

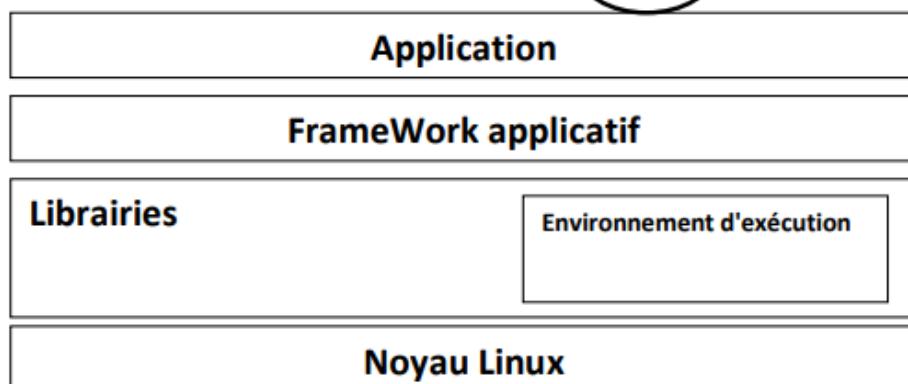
Questions de cours

1. Donner l'architecture en **couche du système Android**, expliquer brièvement le rôle de chaque couche.

Réponse :

- a. L'architecture du système Android est la suivante :

0.75



- b. Le rôle de chaque couche est le suivant :

0.75

Noyaux Linux : c'est la couche qui s'occupe, notamment, de la gestion du matériel à travers des pilotes.

Librairies: elle fournit un ensemble de fonctionnalité à travers un est un ensemble de bibliothèques :WebKit, libc SQLite , etc.

Environnement d'exécution : cette couche contient les librairies coeurs du Framework ainsi que la machine virtuelle exécutant les applications.

Framework Applicatif : il permet au développeur de créer des applications en utilisant des composants prêt à l'emploi.

Applications : c'est la seule couche que voit l'utilisateur, on trouve des applications soit natives ou installées par l'utilisateur.

2. Expliquer brièvement, le rôle des outils suivants : **ADT**, **ADB** et **AAPT**

0.75

Réponse :

ADT : il simplifie le développement Android. Il intègre les outils de développement comme l'émulateur sous eclipse.

AAPT: *Android Asset Packaging Tool*. Le rôle de cet outil est de compiler les ressources des applications Android et il produit la classe R.

ADB: *Android Debug Bridge*. il permet la communication avec l'émulateur.

3. Quels sont les sont **les composants d'une application Android** et quel est leur rôle.

Réponse : Les composants d'une application Android sont les suivants :

0.75

Activity : est le composant principal d'une application Android, il offre une interface graphique pour l'utilisateur (interaction).

Service : est un composant qui ne dispose pas d'interface, il s'exécute en arrière plan.

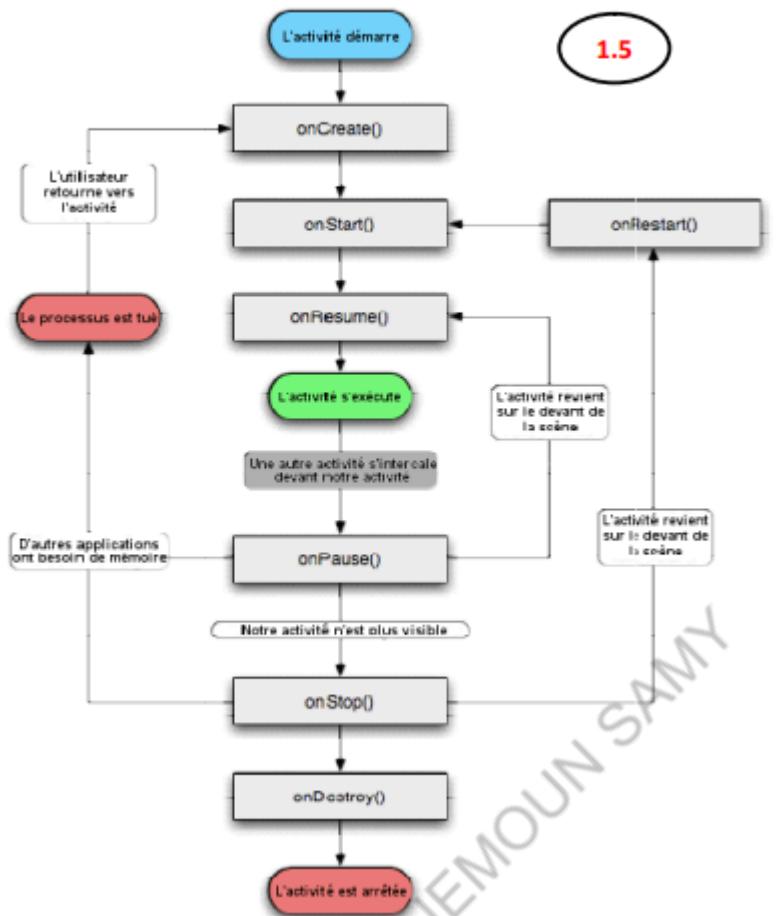
ContentProvider : est un composant qui permet de partager des données entre applications.

Intent : est un composant qui permet le lancement d'autres composants Android (Activity, Service et BroadcastReceiver) et il permet aussi de communiquer des données entre composants.

BroadcastReceiver : est une composant qui est à "l'écoute" des autres applications. Ce type de composant tente de répondre au Intent qui lui sont adressés.

4. Donner le diagramme de cycle de vie d'une activité.

Réponse : Le cycle de vie d'une activité Android est le suivant :



5. Quels sont les rôles de fichier **AndroidManifest.xml** et de la classe **R**.

Réponse : Les rôles de fichier **AndroidManifest.xml** et de la classe **R** sont les suivants :

Les fichiers **AndroidManifest.xml** : son rôle est de contenir les informations sur l'application :

- Version SDK utilisée
- Permissions
- Les composants de l'application: Activity, Service, etc.
- etc.

0.75

Il est utilisé par le système "Android" pour déterminer les capacités de l'application.

La classe **R** : elle contient les identificateurs des ressources de l'application : **layout**, **drawable**, etc.

Elle est utilisée pour accéder aux ressources via le code java de l'application.

6. Expliquer la relation entre un **ContentProvider** et un **ContentResolver**.

Réponse :

Le **ContentProvider** est un composant qui se trouve au niveau de l'application fournisseur de données. Il partage des données via des méthodes : **insert()**, **query()**, etc.

