1. **Was sind Funktionen höhere Ordnung?**

Eine Funktion höherer Ordung kann andere Funktionen als Parameter entgegennehmen oder zurückgeben.

1. **Beschreiben sie die Syntax von Lambda ausdrücken. Und wie unterscheiden sie sich mit Lamdas in Java?**

Lamdas in Kotlin werden in geschweiften Klammern geschrieben, Parameter und Ausdruck werden durch den -> Operator getrennt. Zb: val add(Int, Int)-> Int = {a,b -> a+ b }.

Lamdas in Jav sind sehr Ähnlich sind aber weniger Flexibel.

1. **Was ist ein funktionaler Typ? Was ist der Unterschied zwischen funktionalen Typen in Kotlin und funktionalen Schnittstellen in Java?**

Ein funktionaler Typ in Kotlin beschreibt eine Funktion als Datentyp. In Java müssen funktionale Schnittstellen verwendet werden, um eine ähnliche Funktionalität zu erreichen.

1. **Vergleichen Sie funktionale Typen mit Empfänger und solche ohne Empfänger.**

Funktionale Typen mit einem Empfänger ermöglichen den Zugriff auf die Mitglieder des Empfängers innerhalb des Lambda-Ausdrucks. Ohne Empfänger verhält sich der Lambda-Ausdruck wie eine normale Funktion ohne zusätzlichen Kontext.

1. **Was ist der Unterschied zwischen Lambda und anonymen Funktionen? Wann würden Sie eine anonyme Funktion bevorzugen und wann einen Lambda ausdruck?**

Lambdas habne eine kürzere schreib weise als anonyme Funktionen, sie sind besser wenn der Rückgabewert impliziet abgeleitet werden kann und kein komplexes verhalten gefordert wird. Bei Anonymen Funktionen hat man mehr Kontrolle über der Rückgabewert und das Verhalten der Funktion.

1. **Beschreiben Sie die Vor- und Nachteile von Inline-Funktionen. Was sind die Einschränkungen?**

Inline-Funktionen ermöglichen es, Code direkt an der Aufrufstelle einzufügen, was zu einer besseren Leistung führen kann. Nachteile sind größere Binärgrößen und der eingeschränkte Einsatz von Nicht-inlinierbaren Funktionen wie noinline. Auch der Rekursionseinsatz ist limitiert.

1. **Was ist eine callable reference? Beschreiben Sie die Formen von callable references. Wie unterscheiden sie sich von Methodenreferenzen in Java?**

Eine callable reference ist eine Referenz auf eine Funktion oder Eigenschaft. Sie werden mit :: deklariert (z. B. ::funktionsName). In Java gibt es ähnliche Methodenreferenzen, allerdings sind callable references in Kotlin vielseitiger, da sie auch auf Konstruktoren und Eigenschaften verweisen können.

1. **Beschreiben Sie das Verhalten von return-Anweisungen in Lambdas und anonymen Funktionen. Was ist eine qualifizierte return-Anweisung?**

In einem Lambda wird return verwendet, um aus der umgebenden Funktion zu springen, nicht nur aus dem Lambda selbst. Eine qualifizierte return-Anweisung (return@label) erlaubt die Rückgabe aus dem Lambda oder der anonymen Funktion selbst, was explizit gekennzeichnet wird.

1. **Vergleichen Sie die Inline-Modi von funktionalen Parametern: default, noinline, crossinline.**

Der Standardmodus erlaubt das Inlinieren. Mit noinline wird verhindert, dass ein Parameter inlinierbar ist, während crossinline sicherstellt, dass keine nicht-lokalen Rückgaben (also return) von diesem Parameter verwendet werden dürfen.

1. **Wie definiert man eine Erweiterungsfunktion? Verändern Erweiterungen die Klassen, auf die** sie angewendet werden?

Eine Erweiterungsfunktion wird außerhalb einer Klasse mit der Syntax Klassenname.Funktion() definiert. Erweiterungen verändern die Klasse selbst nicht, sondern bieten lediglich zusätzliche Funktionen für deren Instanzen.

1. **Wie würde man eine Begleitobjekt-Erweiterung verwenden?**

Eine Begleitobjekt-Erweiterung wird durch Deklaration einer Erweiterungsfunktion auf dem Begleitobjekt einer Klasse erreicht, z. B. Klassenname.Companion.Funktion().

1. **Was sind die Einschränkungen von Erweiterungseigenschaften?**

Erweiterungseigenschaften können keinen Zustand in der Klasse speichern, sie haben keinen Zugriff auf private Felder oder Funktionen der Klasse.

1. **Beschreiben Sie die Formen des this-Ausdrucks. Wofür wird qualifiziertes this verwendet?**

In Kotlin kann this auf das aktuelle Objekt oder einen äußeren Kontext verweisen. Ein qualifiziertes this wird verwendet, um zwischen mehreren Ebenen von verschachtelten Klassen oder Funktionen zu unterscheiden ( this@outerClass).

1. **Was sind die Besonderheiten der Deklaration von Erweiterungsfunktionen innerhalb von Klassen?**

Erweiterungsfunktionen, die innerhalb einer Klasse deklariert werden, können auf die Mitglieder der umgebenden Klasse zugreifen, allerdings haben sie keinen Zugriff auf private oder geschützte Mitglieder der Klasse, auf die sie angewendet werden.

1. **Was sind die Scope-Funktionen? Wie entscheidet man, welche Scope-Funktion besser für eine bestimmte Aufgabe geeignet ist?**

Scope-Funktionen wie let, run, apply, also und with bieten verschiedene Möglichkeiten, auf den Kontext eines Objekts zuzugreifen. Man wählt sie basierend auf der Notwendigkeit, ob das Objekt (this) oder das Ergebnis (it) im Lambda verwendet werden soll, sowie der gewünschten Rückgabe (das Objekt oder ein neuer Wert).