UI (such as an Action Bar).
- C. whether this window's Action Bar should overlay application content.
- D. whether this window should have an Action Bar in place of the usual title bar.



**Correct Answer:** D  
**Section:** JAVA only  
**Explanation**

**Explanation/Reference:**

Reference:

<https://developer.android.com/guide/topics/ui/look-and-feel/themes>

<https://developer.android.com/reference/android/R.styleable.html>