**🔁 Transtypage en Kotlin (Annexe 1 Transtypage)**

**Q1. Quelle est la différence entre le transtypage en Java et en Kotlin?  
R:**

* **Java : (Button) linearLay.getChildAt(0)**
* **Kotlin : linearLay.getChildAt(0) as Button  
  Kotlin utilise l'opérateur as pour le transtypage.**

**Q2. Quelle est la syntaxe Kotlin pour vérifier le type d’un objet?  
R: if (element is TextField) — équivalent à instanceof en Java.**

**🧠 Importance du buffer (Annexe 2)**

**Q3. Pourquoi utilise-t-on un tampon (buffer) lors de la lecture de fichiers?  
R: Le buffer permet de regrouper les accès aux fichiers pour éviter de lire octet par octet, ce qui accélère les lectures/écritures surtout sur de gros fichiers.**

**Q4. Quelle est la différence entre stockage interne et scoped storage?  
R:**

* **Stockage interne : privé à l'application.**
* **Scoped Storage : accès à des dossiers publics (Music, Downloads) sans demander de permission (à partir de l’API 29).**

**📁 Fichiers texte en mémoire interne (Annexes 1 et 1B)**

**Q5. Quelle est la structure de flux de lecture d’un fichier texte en Kotlin dans Android?  
R:**

val reader = BufferedReader(InputStreamReader(openFileInput("memo.txt")))

**Q6.** Quelle méthode utilise-t-on pour écrire un fichier texte en mémoire interne?  
**R:**

val writer = BufferedWriter(OutputStreamWriter(openFileOutput("memo.txt", MODE\_PRIVATE)))

**Q7.** Pourquoi utiliser .use { } en Kotlin?  
**R:** Pour s'assurer que les flux sont fermés automatiquement à la fin du traitement.

🧩 Sérialisation (Annexe 3)

Q8. Qu'est-ce que la sérialisation?

R: C’est le processus qui permet de sauvegarder l’état d’un objet (comme une liste de mémos) dans un fichier binaire afin de pouvoir le restaurer plus tard.

Q9. Que faut-il faire pour rendre une classe sérialisable en Kotlin?

R: La classe doit implémenter l’interface Serializable.

Q10. À quel moment appelle-t-on la méthode qui sauvegarde la sérialisation?

R: Généralement dans la méthode onStop() d’une activité.

📲 startActivityForResult et cycle de vie (Annexes 3d et 4)

Q11. À quoi sert startActivityForResult?

R: À lancer une autre activité et récupérer un résultat (par exemple : nom/prénom d’un utilisateur).

Q12. Quelle méthode est appelée lors d’un changement d’orientation du téléphone?

R: onSaveInstanceState() est appelée avant onStop() pour sauvegarder les données. Ensuite, onCreate() est rappelée.

Q13. Qu'est-ce qu'un "boomerang" dans le contexte Android?

R: C’est le retour d’une activité lancée avec startActivityForResult.

📡 Intents implicites (Annexe 7)

Q14. Quelle est la différence entre un intent explicite et implicite?

R:

Explicite : on cible une activité précise de notre app.

Implicite : on demande à Android de choisir une app capable de gérer l'action (ex: ouvrir Google Maps).

Q15. Donne un exemple d’intent implicite pour envoyer un SMS.

R:

val intent = Intent(Intent.ACTION\_SEND)

intent.putExtra(Intent.EXTRA\_TEXT, "Bonjour")

intent.type = "text/plain"

startActivity(Intent.createChooser(intent, "Partager via"))

**🎨 Thèmes et styles (Annexe 7C)**

**Q14.** Quelle est la différence entre un thème et un style ?  
**R :**

* *Thème* : affecte l'apparence globale de l'app.
* *Style* : s’applique à un seul composant (ex. TextView)

**🎭 Animations (Annexe 8)**

**Q15. À quoi sert ObjectAnimator ?  
R : À animer les propriétés visuelles d’un composant (ex. translationY, alpha)**

Q16. Quelle méthode permet de faire apparaître un menu en translation ?

R : ObjectAnimator.ofFloat(..., "translationY", ...).

**🧱 Patron de conception Builder (Atelier 1)**

**Q17.** Pourquoi utilise-t-on le pattern Builder ?  
**R :** Pour créer des objets complexes de manière claire et structurée sans surcharger le constructeur

**📡 Patron Observer (Atelier 3)**

**Q18.** Quel est le rôle du "sujet" dans le patron Observer ?  
**R :** Le sujet maintient une liste d'observateurs et les notifie en cas de changement

**Cycle de vie et startActivityForResult (Annexes 3d, 4)**

**Q9.** Quelles sont les méthodes du cycle de vie d'une activité ?  
**R :** onCreate, onStart, onResume, onPause, onStop, onRestart, onDestroy