1. Was versteht man unter einer Datenstruktur? Nennen Sie mindestens zwei Beispiele!

Unter einer Datenstruktur versteht man den Datentypen an sich, zusammen mit der Menge von Operationen, die auf diesen Datentyp erlaubt sind. Das Ziel ist die Organisation, Strukturierung und Verwaltung von Daten.

Beispiele:
komplexe Datenstrukturen : **Queues, Stack ...**
einfach (in java vordefinierte) DS : **short, int, float …**

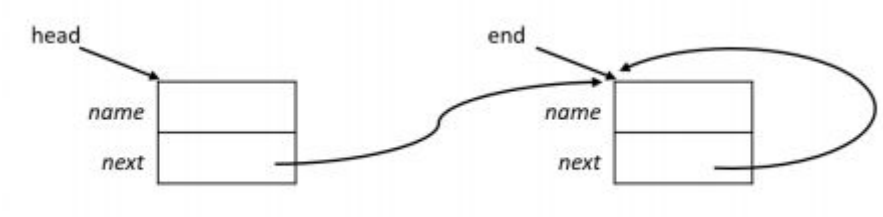
2. Beschreiben Sie kurz den grundlegenden Aufbau einer Liste und nennen Sie drei Listen Konzepte!

Eine Liste ist eine Sequenz von Elementen, also eine geordnete Sammlung von Objekten. Hierbei ist es wichtig, dass Listen während des Programmablaufs dynamisch verlängert aber auch verkürzt werden können. In einer Liste dürfen auch mehrfach gleiche Werte gespeichert werden. Der Benutzer kann auf alle Element in der Liste zugreifen und auf diese Operationen ausführen. DIe Elemente einer Liste müssen im Speicher nicht hintereinander liegen.
Listenelement referenzieren auf das vorherige Element oder auf das nachkommende Element. An den Enden einer LIste können hier die “Zeiger” entweder auf Null zeigen, auf sich selbst oder auf das erste bzw. letzte Element und bilden so eine Ringliste. Bei doppelt verlinkten Listen zeigen die Zeiger auch immer noch auf das vorige Element.

3 Listen Konzepte: Einfach verkettete Liste, Doppelt verlinkte Liste, Ringliste

3. Warum kann man eine einfach verkettete Liste nur in einer Richtung durchlaufen?

Dies liegt an der typischen Implementierung einer einfach verketteten Liste.



Hier gibt es lediglich einen “Zeiger”, also next. Dieser referenziert auf das nächste Element in der Liste. Geht man also nun von dem ersten Element zu dem zweiten Element, kennt

das zweite Element nicht die “Adresse”, also die Referenz, des ersten Elements und somit kann in die Liste auch nicht rückwärts durchlaufen werden, denn Java weiß nicht wo es nach dem “gefragten” Element suchen sollte.

4. Beschreiben Sie kurz den Unterschied zwischen einer Klasse und einem Objekt!

Die Klasse definiert was ein späteres aus der Klasse instanziiertes Objekt ausmacht. In einer Klasse sind alle Klassenspezifischen Werte festgelegt, sowie der Konstruktor (mit dem die Klasse erstellt wird) und die klassenspezifischen Methoden. Die Klasse ist wie ein Art Bauplan für das spätere Objekt.

Ein Objekt ist eine Instanziierung einer Klasse.

5. Nennen Sie mindestens zwei Typen von Klassen und geben Sie an, ob der Typ instanziierbar ist und ob von ihm geerbt werden kann!

Normale Klasse	**Abstrakte Klasse**
Von einer normalen Klasse kann geerbt werden, die Methoden sind ausführbar und können überschrieben werden mit @override	Von einer abstrakten Klasse kann geerbt werden, diese Methoden sind jedoch nicht auf das Kind anwendbar sonder geben vielmehr an, das diese Funktion auch bei der erbenden Klasse implementiert werden müssen.
Eine normale Klasse kann und muss sogar instanziiert werden um die in der Klasse, definierten Methoden und Werte verfügbar zu machen.	Nein, eine abstrakte Klasse kann nicht instanziiert werden. Das ist auch der Sinn einer abstrakten Klasse, somit bildet SIe vielmehr eine Modellierung Klasse die Signaturen für die erbenden Klassen vorgibt.

6. Welche Information einer Klasse werden bei der Vererbung in Java von der erbenden Klasse übernommen?

Von der einer (normalen) Klasse werden alle Attribute, Konstruktoren und Methoden übergeben. Es kommt hier jedoch auf die Signatur an. Beispielsweise private Signaturen verhindern das Zugreifen auf Informationen der Kinderklassen,

https://stackoverflow.com/questions/4716040/do-subclasses-inherit-private-fields

7. Fehlerhafte Implementation

```
public class HumanAge {
    private int age;

    public long getAge() { return this.age; }

    public void setAge(int age) {
        if (age > 0) {
            System.out.println("Das Alter darf nicht kleiner als 0 sein!");
        }
        this.age = age;
    }

    public HumanAge(int age) { this.setAge(age); }

    protected void HaveBirthday() { this.age++; }
}

public class HumanName {
    private String name;

    public int getName() { return this.name; }

    private void setName(String name) {
        this.name = name;
    }

    public HumanName() { this.setName("Franz Josef"); }
}

public class Person extends HumanAge, HumanName {
    public Person(String name, int age) {
        this.setAge(age);
        this.setName(name);
        super(0);
    }

    public String PrintPersonInfo(boolean withAge) {
        System.out.println(
            this.getName() + " ist " + this.getAge() +" Jahre alt."
            );
    }
}
```

age < 0 , Alter darf nicht unter 0

String

Falsche Signatur

Es kann nicht von mehreren Klassen geerbt werden

Der ganze Konstruktor macht keinen Sinn, da nicht von mehreren Klassen geerbt werden darf und zwei Setter von unterschiedlichen vererbeenden Klassen

super muss an erster Stelle stehen

Super ruft Konstruktor der vererbenden Klasse auf. Integer --> daher vermutlich HumanAge. Dieser ruft setAge auf. Daher neben falscher Implementierung auch noch doppelt sinnlos.

Warum bool ? wird nicht genutzt

Kein Return Value !

this.getAge() , getName() aufgrund falscher Vererbung nicht nutzbar!