Aufgaben für das Lernen von Java

Eine Java-Anwendung wird über durch Ausführen einer main-Methode gestartet, welches immer den folgenden Aufbau hat.

public ***static*** **void** main(String[] args){
…
}

- **public** ist hierbei eine Angabe über die Sichtbarkeit der Methode in Bezug auf das Projekt.
    - Unterschieden wird hier grob
        - public – projektweit bekannt
        - protected – packageweit bekannt
        - private – klassenweit bekannt
- **static** ist eine Angabe, welche Aussage darüber trifft, ob eine Methode von einer Klasse aufgerufen werden kann ohne ein Objekt dieser Klasse zu erzeugen.
- **void** ist die Angabe eines Rückgabewertes einer Methode. Der Rückgabewert einer Methode steht immer genau VOR dem Methodennamen.
    - Mögliche Rückgabewerte können primitive Datentypen sein wie z.B. int, byte, String, usw. und sogar Objekte oder oder oder …
- **main** ist der Name der Methode
- in den Klammern befinden sich die **Parameter** einer Methode, welche zum Aufruf einer Methode benötigt werden. Bei der main-Methode handelt es sich um ein Array bestehend aus Strings. Da die main-Methode vom System her aus aufgerufen wird, brauch sich mit den Argumenten nicht beschäftigt werden.

Die Lösungen der jeweiligen Aufgaben findest du unter folgender Adresse:
https://github.com/VtheunforgivenV/java/tree/master/src

Aufgabe 1 – Mathematische Operationen

1. Schreibe innerhalb der main-Methode eine Operation, wo zwei Variablen mit den Werten
    a. x = 2 & y = 1

   addiert werden (x+y).
2. Schreibe innerhalb der main-Methode eine Operation, wo zwei Variablen mit den Werten
    a. x = Ergebnis aus 1.1 & y = 2

   subtrahiert werden. (x-y)
3. Schreibe innerhalb der main-Methode eine Operation, wo zwei Variablen mit den Werten
    a. x = 10 & y =3

   multipliziert werden.
4. Schreibe innerhalb der main-Methode zwei Operationen, wo zwei Variablen mit den Werten
    a. x = 25 & y = 6

   dividiert werden. Beide Arten der Division sollen hier betrachtet werden.

   Hinweis:

   Es gibt 2 Arten von Division (bei Integer-Datentypen)
    - / oder auch „div“ genannt – gibt die Ganzzahl bei einer Division zweier Zahlen zurück
    - % oder auch „mod“ genannt – gibt den Divisionsrest bei einer Division zweier Zahlen zurück

   Hinweis 2:

   Bei Double- oder Float-Zahlen wird bei der Division mit (/ oder „div“) die korrekte Dezimalzahl zurückgeliefert.

Die jeweiligen Ergebnisse sollen auf der Konsole ausgegeben werden.

Aufgabe 2 – Bedingungen

Schreibe innerhalb der main-Methode **mehrere** if-Abfragen, wobei ein vorher definierter Wert(int) abgeprüft wird. Die Abfragen soll folgendermaßen lauten:

- Wenn der Wert kleiner ist als 10 dann soll der Text „Ich bin kleiner als 10“ auf der Konsole ausgegeben werden. Ansonsten soll der Text „Ich bin größer als 10“ auf der Konsole ausgegeben werden.
- Wenn der Wert kleiner gleich 10 ist dann soll der Text „Ich bin kleiner gleich 10“ auf der Konsole ausgegeben werden. Ansonsten soll der Text „Ich bin größer als 10“ ausgegeben werden
- Wenn der Wert größer ist als 10 dann soll der Text „Ich bin größer als 10“ auf der Konsole ausgegeben werden. Ansonsten soll der Text „Ich bin kleiner als 10“ auf der Konsole ausgegeben werden.
- Wenn der Wert größer gleich 10 ist dann soll der Text „Ich bin größer gleich 10“ auf der Konsole ausgegeben werden. Ansonsten soll der Text „Ich bin kleiner als 10“ auf der Konsole ausgegeben werden.
- Wenn der Wert gleich 10 ist dann soll der Text „Ich bin gleich 10“ auf der Konsole ausgegeben werden. Ansonsten soll der Text „Ich bin ungleich 10“ auf der Konsole ausgegeben werden
- Wenn der Wert ungleich 10 ist dann soll der Text „Ich bin ungleich 10“ auf der Konsole ausgegeben werden. Ansonsten soll der Text „Ich bin gleich 10“ auf der Konsole ausgegeben werden.