1

Par contre le **ContentResolver** est un objet qui est invoqué au niveau de l'application cliente (consommatrice de données). L'objet **ContentResolver** utilise les mêmes noms de méthodes que celles de **ContentProvider** (**insert()**, **query()**, etc.). Pour établir la connexion entre un **ContentProvider** et un **ContentResolver**, ce dernier doit fournir au système l'**URI** de **ContentProvider** et l'**action** demandée (méthode). Ce fonctionnement assure une meilleure sécurité.

7. Expliquer brièvement, les **deux modes d'émission** d'un intent vers un **BroadcastReceiver**.

Réponse : il existe deux modes d'émission :

- **Classique** : tous les BroadcastReceiver ont la même priorité.
- **Ordonné** : les BroadcastReceiver ont des priorités différentes, donc c'est le plus prioritaire qui reçoit d'abord l'intent, il peut l'annuler, comme il peut le passer au BroadcastReceiver suivant.

0.75

Questions à choix multiples(4 points) :

1. Quel est le nom du dossier qui contient le fichier R.java ? A) src; B) res; C) bin ;D) gen	2. Lequel des éléments suivants n'est pas activé par un intent? A) Activity, B) Service , C) Broadcast receiver D) SQLite DB Connection.
3. Quelle est la classe parente de tout les composants graphiques? A) ViewGroup; B) Layout; C) Widget; D) View.	4. Quel est le nom de la classe utilisée par un intent pour stocker les informations additionnelles? A) Extra ; B)Bundle; C)Parceble; D) Aucune
5. Lors de la création d'un fichier à l'aide <code>openFileOutput("test.txt", 0)</code> , le fichier est créé dans : A) /data/app/<Nom package>/files B) /data/data/<Nom package>/files C) /system/app/<Nom package>/files	6. Quelle méthode d'activité que vous utilisez pour récupérer une référence à une vue Android en utilisant l'attribut id? A) findViewByIdByReference(int id); B) findViewById(int id) C) retrieveResourceById(int id) D) findViewById(String id)
7. Que contient le dossier src ? A) Images et icônes; B) Fichiers XML; C) Le fichier Manifest; D) Les fichiers sources java	8. Quelle est la méthode qui est appelée lorsque l'activité devient non visible ? A) onPause(); B) onCreate(); C) onStop(); D) onHide();

Réponses :

Numéro question	Réponse
1	D
2	D
3	D
4	B
5	B
6	B
7	D
8	C

4

Questions de cours (8 points):

1. Quelle est la différence entre les ressources de type *assets* et les autres ressources?

0.75

Réponse :

La différence entre les deux types de ressources est : les ressources assets ne sont pas compilées par contre les autres ressources le sont. Par conséquent, un identifiant, pour chaque ressources (les autres ressources), est généré dans la classe R. Cet identifiant est un moyen utilisé pour accéder à cette ressource. Exemple d'accès : `findViewById(R.TypeR.XXX)`; Où XXX est l'identifiant d'une ressource, TypeR : type de la ressource.

2. Pourquoi, à votre avis, l'exécution de la méthode *onPause()* doit être rapide?

0.5

Réponse :

La méthode `onPause()` est exécutée lorsqu'une autre activité demande de passer au premier plan. Tant que cette méthode n'est pas terminée, l'autre activité ne peut être au premier plan. Donc pour plus de fluidité, il est recommandé de ne pas mettre beaucoup de traitements au niveau de cette méthode.

3. Expliquer le rôle des deux méthodes : `onSaveInstanceState` et `onRestoreInstanceState`.

Réponse :

Le système (Android) détruit les activités dans deux cas particuliers:

1

- Changement d'orientation de l'écran (le système détruit et recrée l'activité)
- L'activité en cours d'exécution nécessite plus de mémoire. Alors le système libère la mémoire occupée par les activités qui se trouvent en arrière plan (`onStop`, `onPause`).

Dans ces deux cas, le système offre deux méthodes pour sauvegarder et restaurer l'état des activités détruites par le système.

`onSaveInstanceState(Bundle savedInstanceState)` : est appelée après la méthode `onPause()`. Son rôle est de sauvegarder l'état de l'interface graphique (super, `onSaveInstanceState()`,) et les données temporaires de l'activité. Cet état est sauvegardé dans le paramètre, de type **Bundle**, de la méthode.

`onRestoreInstanceState(Bundle savedInstanceState)` : est appelée après la méthode `onStart()`. Son rôle est de restaurer l'état de l'interface graphique (super, `onRestoreInstanceState()`,) et les données temporaires de l'activité. Cet état est restauré à partir du paramètre de la méthode qui est de type **Bundle**.

4. Quelle est la différence entre `OnClickListener` et `OnTouchListener` ?

0.75

Réponse :

Pour chaque type d'événement Android offre une interface à implementer.

`OnClickListener` est une interface correspondante aux événements de type clic (`onClick`).

`OnTouchListener` est une interface correspondante aux événements de type mouvement (`onTouch`).

La différence principale entre ces deux types d'événements est que `onClick` est utilisé pour des besoins basiques (clic : l'événement est pris en entier) par contre `onTouch` permet de fractionner le mouvement en plusieurs événements (presser, relâcher, etc.)

5. Expliquer la différence entre : "`FLAG_ACTIVITY_NEW_TASK`" et "`FLAG_ACTIVITY_SINGLE_TOP`".

0.75

Réponse :

Le champ Flags est champ d'un Intent, il est utilisé pour informer Android comment lancer un composant et comment le traiter une fois exécuté.

Si la valeur de cette information est égale à :

`FLAG_ACTIVITY_NEW_TASK` : alors Android lance l'activité demandée dans une nouvelle tâche, sauf si cette dernière existe déjà dans une autre tâche.

Par contre dans le cas où elle es égale à `FLAG_ACTIVITY_SINGLE_TOP` Android lance l'activité demandée dans une nouvelle tâche quelque soit les circonstances.

6. Expliquer la différence entre : "`CATEGORY_HOME`" et "`CATEGORY_LAUNCHER`".

1

Réponse :

La catégorie (category) d'un intent est une chaîne (String) qui spécifie le type de composant demandé.

Si elle à la valeur :

- `CATEGORY_HOME` : alors l'activité en question appartient à une application de type lanceur d'application. Donc c'est elle qui doit se trouver sur l'écran d'accueil d'Android.
- `CATEGORY_LAUNCHER`: alors elle indique que c'est ce composant qui doit s'afficher dans le lanceur d'applications.

