Das Thema Ihrer Prufung ist die Implementation der Logik einer einfachen Schuladministration.

a) Implementieren Sie eine Klasse Student und geben Sie ihr **sinnvolle** Attribute fur eine ID

b) (Ganzzahl, bis zu 15 Stellen), den Vornamen und den Nachnamen. Alle Attribute sollen uber Getter-Methoden abgefragt werden konnen und unveranderlich sein. (6P)

c) Erstellen Sie einen Konstruktor mit drei Parametern, uber welchen die drei Attribute initialisiert werden konnen (auch die ID wird manuell vergeben). (1P)

d) Erstellen Sie einen Testfall fur den Konstruktor der Klasse Student und prufen Sie, ob alle Attribute korrekt gesetzt werden. (2P)

e) Uberschreiben Sie in der Klasse Student die toString()-Methode, damit diese einen String in folgender Form erzeugt (*attrN* = ihre individuelle Namensgebung): (2P)

   Student[attr1=wert1; attr2=wert2; attr3=wert3]

f) Erstellen Sie die notwendigen Unit-Test(s) um sicherzustellen, dass die Werte aller drei Attribute von toString() auch tatsachlich im String ausgegeben werden. (3P)

g) Implementieren Sie die equals-Methode (unter Einhaltung des Equals-Contracts!) fur Werte-Gleichheit auf dem Attribut **ID**. Testen Sie die formale Korrektheit mit einem Testfall, in welchem Sie das Tool **EqualsVerifier** (im Projekt bereits integriert) nutzen. (6P)

Wir erweitern die Klasse Student mit zusatzlicher Funktionalitat.

a) Stellen Sie sicher, dass keine Objekte von der Klasse Student erstellt werden konnen, die eine ID kleiner als 20200400000 enthalten. Im Fehlerfall werfen Sie eine geeignete Exception. Fur die Untergrenze der zulassigen ID's definieren Sie an geeigneter Stelle eine Konstante. (5P)

b) Implementieren Sie entsprechende Testfalle, welche die Anforderung von a) prufen. (5P)

c) Der Nachname muss aus mindestens **drei** Zeichen bestehen und darf auch nicht leer (null) sein.

   Werfen Sie auch dafur geeignete Exceptions und achten Sie auf aussagekraftige Messages in den Exceptions. (6P)

d) Implementieren Sie auch dafur wieder entsprechend selektive Testfalle, welche die Anforderung von c) prufen. (4P)

Wir entwerfen und implementieren eine Bewertung. Diese soll einen eigenen Typ fur die Bewertungen **A** bis **E** und **F** (ohne **Fx**) repasentieren, welchen die Noten **6.0**, **5.5**,⋯ bis **4.0** und fur **F** die Note **3.0** zugeordnet werden sollen.

a) Wahlen Sie das einfachste und am Besten geeignete Sprachkonstrukt von Java aus, um die obenstehenden Anforderungen umzusetzen. Implementieren Sie es mit **minimalem**Codeaufwand. (4P)

b) Der numerische Notenwert soll als **float** mit einer getter-Methode abgefragt werden konnen.Die Genauigkeit soll +/**-0.01** betragen. (2P)

c) Schreiben Sie einen exemplarischen Testfall, der fur **zwei** unterschiedliche Bewertungen pruft, ob der korrekte Notenwert geliefert wird. (5P)

d) Erganzen Sie einen Testfall mit einer mustergultigen JavaDoc, und erklaren Sie darin, wie Sie die verlangte Genauigkeit im Testfall berucksichtigt haben. (5P)

Ein Objekt der Klasse Student soll eine Liste von Anlassen mit Bewertung enthalten konnen.

a) Entwerfen Sie dazu eine Klasse Modul welche einen besuchten Anlass mit der dabei erreichten Bewertung (Bewertung) modelliert. Der Anlass selber soll uber eine einfach Kurzbezeichnung (String, z.B. “OOP”) identifiziert werden. Implementieren Sie die Klasse so, dass die zwei Attribute direkt mit dem Konstruktor (ohne weitere Uberprufung) initialisiert werden konnen. (5P)

b) Erweitern Sie die bereits bestehende Klasse Student mit einem neuen Attribut, so dass sie eine **beliebige** Anzahl von Modul-Objekten hinterlegen konnen. (1P)

c) Implementieren Sie auf der Klasse Student eine Methode, mit welcher ein Anlass (inklusive Bewertung) hinzugefugt werden kann (auch hier ohne weitere Uberprufung). Beachten Sie dabei die Datenkapselung und Information Hiding. (2P)

d) Erganzen Sie auf der Klasse Student eine Methode, welche die Anzahl der bereits besuchten Anlasse zuruckliefert. (2P)

e) Erganzen Sie eine Methode, welche die Durchschnittsnote (float) aller bewerteten Anlasse berechnet und zuruckliefert. Berucksichtigen Sie den Fall, dass diese Methode eine IllegalStateException zuruckliefern muss, und schreiben Sie dafur einen Testfall. (5P)

f) Schreiben Sie einen Testfall (nach Triple-A Pattern), welcher fur einen Student, welcher drei Anlasse besucht, und diese mit unterschiedlichen Bewertungen abgeschlossen hat, die korrekte Berechnung der Durchschnittsnote pruft. (3P)

Objekte der Klasse Student sollen uber einen Event melden konnen, wenn eine Bewertung mit Note **6.0** erreicht wird.

a) Entwerfen und implementieren Sie eine passende Event-Klasse fur das Ereignis. Sowohl der Anlass (String) als auch die Note (float) sollen Teil des Events sein. (4P)

b) Entwerfen Sie ein passendes (functional-)**Listener Interface** fur das Ereignis. (3P)

c) Implementieren Sie auf der Eventquelle die notwendigen Methoden fur die Registrierung/Deregistrierung und das Versenden des Events. (4P)

d) Erstellen Sie eine Demo-Klasse mit einer main()-Methode, welche einen Student erzeugt, sich darauf als Listener registriert (bzw. Sie sind frei zu entscheiden wer/was genau sich registriert), und dann Bewertungen erganzt, so dass mindestens ein Event auftritt. Ein einfaches System.out.println(…) zur Behandlung reicht aus. (7P)