**Objektmodellierung**

1. Was versteht man unter dem Begriff „Objektmodellierung“ in der objektorientierten Programmierung?
2. Was ist der Unterschied zwischen einer Klasse und einem Objekt?
3. Welche Rolle spielen Eigenschaften (Properties) und Methoden in einer Klasse?
4. Warum ist es sinnvoll, Klassen möglichst so zu modellieren, dass sie einen konkreten realen Zusammenhang widerspiegeln?

**IS-A- und HAS-A-Prinzip**

1. Erklären Sie den Unterschied zwischen dem IS-A-Prinzip und dem HAS-A-Prinzip.
2. Geben Sie je ein eigenes Beispiel für eine IS-A-Beziehung und eine HAS-A-Beziehung.
3. Warum wird bei einer IS-A-Beziehung oft Vererbung eingesetzt?
4. Welche Beziehung wird bei einer HAS-A-Beziehung oft durch Felder oder Eigenschaften abgebildet?

**Vererbung**

1. Was ist Vererbung und wie funktioniert sie in C#?
2. Wie wird eine Klasse als abgeleitete Klasse einer anderen deklariert?
3. Was passiert mit den Feldern und Methoden der Basisklasse bei Vererbung?
4. Was versteht man unter dem Begriff „Überschreiben“ (Overriding)?
5. Warum ist es in C# erforderlich, virtual und override zu verwenden?

**Interfaces**

1. Was ist ein Interface und wie unterscheidet es sich von einer abstrakten Klasse?
2. Wie wird ein Interface in C# definiert und implementiert?
3. Können Klassen mehrere Interfaces gleichzeitig implementieren?
4. Was ist der Vorteil von Interfaces im Hinblick auf Polymorphie?
5. Geben Sie ein Beispiel für eine Klasse, die ein Interface implementiert.

**Abstrakte Klassen**

1. Was ist eine abstrakte Klasse?
2. Was unterscheidet eine abstrakte Methode von einer normalen Methode?
3. Kann man Objekte von einer abstrakten Klasse erstellen? Begründen Sie.
4. Warum setzt man abstrakte Klassen oft als Basisklassen ein?
5. Wie unterscheiden sich Interfaces und abstrakte Klassen konzeptionell?

**Versiegelte Klassen**

1. Was ist eine versiegelte (sealed) Klasse?
2. Wie wird eine versiegelte Klasse in C# deklariert?
3. Warum sollte eine Klasse versiegelt werden?
4. Kann eine versiegelte Klasse dennoch von einer anderen Klasse erben?

**Polymorphie**

1. Was bedeutet der Begriff Polymorphie?
2. Erklären Sie den Unterschied zwischen Compile-Time-Polymorphie und Run-Time-Polymorphie.
3. Welche Rolle spielen virtual, override und new in der Polymorphie?
4. Geben Sie ein Beispiel für eine polymorphe Methode.
5. Warum ist Polymorphie ein zentrales Konzept in der objektorientierten Programmierung?

**Generische Klassen**

1. Was ist eine generische Klasse?
2. Warum sind generische Klassen hilfreich?
3. Wie wird eine generische Klasse mit mehreren Typparametern deklariert?
4. Was bewirkt ein where T : ...-Constraint?
5. Geben Sie ein eigenes Beispiel für eine generische Klasse mit zwei Parametern.

**Statische Klassen**

1. Was ist eine statische Klasse?
2. Welche Einschränkungen gelten für statische Klassen in C#?
3. Wie unterscheiden sich statische Methoden von Instanzmethoden?
4. Geben Sie ein Beispiel für den sinnvollen Einsatz einer statischen Klasse.
5. Warum können statische Klassen nicht instanziiert werden?

**Strukturen (structs)**

1. Was ist der Unterschied zwischen einer Struktur (struct) und einer Klasse (class)?
2. Warum sind Strukturen in C# Werttypen und keine Referenztypen?
3. Wann sollte man struct anstelle von class verwenden?
4. Können Strukturen Interfaces implementieren?
5. Warum dürfen Strukturen keine parameterlosen Konstruktoren definieren?