7. Expliquer le rôle des classes suivantes : *SQLiteOpenHelper* et *SQLiteDatabase*.

Réponse :

SQLiteOpenHelper : est une classe abstraite qui offre des méthodes (à implémenter, pas toutes) pour la gestion de la base de données :

- La création de la bases : `onCreate(SQLiteDatabase db)`
- La mise à jours de la base de donnée : `onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)`
- L'ouverture de la base de données : `getWritableDatabase()` (en écriture)
`getReadableDatabase()` (en lecture)

1.5

SQLiteDatabase : Cette classe offre des méthode de manipulation de la base de données :

- Création : `execSQL`
- Insertion des données : `insert`
- Sélection des données : `rawQuery`

8. Quels est le rôle des classes suivantes: *InputStream*, *InputStreamReader* et *BufferedReader*.

0.75

Réponse :

InputStream : La classe de base des flux d'entrée d'octets.

InputStreamReader : permet de transformer le flux d'entrée (*InputStream*) d'octets en flux de caractères.

BufferedReader : permet d'accélérer la lecture d'un flux d'entrée; la lecture est réalisée à partir d'une mémoire tampon (buffer).

9. Quelles sont les étapes nécessaires pour accéder aux fichiers de la carte SD?

Réponse :

L'accès aux fichiers de la carte SD est réalisé comme suit :

1

1. Spécifier le chemin de la carte SD, en le récupérant avec l'une des méthodes suivantes :

- `getExternalFilesDir()` - *activity ou Context*-
- `getExternalStoragePublicDirectory()` - *Environnement*-

2. Déclarer les permissions dans le fichier AndroidManifest :

- `android.permission.READ_EXTERNAL_STORAGE`
- `android.permission.WRITE_EXTERNAL_STORAGE`

Questions à choix multiples (3 points) :

1. Quel est le plugin d'Eclipse nécessaire pour développer une application Android? A) J2EE B) Android Software Development Kit C) Android Development Tools D) Web Development Tools	2. Qu'est-ce qui n'est pas vrai à propos d'AndroidManifest.xml? A) Il déclare les vues utilisées dans l'application. B) Il déclare les composants de l'application. C) Il déclare les permissions requises. D) Il déclare les ressources de l'application.
3. Laquelle des méthodes suivantes est appelée dans une activité lorsqu'une autre activité passe au premier plan? A) <code>onPause()</code> B) <code>onCreate()</code> C) <code>onStop()</code> D) <code>onHide()</code>	4. Quelle est la méthode à utiliser pour ouvrir un fichier raw A) <code>openFileInput(String NomFichier)</code> B) <code>openRawResource(int idFichier)</code> C) <code>openRawFile(int idFichier)</code> D) <code>openRawFile(String NomFichier)</code>
5. Quel est l'élément d'information nécessaire pour définir un intent explicite? A) Une action B) Des données C) A et B D) Le nom de composant	6. Quel est l'emplacement recommandé pour un fichier ressource de type texte? A)res/anim B)res/files C) res/raw D)res/values

1. Donner la structure d'un projet Android. Expliquer, brièvement, le rôle de chaque dossier du projet.
2. Expliquer le processus de construction d'une application Android (de la compilation à l'installation).
3. Quels sont les principaux conteneurs (ViewGroup) prédéfinis dans Android, expliquer le fonctionnement global de trois conteneurs.
4. Définir la notion d'intent, quelles informations contient 'il?
5. Expliquer le processus de résolution d'un intent implicite?
6. quelles sont les techniques de stockage de données offertes par Android, expliquer brièvement leur principe?

1. Structure d'un projet Android :

- **app/src/main/java** : Contient les fichiers source de l'application, y compris les activités, les services, etc.
- **app/src/main/res** : Contient les ressources de l'application, telles que les fichiers de mise en page XML, les images, les chaînes de texte, etc.
- **app/src/main/AndroidManifest.xml** : Fichier de manifeste qui déclare les composants de l'application et fournit des informations sur l'application au système Android.

2. Processus de construction d'une application Android :

- Compilation des fichiers source en bytecode dex.
- Empaquetage des fichiers compilés, des ressources et du manifeste dans un fichier APK.
- Installation de l'APK sur un appareil Android à l'aide d'ADB (Android Debug Bridge) ou d'un gestionnaire de package sur le dispositif.

3. Principaux conteneurs ViewGroup :

- **LinearLayout** : Organise les enfants dans une seule direction, horizontale ou verticale.
- **RelativeLayout** : Place les enfants relativement les uns aux autres ou aux bords du conteneur parent.
- **ConstraintLayout** : Permet de créer des mises en page complexes en utilisant des contraintes entre les vues.

4. Intent : Un Intent est un objet utilisé pour communiquer entre les composants de l'application ou entre différentes applications. Il contient des informations telles que l'action à effectuer, les données à manipuler, le composant de destination, etc.

5. Résolution d'un intent implicite : Le système Android recherche dynamiquement les applications capables de traiter l'intent implicite en fonction de l'action spécifiée et des données associées. Il choisit ensuite l'application la plus appropriée pour gérer l'intent.

6. Techniques de stockage de données :

- **SharedPreferences** : Stocke des paires clé-valeur de données primitives sous forme de fichiers XML.
- **Fichiers** : Stocke des données dans des fichiers locaux sur le système de fichiers de l'appareil.

- **Base de données SQLite** : Stocke des données structurées dans une base de données relationnelle embarquée.

1. Quel est le fichier qui spécifie la version minimale du SDK requise pour exécuter votre application? A) version.xml; B) R.java; C) strings.xml ; D) AndroidManifest.xml	2. Quel est le fichier qui contient des chaînes de caractères que vous pouvez utiliser dans votre application? A) AndroidManifest.xml ; B) res/Text.xml; C) res/layout/Main.xml ; D) res/values/strings.xml
3. Quelle est la méthode utilisée pour fermer une activité? A) destroy(); B) finish(); C) stop(); D) close().	4. Que fait la méthode StartActivity? A) Lance une nouvelle activité et met la précédente en arrière plan. B) Lance une nouvelle activité et détruit la précédente.
5. Quels sont les éléments d'information nécessaires pour définir un intent implicite? A) Une action ; B) Des données C) A) et B) ; D) Aucun	6. Quel est le composant non disponible dans la couche android application framework ? A) WindowManager ; B) NotificationManager C) DialerManager ; D) PackageManager
7. Quelle est la méthode invoquée lorsqu'on clique sur une vue (view) ? A) OnClick; B) OnTapListener; C) OnClickDetector; D) OnClickListener	8. Quel est l'outil qui gère les messages log? A) DDMS; B) Logcat; C) ADB; D) LogTool;

1. **D) AndroidManifest.xml**
2. **D) res/values/strings.xml**
3. **B) finish()**
4. **A) Lance une nouvelle activité et met la précédente en arrière-plan.**
5. **C) A) et B) (Une action et des données)**
6. **C) DialerManager**
7. **D) OnClickListener**
8. **B) Logcat**

1. Expliquer le processus de compilation sous Android.
2. Expliquer les notions d'application, d'activité, de tâche et de processus sous Android.
3. Expliquer la différence entre les deux mesures : pixel (px) et dp.
4. Expliquer le fonctionnement du conteneur RelativeLayout.
5. Expliquer le processus de lecture et d'écriture dans une préférence partagée.
6. Quand est ce que l'exception "ActivityNotFoundException" est levée?
7. Expliquer la différence entre un intent explicite sans résultat et avec résultat.

