it.schule stuttgart	BfK-S Vorbereitung auf die Klassenarbeit OOP	it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart

Ich kann:	Hierbei fühle ich mich sicher	Ziemlich sicher	Hier habe ich kleine Lücken	Hier muss ich noch üben
Grundlagen				
Erklären, warum man die OOP anwendet				
den Unterschied zwischen Klasse und Objekt erklären				
den Zweck von public und private und protected in UML und OOP.				
erklären, wozu ein Konstruktor dient				
Konstruktoren mit oder ohne Parameter implementieren und aufrufen.				
public und private in C# programmieren				
eine Methode auf einem Objekt aufrufen.				
Eigenschaften eines Objekts setzen und lesen.				
eine Eigenschaft als Property deklarieren.				
ein Objekt mit new instanziieren.				
den Unterschied zwischen statischen und nicht-statischen Attributen erklären.				
UML				
in UML das Klassendiagramm einer Klasse zeichnen				
Attribute in ein UML-Klassendiagramm einzeichnen.				
Methoden mit Parameter(n) in ein UML-Klassendiagramm einzeichnen.				
Public, private und protected in UML einzeichnen. (+/-)				
Klassen voneinander ableiten in UML.				
Assoziationen in UML-Diagrammen darstellen				

it.schule stuttgart	BfK-S Vorbereitung auf die Klassenarbeit OOP	it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart

Ich kann:	Hierbei fühle ich mich sicher	Ziemlich sicher	Hier habe ich kleine Lücken	Hier muss ich noch üben
Vererbung				
das Konzept der Vererbung erklären				
die base.-Weiterleitung verwenden in C#				
Methoden in Unterklassen überschreiben in C#				
erklären, wann man abstrakte Methoden und abstrakte Klassen verwendet				
abstrakte Methoden in C# implementieren (Ober- und Unterklassen)				
Das Konzept der Polymorphie erklären				
Den Zweck von Interfaces erklären und eigene Interaces erstellen				
Interfaces für Klassen implementieren lassen				
Assoziationen				
erklären, wie sich die Richtung einer Assoziationen auf die Implementierung auswirkt				
Listen von Objekten in C# verwenden, um Assoziationen umzusetzen				
eine Assoziation implementieren und dabei die Multiplizität beachten				
den Unterschied zwischen Komposition und Aggregation erklären				
eine „1..*“-Multiplizität in C# umsetzen				