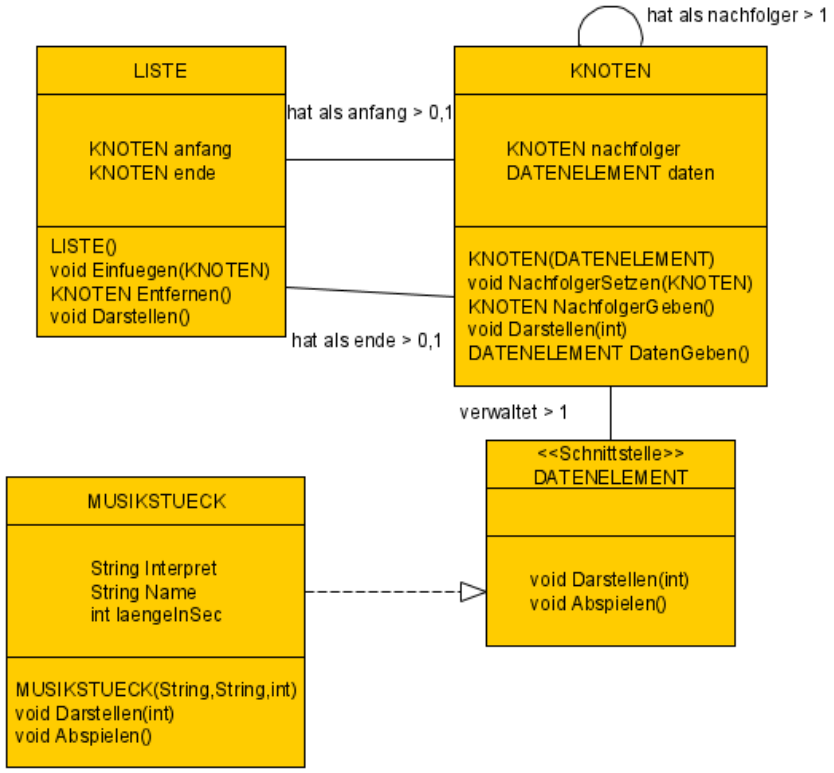
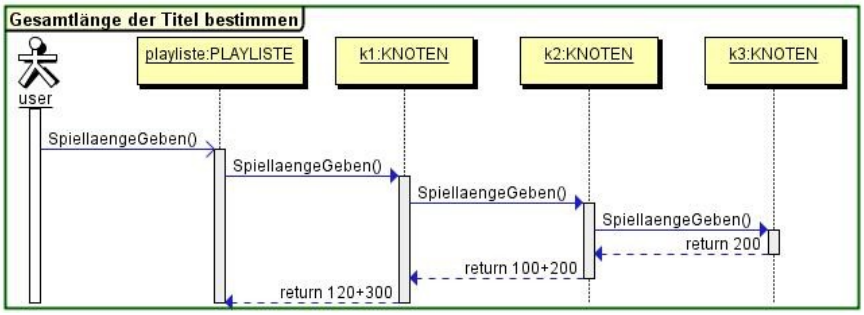
Checkliste: LISTEN LÖSUNG

	Ich kann …	Beispiele
1	… Klassendiagramme, Objektdiagramme, Struktogramme und Sequenzdiagramme zeichnen und vorgegebene Diagramme in Code umschreiben und umgekehrt.	
2	… verschiedene Implementierungen der Liste bewerten und erläutern wie man eine Liste mit Feld in eine dynamische Liste umbaut.	a) 1.Problem: Playliste ist nicht dynamisch, d.h. man kann nicht beliebig viele Elemente einfügen. 2. Problem: Bei einer langen Liste müsste beim Entfernen jeder Element um einen Platz nach vorne geschoben werden (im Feld) das kann ziemlich lange dauern. b) Um dynamisch zu werden müsste man das Feld durch eine Listenstruktur ersetzen. D.h. Die PLAYLISTE braucht zugriff auf anfang und ende der Liste. Jedes Element der Liste (z.B. TITEL) braucht zugriff auf seinen Nachfolger. Die typischen Methoden Einfügen und Entfernen bleiben vorhanden und müssen nur an die Struktur angepasst werden. Der TITEL bekommt die Methoden zum Setzen und Abfragen des Nachfolgers.
3	…darstellen, wie Einfügen und Entfernen in der Liste ablaufen.	Vergleiche Hefteintrag über Dynamik durch Referenzen.
4	… Struktur und Datenbereich der Liste gegeneinander abgrenzen und entsprechende Kommunikationswege und Klassendiagramme zeichnen.	a) Struktur: LISTE + KNOTEN; Daten: DATENELEMENT und TITEL. Durch die Trennung der Struktur von Daten ist die Liste jederzeit wiederverwendbar. Gegebenenfalls muss für eine neue Liste nur das DATENELEMENT andere Zugriffsmöglichkeiten auf Hintergrundinformationen erhalten. Die LISTE und der KNOTEN enthalten alle Struktur-relevanten Bestandteile der Liste. Einfügen fügt hinten ein und Entfernen entfernt das erste Element der Liste (und gibt es zurück). Die Daten sind in der Klasse TITEL, die die Schnittstelle DATENELEMENT implementiert zu finden. Die Methoden des DATENELEMENTs geben Möglichkeiten für den Zugriff auf die Daten. b) Erster Teil siehe oben. 2.Teil: Siehe Webseiten Lösung des Hefteintrages „Trennung von Struktur und Daten“ (Trifft auf die Liste zu) Die Methode TitelInformationAusgeben() läuft analog zur Methode InformationAusgeben() im Beispiel. Siehe
5	… Schnittstellen in Java implementieren und deren	a) Eine Schnittstelle garantiert Kommunikationswege zu einer Klasse, die die Schnittstelle implementiert. Das bedeutet, alle Methoden der Schnittstelle in der

	Funktion und Anwendungsbereich erläutern.	implementierenden Klasse über schrieben werden müssen. b) Man braucht die Schnittstelle z.B. bei PATIENT. Der Anfang der Klasse PATIENT muss mit „class PATIENT **implements** DATENELEMENT“ beginnen. Die Schnittstelle selber wird mit „interface DATENELEMENT“ statt „class DATENELEMENT“ definiert (in einer eigenen Datei!)

6	… den Begriff rekursive Methode definieren und rekursive Methoden der Liste implementieren.	a)
		 
		
		b)
		int SpiellaengeGeben() (im KNOTEN)
		nachfolger != null
		Wahr / Falsch
		return laengeInSekunden + nachfolger.SpiellaengeGeben() / return laengeInSekunden
		int SpiellaengeGeben() in PLAYLISTE
		anfang == null
		Wahr / Falsch
		return 0
		//Theoretisch nicht gebraucht bei
		//nicht leerer Liste / return anfang.SpiellaengeGeben()

7	… rekursive Methoden außerhalb der Liste anwenden.	Eine Methode folgt folgenden Pseudocode:
		`int Geheim(int i){`
		`if(i<=0)`
		`{`
		`return 1;`
		`}`
		`else`
		`{`
		`return i*Geheim(i-1);`
		`}`
		Die Methode ist rekursiv, weil sie sich selber wieder aufruft. Die Methode berechnet die Fakultät von „i“.