1. Processus de compilation sous Android

Le processus de compilation d'une application Android comporte plusieurs étapes :

1. **Compilation des fichiers source Java en bytecode (.class) :**
 - o Les fichiers `.java` sont compilés en fichiers `.class` à l'aide du compilateur `javac`.
2. **Conversion des fichiers .class en fichiers .dex (Dalvik Executable) :**
 - o Les fichiers `.class` sont ensuite convertis en un ou plusieurs fichiers `.dex` (Dalvik Executable) via l'outil `dx`. Android Runtime (ART) et Dalvik Virtual Machine exécutent ces fichiers `.dex`.
3. **Compilation des fichiers de ressources :**
 - o Les fichiers de ressources XML sont compilés en fichiers binaires `.arsc` par l'outil `aapt` (Android Asset Packaging Tool).
4. **Génération du fichier APK :**
 - o Les fichiers compilés, les fichiers de ressources et le fichier `AndroidManifest.xml` sont empaquetés dans un fichier APK (`.apk`) à l'aide de l'outil `aapt`.
5. **Signature de l'APK :**
 - o Le fichier APK est signé avec une clé de signature, soit une clé de débogage pour les versions de développement, soit une clé de signature de production pour les versions de production.
6. **Alignment de l'APK :**
 - o Enfin, le fichier APK est optimisé (aligné) pour un accès plus rapide aux ressources en utilisant l'outil `zipalign`.

2. Notions d'application, d'activité, de tâche et de processus sous Android

- **Application :**
 - o Un ensemble de composants (activités, services, fournisseurs de contenu, récepteurs de diffusion) définis dans un fichier `AndroidManifest.xml`. Une application correspond à un package Android (`.apk`).
- **Activité :**
 - o Une composante représentant une seule interface utilisateur. Une application peut avoir plusieurs activités, et une activité correspond généralement à un écran de l'application.
- **Tâche (Task) :**
 - o Une séquence d'activités que les utilisateurs interagissent avec pour accomplir une action. Une tâche peut inclure des activités de différentes applications.
- **Processus :**
 - o Une instance de l'application en cours d'exécution. Chaque application Android s'exécute par défaut dans son propre processus Linux isolé.

3. Différence entre pixel (px) et dp

- **Pixel (px) :**

- Une unité de mesure physique basée sur la densité de l'écran. 1 px représente un point sur l'écran, et la taille en pixels peut varier en fonction de la densité de l'écran.
- **Density-independent pixel (dp) :**
 - Une unité abstraite indépendante de la densité de l'écran. 1 dp est approximativement égal à 1 pixel sur un écran de densité de référence (160 dpi). Utiliser dp permet de créer des interfaces utilisateur cohérentes sur différents écrans avec des densités variées.

4. Fonctionnement du conteneur **RelativeLayout**

RelativeLayout est un ViewGroup qui permet de positionner ses éléments enfants de manière relative les uns par rapport aux autres ou par rapport à son parent :

- Les vues peuvent être alignées par rapport au parent (`alignParentTop`, `alignParentBottom`, etc.).
- Les vues peuvent être alignées par rapport à d'autres vues (`layout_below`, `layout_toRightOf`, etc.).
- Les vues peuvent être centrées dans le parent (`centerInParent`, `centerHorizontal`, etc.).

5. Processus de lecture et d'écriture dans une préférence partagée

- **Écriture :**

```
SharedPreferences preferences =
getSharedPreferences("nom_preferences", MODE_PRIVATE);

SharedPreferences.Editor editor = preferences.edit();

editor.putString("clé", "valeur");

editor.apply(); // Ou commit() pour une écriture synchrone
```

Lecture :

```
SharedPreferences preferences =
getSharedPreferences("nom_preferences", MODE_PRIVATE);

String valeur = preferences.getString("clé", "valeur_par_défaut");
```

6. Quand est-ce que l'exception "ActivityNotFoundException" est levée ?

L'exception `ActivityNotFoundException` est levée lorsqu'un Intent explicite ou implicite ne trouve pas d'activité correspondante pour traiter l'Intent. Cela peut se produire si aucune application n'est installée pour gérer l'action spécifiée dans l'Intent.

7. Différence entre un intent explicite sans résultat et avec résultat

- **Intent explicite sans résultat :**

- Utilisé pour démarrer une activité sans attendre de réponse. Par exemple :

```
Intent intent = new Intent(this, AutreActivité.class);  
startActivity(intent);
```

Intent explicite avec résultat :

- Utilisé pour démarrer une activité et attendre une réponse (données) en retour. Par exemple :

```
Intent intent = new Intent(this, AutreActivité.class);  
startActivityForResult(intent, REQUEST_CODE);
```

La méthode `onActivityResult(int requestCode, int resultCode, Intent data)` est appelée lorsque l'activité se termine pour traiter le résultat.

1. C) **Manifest**
2. A) **SQLite** et C) **Layout**
3. B) **onCreate()**
4. B) **Intent Filters**
5. B) **Cursor**
6. B) **Layout**
7. B) **Content Provider**
8. C) **ADB**

1. Expliquer, brièvement, la différence entre la JVM et la Dalvik VM.
2. Expliquer le rôle de l'ActivityManager.
3. Citer les différentes couches du système Android, puis expliquer les interactions entre ces couches pour l'exécution d'une application.
4. Expliquer le rôle des méthodes `onResume()` et `onPause()`, quelle est la relation existante entre ces deux méthodes?
5. Expliquer le fonctionnement d'un `LinearLayout`, préciser le rôle de l'attribut : `android:layout_weight`.
6. Expliquer le rôle de l'attribut catégorie (`category`) dans un intent, préciser la différence entre les deux valeurs suivantes : `CATEGORY_HOME` et `CATEGORY_LAUNCHER`.