Hierbei soll im Nachgang der vorher definierte Wert manipuliert werden, sodass die verschiedenen Fälle sichtbar werden. Z.B. (mögliche Werte …,9, 10, 11, …)

Aufgabe 3 – verknüpfte Bedingungen

Schreibe innerhalb der main-Methode **mehrere** if-Abfragen, wobei ein vorher definierter Wert(int) abgeprüft wird. Die Abfragen sollen folgendermaßen lauten:

- Wenn der Wert größer ist als 10 **und** kleiner ist als 20 dann soll der Text „ich liege zwischen 10 und 20“ auf der Konsole ausgegeben werden. Ansonsten soll der Text „Ich bin entweder kleiner gleich 10 oder größer gleich 20“ auf der Konsole ausgegeben werden
- Wenn der Wert kleiner ist als 10 **oder** größer ist als 20 dann soll der Text „ich bin kleiner 10 oder größer als 20“ auf der Konsole ausgegeben werden. Ansonsten soll der Text „Ich bin größer gleich 10, aber kleiner gleich 20“ auf der Konsole ausgegeben werden.

Hierbei soll im Nachgang der vorher definierte Wert manipuliert werden, sodass die verschiedenen Fälle sichtbar werden. Z.B. (mögliche Werte …,9, …, 15, …, 21, …)

Aufgabe 4 – verschachtelte Bedingungen

Schreibe innerhalb der main-Methode **eine** if-Abfrage, wobei ein vorher definierter Wert(int) abgeprüft wird. Die Bedingunen lauten folgendermaßen:

- Wenn der Wert gleich 5 dann soll der Text „Ich bin 5“ auf der Konsole ausgegeben werden.
- Wenn der Wert gleich 10 dann soll der Text „Ich bin 10“ auf der Konsole ausgegeben werden
- Ansonsten soll der Text „Ich bin eine Zahl aber nicht 5 oder 10“ auf der Konsole ausgegebenen werden

Hierbei soll im Nachgang der vorher definierte Wert manipuliert werden, sodass die verschiedenen Fälle sichtbar werden. Z.B. (mögliche Werte …,4, 5,…, 10, 11, …)

Aufgabe 5 – Schleifen

Schreibe Schleifen, wobei bei jedem Schleifendurchlauf ein Wert von 0 hochgezählt werden soll. Die Schleifen sollen 10 Durchläufe haben und dabei den Wert jedes Mal um 1 erhöhen. Dies soll auf der Konsole ausgegeben werden. Die folgenden Arten von Schleifen sollen dabei umgesetzt werden:

- For-Schleife (Zählschleife)
- Do-while-Schleife (Endgeprüfte Schleife)
- While-schleife (Anfangsgeprüfte Schleife)

Ergebnis jeder Schleife: Ausgabe der Werte 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Aufgabe 6 – Felder

Erstelle innerhalb der main-Methode ein Feld von int-Werten und gib diese Werte nacheinander in einer gesonderten Zeile in der Konsole aus. Das Feld sollte die Größe von 10 besitzen und die folgenden Werte beinhalten: 3, 6, 9, 12, 15, 18, 21, 24, 27, 30.

Hinweis: Felder fangen beim Index 0 an. Das bedeutet, dass bei einer Größe von 10 die Indizes 0…9 beschreibbar sind.

Hinweis 2: Bei Feldern kann ein weiterer Schleifentyp benutzt werden – die foreach-Schleifen

Aufgabe 7 – Listen

Erstelle innerhalb der main-Methode eine Liste von int-Werten und gib diese Werte nacheinander in einer gesonderten Zeile in der Konsole aus. Die folgenden Werte sollten in der Liste vorhanden sein: 3, 6, 9, 12, 15, 18, 21, 24, 27, 30.
Nach der Ausgabe sollen die Einträge 6, 12, 18, 24 und 30 entfernt werden und danach die Liste erneut ausgegeben werden.
Hinweis: foreach-Schleifen sind auch bei Feldern anwendbar.
Hinweis 2: Listen sind die besseren Felder, da diese dynamisch erweiterbar und reduzierbar sind.
Hinweis 3: Bei dem Entfernen von Eintragen in einer Liste gilt das Prinzip – Never Change a running System. Einträge dürfen daher nicht innerhalb von Schleifen entfernt werden.

Aufgabe 8 – Methoden & Rückgabewerte

Innerhalb der Startklasse sollen neben der main-Methode zwei weitere Methoden erstellt werden. Die erste Methode soll eine Liste von int-Werten erstellen (siehe Aufgabe 7) und diese zurück liefern(Liste von int-Werten). Der Rückgabewert soll dann in einer Liste abgespeichert werden. Die zweite Methode soll als Parameter eine Liste aufnehmen und wie in Aufgabe 7 die Einträge in der Konsole ausgeben.
Hinweis: Die Methoden sollen von der main-Methode aufgerufen werden.

