**Android:**

* **Android basé sur Linux:** Android est un système d'exploitation mobile basé sur le noyau Linux. Cela signifie qu'il utilise la même base de code fondamentale que Linux, mais avec des fonctionnalités supplémentaires optimisées pour les appareils mobiles.
* **Deux attributs pour un View:**
  + **layout\_width:** Définit la largeur de la vue.
  + **layout\_height:** Définit la hauteur de la vue.
* **Unité privilégiée:**
  + **dp (density-independent pixels):** Unité relative à la densité d'écran de l'appareil. Cela garantit que la vue aura la même taille physique sur différents appareils, quelle que soit leur résolution d'écran.
* **Racine du fichier AndroidManifest.xml:**
  + **manifests:** C'est le nom du fichier racine du manifeste Android. Ce fichier contient des informations essentielles sur votre application, telles que son nom, son package, ses composants et ses autorisations.

**Calcul mental:**

* Pas de notes spécifiques fournies pour cette section.

**Gestion des exceptions:**

* **Superclasse de toutes les classes non contrôlées:**
  + **RuntimeException:** C'est la superclasse de toutes les exceptions non contrôlées. Les exceptions non contrôlées ne sont pas obligatoires à capturer et peuvent propager vers le haut de la pile d'appels jusqu'à ce qu'elles soient gérées par un gestionnaire d'exceptions global ou qu'elles provoquent la terminaison du programme.
* **Combien de blocs catchs peut contenir une structure try-catch:**
  + **Autant qu'on veut:** Il peut y avoir un nombre illimité de blocs catch dans une structure try-catch. Chaque bloc catch doit spécifier un type d'exception spécifique à capturer.
* **print StackTrace:**
  + Cette expression n'est pas une instruction Java valide. Il est possible que vous vouliez dire printStackTrace(), qui est une méthode utilisée pour imprimer la pile d'appels d'une exception.
* **Throws obligatoire dans l'entête d'une méthode:**
  + **Exception contrôlée non captée:** Si une méthode lance une exception contrôlée et que cette exception n'est pas captée par un bloc catch dans la méthode ou dans une méthode appelante, la méthode doit déclarer une clause throws dans son entête. Cela permet aux appelants de savoir quelles exceptions peuvent être lancées par la méthode.

**Visuel:**

* Pas de notes spécifiques fournies pour cette section.

**Patron de conception:**

* **Singleton:**
  + Un patron de conception qui garantit qu'une classe n'a qu'une seule instance à un moment donné. Cela est utile pour les classes qui fournissent des ressources partagées ou des services globaux.

**Documentation Android:**

* **deprecated:**
  + Ce terme indique qu'une API ou une fonctionnalité est obsolète et ne devrait plus être utilisée. Il est généralement recommandé d'utiliser une alternative plus récente et recommandée.

**Remplir un Spinner avec:**

* **ArrayAdapter:**
  + Une classe utilisée pour remplir un Spinner avec des données provenant d'un tableau ou d'une liste.

**Espace entre 2 objets:**

* **LayoutMargin:**
  + Un attribut utilisé pour définir la marge autour d'une vue. La marge est l'espace entre la bordure de la vue et les éléments environnants.

**Symbole expression régulière pour un chiffre:**

* **/d:**
  + Ce symbole correspond à n'importe quel chiffre de 0 à 9.

**Tests JUnit, méthode escomptée avec:**

* Pas de notes spécifiques fournies pour cette section.

**Intent pour passer d'une activité visible à une autre:**

* **Intent:**
  + Une classe utilisée pour lancer une nouvelle activité ou pour envoyer des messages entre différentes activités.

**Requête SQL:**

* **d.rawQuery(SELECT MAX(PRIX) FROM BISCUITS, null):**
  + Cette requête récupère le prix maximum de la table BISCUITS. Le paramètre null n'est pas nécessaire dans ce cas.
* **(Print le maximum) ; c.moveToNext() c.getDouble(0):**
  + Ce code exécute la requête, récupère le résultat (le prix maximum) et l'imprime.
* **Récupérer un nom à la fin du tableau:**
  + **c.moveToLast() c.getString(1):**
    - Ce code se déplace vers le dernier enregistrement du résultat et récupère la valeur de la deuxième colonne (supposant que la première colonne est l'index et la deuxième colonne contient le nom).

**"Scalable Pixel"**: Il s'agit d'une unité de mesure relative à la densité d'écran utilisée pour définir la taille des éléments d'interface utilisateur dans les applications Android. Les sp s'adaptent automatiquement à la densité de pixels de l'appareil, garantissant que les éléments apparaissent à la même taille physique sur différents appareils, quelle que soit leur résolution d'écran. Cela permet de créer des interfaces utilisateur plus cohérentes et plus adaptées à une large gamme d'appareils.