7. Citer et expliquer les étapes de manipulation d'un fichier Raw.

1. Différence entre la JVM et la Dalvik VM

- **JVM (Java Virtual Machine) :**
 - Exécute le bytecode Java standard (fichiers `.class`).
 - Chaque application s'exécute dans sa propre instance de JVM.
 - Conçu pour une large gamme de plateformes, avec une gestion mémoire et un modèle de threading adaptés à des systèmes plus puissants.
- **Dalvik VM :**
 - Conçu spécifiquement pour Android.
 - Utilise un bytecode différent (fichiers `.dex` - Dalvik Executable).
 - Plusieurs applications partagent la même instance de la Dalvik VM, mais chaque application s'exécute dans un processus isolé.
 - Optimisé pour les appareils à mémoire limitée et à processeur moins puissant.

2. Rôle de l'ActivityManager

- **ActivityManager :**
 - Gère le cycle de vie des activités dans une application Android.
 - Supervise la création, l'arrêt, la reprise et la destruction des activités.
 - Gère les tâches et les piles d'activités, assurant ainsi le bon fonctionnement de la navigation et de la gestion des ressources.

3. Couches du système Android

- **Couches du système Android :**
 - **Applications** : Les applications utilisateur.
 - **Framework d'applications** : Fournit des API pour les applications.
 - **Bibliothèques Android** : Incluent les bibliothèques C/C++ essentielles utilisées par les différents composants du système Android.
 - **Android Runtime** : Comprend la Dalvik VM et les bibliothèques de base.
 - **Linux Kernel** : Gère la sécurité, la gestion de la mémoire, la gestion des processus, etc.
- **Interactions entre les couches :**
 - Les applications utilisent le framework d'applications pour accéder aux composants du système.
 - Le framework d'applications repose sur les bibliothèques Android pour les fonctionnalités de bas niveau.
 - La runtime Android exécute le code des applications sur la Dalvik VM.
 - Le noyau Linux gère les interactions avec le matériel et les ressources système.

4. Rôle des méthodes onResume() et onPause()

- **onResume()** :
 - Appelée lorsque l'activité entre au premier plan et devient interactive pour l'utilisateur.
 - Utilisée pour démarrer ou redémarrer des tâches qui doivent s'exécuter en premier plan.
- **onPause()** :
 - Appelée lorsque l'activité perd le focus, mais reste partiellement visible (par exemple, lorsqu'une nouvelle activité partiellement transparente est lancée).
 - Utilisée pour suspendre ou ajuster les opérations sensibles au cycle de vie, comme arrêter les animations, libérer des ressources.
- **Relation entre onResume() et onPause()** :
 - Ces méthodes marquent les transitions d'une activité entre les états actifs et non actifs. `onPause()` est toujours appelée avant `onResume()` lorsque l'activité revient en avant-plan.

5. Fonctionnement d'un LinearLayout et rôle de l'attribut android:layout_weight

- **LinearLayout** :
 - Organise les enfants en une seule ligne, soit horizontale soit verticale.
 - Les vues enfants sont disposées les unes après les autres.
- **android:layout_weight** :
 - Définit l'importance relative d'une vue dans le LinearLayout.
 - Une vue avec un `layout_weight` plus élevé occupe plus d'espace que les autres vues proportionnellement.

6. Rôle de l'attribut catégorie (category) dans un intent

- **Attribut catégorie (category)** :
 - Spécifie des informations additionnelles sur le type d'action à effectuer par l'Intent.
 - Utilisé pour définir le contexte dans lequel l'Intent doit être traité.
- **Différence entre CATEGORY_HOME et CATEGORY_LAUNCHER** :
 - **CATEGORY_HOME** : Indique que l'activité est une activité de type Home, c'est-à-dire l'écran d'accueil par défaut.
 - **CATEGORY_LAUNCHER** : Indique que l'activité doit être listée dans le lanceur d'applications et peut être lancée depuis l'écran d'accueil.

7. Étapes de manipulation d'un fichier Raw

1. **Placer le fichier dans le répertoire res/raw** :
 - Copier le fichier dans le dossier `res/raw` de votre projet Android.
2. **Accéder au fichier depuis le code** :

- Utiliser l'ID généré automatiquement pour accéder au fichier.

3. Lecture du fichier :

```
InputStream inputStream =
getResources().openRawResource(R.raw.nom_du_fichier);

BufferedReader reader = new BufferedReader(new
InputStreamReader(inputStream));

StringBuilder stringBuilder = new StringBuilder();

String line;

try {

    while ((line = reader.readLine()) != null) {

        stringBuilder.append(line).append("\n");
    }
} catch (IOException e) {
    e.printStackTrace();
} finally {

    try {

        inputStream.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

String fileContents = stringBuilder.toString();
```

Utilisation des données :

- Traiter les données lues selon les besoins de l'application.

1. Combien de paramètres a la méthode <i>onActivityResult()</i> ? A) 1; B) 2; C) 3; D) 4.	2. Dans quel <i>dossier</i> sont placées les ressources de type String ? A)drawable; B)ressources; C)values; D)strings.
3. Que renvoie la méthode <i>getExtras()</i> ? A)Extras; B) Parcable; C)Extra ; D) Bundle.	4. Que font les instructions suivantes : Uri uri=Uri.parse("tel:0555555555"); Intent i= new Intent(Intent.ACTION_DIAL, uri); startActivity(i); A)Appeler le numéro; B)Afficher le numéro sur le composeur téléphonique; C)A et B; D)Aucune.
5. Que renvoie la méthode <i>rawQuery ()?</i> A)ResultSet; B)ContentValues; C)Bundle ; D)Cursor.	6. La méthode <i>commit()</i> permet de : A)Ouvrir une préférence partagée ;B)supprimer; C) Valider les modifications; D) Aucune.
7. Que renvoie la méthode <i>openFileInput()</i> ? A)Stream; B)Reader; C) OutputStream; D) Aucune.	8. Quelle est la classe mère de tous les <i>Layouts</i> (Conteneur) Android ? A) Container; B) View; C) ViewGroup; D)Aucune.

1. C) 3

2. C) values

3. D) Bundle

4. B) Afficher le numéro sur le composeur téléphonique

5. D) Cursor

6. C) Valider les modifications

7. A) Stream

8. C) ViewGroup

Réponses aux questions à choix unique (0.5 point pour chaque réponse juste) :

1. Quelle est la classe mère de tous les composants graphiques Android ? A) Component; B) ViewGroup; C) Widget; D)Aucune.	2. Que font les deux instructions suivantes ? Intent intent = new Intent(ActivityA.this, ActivityB.class); startActivity(intent); A)Créer un Intent implicite; B) Créer un Intent explicite; C) Lancer une activité; D) aucune.
3. Quelle est la classe utilisée par un Intent pour stocker les données supplémentaires (extra) ? A)Extras; B) Parcable; C) Bundle; D) Extra.	4. Les préférences partagées sont sauvegardées dans le dossier : A) /data/app/packageName/shared_preferences; B) /data/data/packageName/shared_prefs/ C) /data/data/packageName/shared_preferences; D) /system/data/packageName/shared preferences.
5. Quel est l'outil qui établit la communication avec l'émulateur ? A) AAPT; B)Logcat; C)ADB; D) LogTool.	6. Vous pouvez arrêter une activité en appelant sa méthode: A) onDestroy(); B) terminate(); C) finish(); D) exit().
7. La classe mère des boutons à deux états est : A)CheckBox; B)RadioButton; C)ToggleButton; D) Aucune.	8. La classe utilisée pour encapsuler les résultats d'une requête Select sur une base de données SQLite est : A) ResulSet; B) ContentValues; C) Cursor; D) Bundle.