Aufgabe 9 – Objekte

Es soll eine neue Klasse erstellt werden mit dem Namen „JavaInt“. Diese Klasse soll eine private Variable vom Typ int mit dem Namen „value“ beinhalten. Zusätzlich soll der Wert nicht beschreibbar von außen sein, jedoch soll auf dessen Wert zugegriffen werden können. Der int-Wert der Variablen „value“ soll nur durch den Konstruktor der Klasse „JavaInt“ beschrieben werden können.
Nachdem die Klasse erstellt wurde soll innerhalb der main-Methode der Startklasse eine Liste von „JavaInt“ erzeugt werden, welche die gleichen Werte besitzen. D.h. Liste von 10 „JavaInt“-Objekten mit den Werten 3, 6, 9, 12, 15, 18, 21, 24, 27, 30.
Ob die Generierung und Ausgabe der Liste in einer eigenen Methode geschehen soll, bleibt selbst überlassen.
Hinweis: Der Konstruktor einer Klasse wird beim Erzeugen einer neuen Instanz aufgerufen. Meist ist dieser leer. Hier kann jedoch ein eigener Konstruktor angelegt werden, wobei Werte in den Konstruktor (wie beim Aufruf einer Methode) Parameter mitgeben werden können.

Aufgabe 10 – Vererbungen (extends)

Es sollen mehrere Klassen geschrieben werden, wobei eine Variablen und Methoden vererbt werden sollen. Folgende Klassen sollen dabei erstellt werden:

- Klasse „Person“ mit den Werten(String) Vor- und Nachname. Des Weiteren soll diese Klasse eine Methode besitzen um Vor- und Nachnamen einer Person auszugeben.
- Klasse „Lehrer“ mit dem Wert (String) Fach. Die Klasse „Lehrer“ soll von der Klasse „Person“ erben und soll eine Methode besitzen, um das Fach des Lehrers ausgeben zu können. Hierbei soll jedoch davor der Vor- und Nachname des Lehrers ausgegeben werden.
- Klasse „Schüler“ mit dem Wert (String) Klassenstufe. Die Klasse „Schüler“ soll von der Klasse „Person“ erben und soll eine Methode besitzen, um die Klassenstufe eines Schülers ausgeben zu können. Hierbei soll jedoch davor der Vor- und Nachname des Schülers ausgegeben werden.

In der main-Methode soll dabei jeweils ein Objekt vom Typ Schüler und ein Objekt Lehrer erstellt werden. Die Werte dabei sollen über den Konstruktor an die Objekte überreicht werden und dann im Objekt gesetzt werden. Nach Erstellung der Objekte sollen die Daten des Schülers und des Lehrers auf der Konsole ausgegeben werden.

- **Hinweis:** Durch Vererbungen werden alle Attribute auch den geerbten Klassen zugänglich, sofern die zu erbenden Methoden nicht *private* sind.

Aufgabe 11 – Maps (HashMap)

Es sollen Lehrer (aus der Aufgabe 10) in einer Map (Schlüssel-Wert-Paare) gespeichert werden. Hierbei sollen 5 Lehrer hinzugefügt werden mit einer fortlaufenden Nummer, welche als Schüssel benutzt werden soll. Hierbei soll folgendes die Aufgabe sein:

- Lehrer mit dem Schlüssel „3“ soll mit seinen Daten ausgegeben werden. Danach soll der Lehrer mit der Nummer überschrieben werden (neues Lehrerobjekt (nicht Daten ändern)).
- Alle Lehrer innerhalb der Map sollen nacheinander ausgegeben werden.

Aufgabe 12 – Interfaces bzw. Schnittstellen (implements)

Es soll ein Interface (Schnittstelle) erstellt werden, welches die folgende Methode besitzt „getScopeValue“. Diese Methode soll einen Integer als Wert zurückliefern. Zudem soll es drei Klassen geben:

- Dreieck (Triangle) mit einer Kantenlänge (hier soll dies ein gleichschenkliges Dreieck sein) als Integer-Wert. Diese Klasse soll das vorher erstellte Interface implementieren und die Methode für die Berechnung des Umfangs korrekt ausführen.
- Quadrat (Quad) mit einer Kantenlänge als Integer-Wert. Diese Klasse soll das vorher erstellte Interface implementieren und die Methode für die Berechnung des Umfangs korrekt ausführen.
- Rechteck (Rectangle) mit 2 Kantenlängen als Integer-Wert. Diese Klasse soll das vorher erstellte Interface implementieren und die Methode für die Berechnung des Umfangs korrekt ausführen.

Nachdem das Interface und die Klassen erstellt wurden, soll eine Liste vom Typen des Interfaces erstellt werden und dieser Liste jeweils ein Objekt der genannten Klassen hinzugefügt werden. Anschließend soll die Liste ausgegeben werden. Hierbei soll zum einen die Klasse (mit Namen) und der Umfang ausgegeben werden.