1. **Was ist eine Enum-Klasse? Welche eingebauten Operationen sind für Enums verfügbar?**  
   Eine Enum-Klasse ist eine spezielle Klasse, die eine Gruppe vordefinierter Konstanten (Enum-Werte) repräsentiert. Für Enums sind Operationen wie das Abrufen des Namens (name), des Ordinals (Index des Werts, ordinal) und das Auflisten aller Enum-Werte (values()) verfügbar.
2. **Was sind die Besonderheiten der Verwendung von when-Ausdrücken mit Enum-Klassen?**  
   Bei der Verwendung von when-Ausdrücken mit Enum-Klassen müssen alle Enum-Werte explizit behandelt werden, um sicherzustellen, dass kein Wert übersehen wird. Wenn alle Enum-Werte berücksichtigt werden, ist keine else-Klausel erforderlich.
3. **Wie kann man eine Enum-Klasse mit benutzerdefinierten Funktionen oder Eigenschaften definieren?**  
   Enum-Klassen können benutzerdefinierte Funktionen und Eigenschaften haben, um zusätzliche Logik oder Werte pro Enum-Wert zu speichern. Jeder Enum-Wert kann eigene Implementierungen für diese Funktionen und Eigenschaften definieren.
4. **Was ist eine Datenklasse? Welche Operationen werden automatisch für eine Datenklasse generiert? Wie kopiert man eine Instanz einer Datenklasse?**  
   Eine Datenklasse ist eine Klasse, die hauptsächlich zur Speicherung von Daten verwendet wird. Für Datenklassen werden automatisch Operationen wie equals(), hashCode(), toString(), copy() und componentN() generiert. Eine Instanz einer Datenklasse kann mit der Funktion copy() kopiert werden.
5. **Was ist eine Destrukturierungsdeklaration? Wo kann man eine verwenden?**Eine Destrukturierungsdeklaration ermöglicht es, die Werte einer Datenklasse oder eines anderen Objekts in mehrere Variablen zu zerlegen. Diese können in for-Schleifen, bei Rückgabewerten von Funktionen und bei Datenklassen verwendet werden.
6. **Was ist der Zweck von Inline-Klassen? Welche Anforderungen muss eine Klasse erfüllen, um inline zu sein?**  
   Inline-Klassen dienen dazu, Datentypen effizient zu erstellen, ohne den Overhead einer vollständigen Klasseninstanz. Sie müssen genau eine Eigenschaft enthalten und bieten eine Möglichkeit, primitiven Typen mehr Kontext zu geben.
7. **Beschreiben Sie Kotlin-Unsigned-Typen und deren eingebaute Operationen. Was sind die Besonderheiten im Vergleich zu signierten Ganzzahlen?**  
   Kotlin unterstützt Unsigned-Typen wie UInt, UByte, UShort und ULong, die nur nicht-negative Werte speichern können. Im Vergleich zu signierten Ganzzahlen haben Unsigned-Typen einen doppelt so großen positiven Wertebereich. Sie bieten ähnliche Operationen wie signierte Ganzzahlen, können jedoch keine negativen Werte darstellen.