Questions de cours (8 points) :

1. Expliquer le rôle d'un intent-filter.

0.75

Réponse :

Le fichier manifest contient des informations sur l'application, notamment les noms de ses composants applicatifs Android (Activité, service et Broadcast Receiver). Un intent filter est attaché pour chacun de ces composants dans ce fichier (manifest). Un intent filter déclare les capacités (sous-éléments) d'un composant. Généralement les sous-éléments suivant : <action>, <category> et <data> sont utilisés.

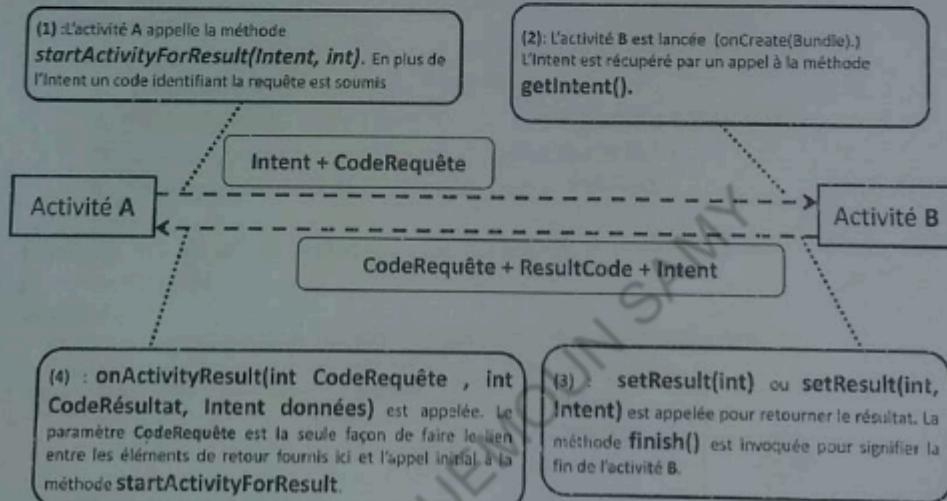
Le rôle de l'intent filter est alors de permettre au Package Manager de résoudre (répondre aux) les intent reçus par le système en comparant les sous-éléments des intents avec ceux des intents filter.

2. Expliquer le fonctionnement d'un intent explicite avec retour de résultats.

1.25

Réponse :

Le fonctionnement est résumé par le schéma suivant qui spécifie le sequencement des étapes:



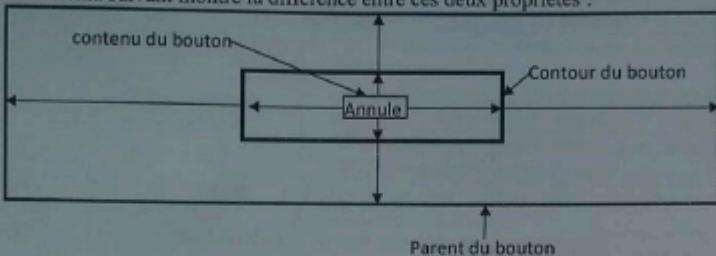
6. Expliquer la différence entre les attributs : **layout_gravity** et **gravity**.

Réponse :

1

- **layout_gravity** : définit la gravité de la vue ou du layout par rapport à son parent.
- **gravity** : définit la gravité du contenu de la vue par rapport au contour de la vue.

Le schéma suivant montre la différence entre ces deux propriétés :



Dans cet exemple les deux propriétés en la même valeur : **center**. Car le bouton est centré à l'intérieur de son parent grâce à `android:layout_gravity = center` et le texte (contenu) du bouton est centré dans sa vue grâce à `android:gravity = center`.

7. Expliquer la différence entre les deux méthodes : `getExternalStoragePublicDirectory()` et `getExternalFilesDir()`.

Réponse :

0.75

- `getExternalStoragePublicDirectory()` : permet de renvoyer l'emplacement public des fichiers (accessibles par toute les autres applications) sur une carte SD. Elle est appelée via la classe Environement.
- `getExternalFilesDir()` : permet de renvoyer l'emplacement privé des fichiers sur une carte SD. Elle est appelée via la classe Context.

8. Expliquer la différence entre les deux méthodes: `getPreferences()` et `getSharedPreferences()`.

Réponse :

0.75

- `getSharedPreferences()`: cette méthode est utilisée si vous avez besoin de plusieurs fichiers de préférences identifiés par leur nom, que vous spécifiez avec le premier paramètre.
- `getPreferences()`: cette méthode est utilisée si vous avez besoin d'un seul fichier de préférences pour votre activité. Parce que ce sera le seul fichier de préférences pour votre activité, vous ne fournissez pas de nom.

Questions de cours (9 points) :

1. Expliquer les étapes de lancement d'une application Android, tout en précisant le rôle des éléments qui interviennent (Activity Manager, Zygote, etc.).

1.25

Lorsque l'utilisateur clique sur l'icône de l'application le **lanceur** de l'application (launcher) appelle l'**activity manager** pour pouvoir lancer l'activité principale de l'application. Pour le faire ce dernier sollicite le **processus zygote** qui se **duplique** en créant un processus identique à lui-même et qui héberge une DVM ensuite l'apk de l'activité est chargée dans ce nouveau processus que la DVK exécutera.

2. Citer et expliquer deux objectifs de la gestion du cycle de vie d'une activité.

Les deux principaux objectifs de la gestion de cycle de vie d'une activité sont :

- 1- La fluidité de l'exécution de l'application : associer toutes les ressources à l'activité en cours d'exécution en libérant l'espace mémoire si nécessaire.

1.00

- 2- La non perte de données pour l'application lors de changement d'orientation ou lorsqu'elle quitte l'application et retourne

3. Citer les deux types de ressources d'une application Android, quelles différences entre elles ?

Les deux types de ressources sont :

- 1- Les ressources compilées : contiennent l'interface, les constantes (couleurs, string, etc.) les images (drawabale), etc. elles sont traitées par l'outil AAPT pour produire la classe java R qui contient les identifiants de ces ressources.

