Objektorientierung 1:

Aufgabe 1:

Erstelle eine Klasse Phone mit den Attributen:

marke : String, model : String, color : String und battery : int

Die folgenden Funktionen sollen erstellt werden:

get_battery_status () : int
print_phone_infos() : void
reduce_battery_by (number : int) : void
charge_battery_by(numver : int) : void

und dem Konstruktor Phone (make : String, model : String , color : String)

Ein Phone hat zu Beginn 100 % Akku

Erstelle dann eine Testklasse und lade das Handy auf. Hol dir den Batteriestatus und reduziere den Akkustand.
Am Ende rufe die Funktion print_phone_infos () auf um alle Infos zum Handy an der Konsole auszugeben.

Ergänzung:

Ergänze der Klasse um eine statische ArrayListe. Jedes Mal wenn ein Phone erzeugt wird, soll der Name der Phone Modells der ArrayListe hinzugefuegt werden.
* Rufe ueber die Klasse dann die ArrayListe mit aller erstellten Phones auf

Aufgabe 2

Modelliere eine Klasse Auto:

Zu einem Auto werden KFZ-Kennzeichen, Kilometerstand, Tankvolumen, Kraftstoffverbrauch und Kraftstoffmenge gespeichert

Ein Auto hat folgende Methoden:

- tanken(double menge): Ausgabe auf Konsole ob Tanken erfolgreich ist mit Ausgabe der Kraftstoffmenge oder Ausgabe das Tanken nicht möglich ist, da das Tankvolumen überschritten wird
- fahren(double strecke): Ausgabe auf Konsole das Fahren erfolgreich ist und Kilometerstand ausgeben oder Ausgabe das Fahren nicht möglich ist, wegen zu wenig Kraftstoff (verbrauch = strecke / Kraftstoffverbrauch)

Die aktuellen Werte der Attribute KFZ-Kennzeichen, Kilometerstand und Tankvolumen können abgefragt werden. (get Methoden)

Ein Auto soll sowohl ohne Werte als auch mit geeigneten Startwerten erzeugt werden können. (Konstruktor mit und ohne Werte)

Implementiere die Klasse *Auto* in Java.

Implementiere eine Klasse *AppAuto*, in der zwei Autos erzeugt werden und diese fahren bzw. tanken.

Aufgabe 3:

Schreibe eine Klasse DiceGame:

Schreibe eine Methode int[] shuffleDice(), die ein int-Array mit 5 zufälligen Zahlen liefert, die zwischen 1 und 6 liegen.

Schreibe eine Methode isHomogeneous(int[] values), die testet, ob alle Werte im Array gleich sind. Prinzipiell darf das Array beliebig groß sein.

Schreibe eine Methode int[] occurrences(int[] values), die ein neues Array der Größe 6 liefert, das anzeigt, welche Augenzahl wie oft vorkommt.

Schreibe eine Methode isFullHouse(int[] values), die prüft, ob drei Würfel den gleichen Wert und zwei andere Würfel einen anderen gleichen Wert haben. Bei {1, 1, 1, 2, 2} liegt ein solcher Fall zum Beispiel vor.

Erstelle main Methode in welcher Würfel gewürfelt werden und mit dem Würfelergebnis die anderen Methoden verwendet werden.

Wir können annehmen, dass die Methoden immer mit korrekten Werten aufgerufen werden, also das Array nie null ist, die Anzahl Elemente immer korrekt ist und die Werte immer in den Wertebereichen liegen.