1.00

- 2- Les ressources non compilées (assets) : contiennent des fichiers images, son, etc. qui ne subissent aucune transformation (compilation). Elles n'ont pas d'identifiant dans la classe R.

5. Quelle est la différence entre les deux déclarations suivantes : « android:@+id/monbouton » et « android:@+id/monbouton »

0.50

La différence entre les deux déclarations est que la première permet de créer un nouvel identifiant dans la classe R associé pour le composant en question (un bouton). La seconde quant à elle permet de faire référence à ce composant en supposant qu'un identifiant associé est créé pour ce composant.

6. Citer et classer les techniques de stockage selon les dimensions suivantes : taille, emplacement et visibilité.

1. Les préférences partagées : forme <clé, valeur> sauvegardé dans un fichier XML. Elles sont de petite taille, elles sont placées en interne. Elles peuvent être publiques ou privées.

2. Les fichiers : pour stocker des données semi-structurées ou non structurées. Elles peuvent être de petite ou grande taille. Elles peuvent être placées en interne ou en externe (carte sd). Ils peuvent être publiques ou privés.

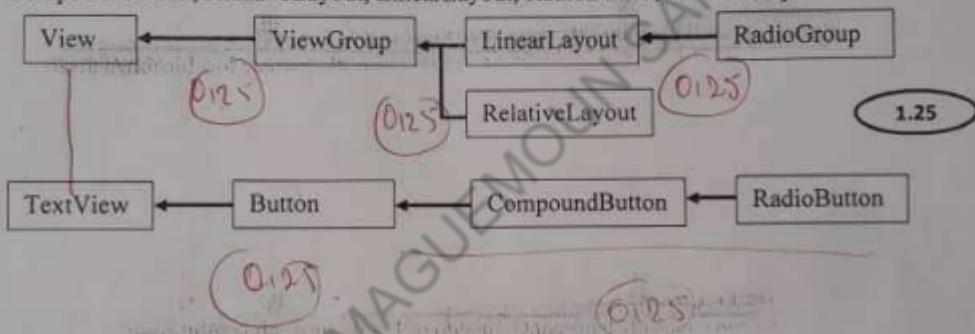
1.00

3. Les bases de données SQLite : pour sauvegarder des données structurées. Elles sont de grandes tailles, elles sont placées en interne. Elles sont privées à l'application.

7. Quel est le rôle des deux méthodes commit() et apply, quelle est la différence entre elles ?
Le rôle des deux méthodes est d'opérer les modifications dans une préférence partagée. La différence entre elles est que la méthode commit() est synchrone et que la méthode apply est asynchrone.
- 0.50
8. Quelle est la différence entre getExternalFilesDir() et getExternalStoragePublicDirectory() ?
La différence entre ces deux méthodes est que lors de l'utilisation de la première, les fichiers créés sous cette direction ils seront détruits à la désinstallation de l'application. Par contre avec la seconde ils ne le seront pas.
- 0.50

9. Citer et expliquer brièvement les classes qui interviennent dans la gestion de base de données SQLite sous Android.
Les classes qui interviennent sont :
- 1- SQLiteOpenHelper : classe abstraite qui offre des méthodes pour la gestion de la base de données : initialisation (nom de la base, etc.), la création, la mise à jours et l'ouverture.
 - 2- Classe concrète qui étend la classe SQLiteOpenHelper et matérialise toutes ces méthodes.
 - 3- SQLiteDatabase : c'est une classe qui permet de manipuler la base de données en lecture ou en écriture.
 - 4- ContentValues : pour encapsuler les données
 - 5- Cursor : pour encapsuler les résultats.
- 1.50

10. Classer les classes suivantes (relation d'héritage) : ViewGroup, Button, TextView, View, CompoundButton, RelativeLayout, LinearLayout, RadioGroup, RadioButton.



Questions à choix multiples (4 points) :

- | | |
|---|---|
| 1. Le composant Android qui permet de partager les données avec les autres applications est : | 2. Quelle est la méthode à utiliser pour ouvrir un fichier de type Raw ? |
| A) Base de données SQLite; B) Intent; C) Broadcast receivers ; D) Content Provider. | A) openFileInput(String NomFichier)
B) openRawResource(int idFichier)
C) openRawFile(int idFichier)
D) openRawFile(String NomFichier) |
| 3. La réponse aux requêtes de sélection dans une base de données SQLite sont de type : | 4. Quels sont les composants non applicatifs Android ? |
| A) ResultSet ; B) DataSet C) SQLite D) Aucune | A)SQLite B)Service C) Layout D) ContentProvider |
| 5. Quelle est la classe mère de tous les Layouts (Conteneur) Android ? | 6. Dans quel dossier sont placées les ressources de type String ? |
| A) Container; B)View; C).ViewGroup; D)Aucune. | A)drawable; B)ressources; C)values; D)strings. |
| 7. Un flux d'entrée de type caractère est modélisé par la classe: | 8. La classe qui permet d'encapsuler les données en entrée d'une base de données SQLite est: |
| A) InputStream B) Reader; C)Writer; - D)Aucune. | A) Data; B) Bundle; C)ContentValues; D)Cursor. |

NOM	Université Mouloud Mammeri de Tizi-Ouzou										
	Faculté de Génie Electrique et d'Informatique										
	Département d'Informatique										
	3 ^{ème} Année LMD : Programmation d'Applications Mobiles (Android).										
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 25%;">Nom</td> <td style="width: 25%;">Prénom</td> <td style="width: 25%;">Section</td> <td style="width: 25%;">Emargement</td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> </table>				Nom	Prénom	Section	Emargement				
Nom	Prénom	Section	Emargement								
EMD <i>Documentation interdite</i>											
<p>Questions du cours (9 points) :</p> <p>1. Citer et expliquer brièvement les deux relations existantes entre les classes : View et ViewGroup.</p> <p>Réponse : Les deux relations existantes entre les classes View et ViewGroup en Android sont l'héritage et la composition. 1.00</p> <p>1. Héritage : La classe ViewGroup hérite de la classe View. ViewGroup est une sous-classe de View qui représente un conteneur pouvant contenir d'autres vues. Par exemple, LinearLayout, RelativeLayout, FrameLayout sont tous des sous-classes de ViewGroup.</p> <p>2. Composition : Les ViewGroup contiennent et organisent les instances de View. Un ViewGroup peut contenir un ou plusieurs objets View en tant qu'enfants. Les vues sont organisées et disposées selon les règles spécifiques définies par le ViewGroup. Par exemple, un LinearLayout peut organiser ses vues enfants de manière linéaire, soit horizontalement soit verticalement.</p> <p>2. Expliquer avec un exemple le rôle des attributs : layout_weight, layout_gravity et gravity.</p> <p>Réponse :</p> <ul style="list-style-type: none"> - « layout_weight » : Permet de spécifier le poids relatif d'une vue dans un 'LinearLayout', pour la répartition de l'espace restant. Exemple : 'android:layout_weight="1"'. Demande une unité de l'espace restant. 0.75 - « layout_gravity » : Définit comment une vue doit être positionnée par rapport à son conteneur parent (RelativeLayout). Exemple : 'android:layout_gravity="center"'. - « gravity » : Détermine l'alignement interne du contenu d'une vue, tel que le texte, dans un « Button ». Exemple : 'android:gravity="center"'. <p>3. Expliquer le modèle MVC et citer un de ses objectifs.</p> <p>Réponse : Le modèle MVC (Modèle-Vue-Contrôleur) est un modèle de conception logicielle.</p> <p>Modèle (M) : Cette couche représente les données et la logique métier de l'application. Elle est responsable de la gestion des données, de leur manipulation et de leur persistance.</p> <p>Vue (V) : Cette couche est responsable de l'interface utilisateur et de l'affichage des données au(x) utilisateur(s). 1.00</p> <p>Contrôleur (C) : Cette couche agit comme un intermédiaire entre la vue et le modèle. Elle gère les interactions de l'utilisateur avec l'interface utilisateur.</p> <p>L'objectif principal du modèle MVC est de séparer les préoccupations et de favoriser la lisibilité et la maintenabilité.</p>											

- 4.** Citer et expliquer les trois méthodes de gestion d'événements (exemple : clic) sous android.

Réponse : Les trois méthodes de gestion d'événements (exemple : clic) sous android sont

 - 1- Classe Écouteur (Listener) : Cette méthode consiste à créer une classe (séparée) qui implémente l'interface appropriée (exemple : View.OnClickListener) pour gérer l'événement.
0.75
 - 2- Classe Écouteur (Listener) interne ou anonyme : Cette méthode consiste à définir une classe écouteur directement dans le code, en tant que classe interne et anonyme à la classe principale. Cela permet de gérer l'événement directement sans créer une classe séparée (setOnClickListener(new View.OnClickListener() {}))
 - 3- Un attribut XML : Cette méthode consiste à spécifier l'écouteur d'événements directement dans le fichier XML de l'interface. Vous pouvez utiliser l'attribut android:onClick pour définir la méthode de rappel à appeler lorsque le clic se produit.

5. Définir les rôles des classes : SQLiteOpenHelper, SQLiteDatabase, Cursor et ContentValues.

Réponse :

 - 1- SQLiteOpenHelper : Cette classe facilite la création (onCreate()), la mise à jour (onUpgrade()) et la gestion d'une base de données SQLite dans une application Android (getReadableDatabase() et getWritableDatabase()).
1.50
 - 2- SQLiteDatabase : Cette classe fournit les méthodes permettant d'effectuer des opérations de lecture (rawQuery()) et d'écriture (insert()) sur une base de données SQLite.
 - 3- Cursor : Cette classe permet de parcourir les résultats d'une requête sur une base de données SQLite et de récupérer les données sous forme de lignes.
 - 4- ContentValues : Cette classe est utilisée pour stocker les valeurs des colonnes d'une table dans une base de données SQLite. Elle facilite l'insertion et la mise à jour des données dans la base de données.

6. Citer les trois types de fichiers (stockage) sous Android et préciser comment ils sont gérés.
Sous Android, voici les trois types de fichiers de stockage couramment utilisés :

1. Fichier interne : Les fichiers internes sont stockés dans le répertoire interne '/data/data/nom_de_l_application/files/' . Ils sont accessibles en utilisant les méthodes openFileInput() et openFileoutput (). 1.00
 2. Fichier externe : Les fichiers externes sont stockés sur le stockage externe de l'appareil, tel qu'une carte SD. Ils peuvent être privés (getExternalFilesDir()) ou publics (getExternalStoragePublicDirectory()).
 3. Fichier Raw (Raw File) : sont des ressources intégrées dans l'application et sont stockés dans le répertoire 'res/raw' de l'application. Ils sont accessibles en utilisant les ressources de l'application via leur ID de ressource, tels que 'R.raw.nom_du_fichier'.

7. Classer les techniques de stockage selon les trois dimensions (taille, emplacement et accessibilité).

Réponse : Le classement des différentes techniques de stockage (préférences partagées, fichiers et SQLite) en précisant leur taille, accessibilité et emplacement est comme suit :

1. Préférences partagées :

- Taille : Les préférences partagées sont utilisées pour stocker de **petites** quantités de données.
 - Accessibilité : peuvent être **public** ou **privé** (activité ou application).
 - Emplacement : stocké en **interne**.

2. Fichiers :

Questions à choix multiples (4 points) :

- | | |
|---|--|
| <p>1. Quel est le nombre de paramètres qu'a la méthode setResult() ?</p> <p>A) un ; B) deux ; C) A ou B; D) 3 ; E) Autre (citer.....).</p> | <p>2. Quelle est la classe utilisée par un Intent pour stocker les données supplémentaires (extra) ?</p> <p>A) Extras; B) Parcable; C) Bundle; D) Extra.
E) Autre (citer.....).</p> |
| <p>3. Quel est le nom de l'outil qui convertit le bytecode Java en byte code Dalvik?</p> <p>A) Mobile Interpretive Compiler ; B) Dex Compiler
C) Dalvik Converter ; D) Android Interpretive Compiler ; E) Autre (citer.....).</p> | <p>4. Quelle est la classe qui s'occupe de représenter les données à sauvegarder dans une BDD Sqlite ?</p> <p>A) Cursor; B) Content; C) DataSqlite;
D) Autre (citer...ContentValues.....).</p> |
| <p>5. La classe mère directe d'un bouton (Button) est :</p> <p>A)CompoundButton; B)View; C)ViewGroup;
D) TextView; E) Autre (citer.....).</p> | <p>6. La méthode qui permet de valider d'une manière asynchrone les modifications sur une préférence partagée est :</p> <p>A)commit (B)apply(); C)edit(); D) changeAsync().</p> |
| <p>7. Quelles sont les classes qui représentent des flux binaires ?</p> <p>A)Writer; B)InputStream; C)Reader;
D)OutputStream; E) Aucune.</p> | <p>8. Laquelle des méthodes suivantes est utilisée pour terminer une activité?</p> <p>A) destroy(); B) close(); C) stop();
D) Autre (citer finish()).</p> |