Aufgabe Objektorientiert: Definieren Sie **eine gemeinsame Oberklasse** und **zwei spezielle Unterklassen** zur Repräsentation der folgenden Objekte: Milch, Wasser, Cola, Apfel, Birne, Banane, ...

a) Speichern Sie folgende allgemeine Eigenschaft für alle Objekte in der **Oberklasse**: - Kalorien pro Gramm
- Bezeichnung (Milch/Wasser/Cola/Apfel/....)

b) Speichern Sie folgende speziellen Eigenschaften nur in den passenden **Unterklassen**:

- - Enthält Kerne (ja / nein)
- - Behälter (Glasflasche / Tetrapak / Plastikflasche / ...)

  Achten Sie darauf, welche Methoden und Eigenschaften in die Oberklasse gehören und welche in die Unterklassen!

c) Überschreiben Sie für alle Klassen die `toString ()` – Methode, geben Sie darin etwas Sinnvolles zurück.

d) Legen Sie in der main()-Methode **drei Objekte ihrer Klassen an** und speichern Sie diese in einer Liste (verwenden sie die vorhandenen Listen aus der Bibliothek, z.B. `listOf` oder `ArrayList`).

e) Ermitteln Sie in der `main()`-Methode, welches Objekt in der Liste die höchste Kalorienzahl pro Gramm hat. Geben Sie dieses Objekt auf der Konsole aus.

Aufgabe Objektorientiert: Definieren Sie eine Klasse `Person` **mit den Eigenschaften** `vorname, name alter, groesse`**.**

a) Definieren Sie eine Klasse `Freundschaft`**. Die Klasse soll die Freundschaft zwischen zwei Personen abbilden, indem sie zwei Personen als Eigenschaften enthält.**

b) Erzeugen Sie ein Objekt, das die Freundschaft zwischen - Fridolin Fröhlich (18 Jahre, 182 cm groß) und
- Hanna Heiter (19 Jahre, 176 cm groß)

abbildet.

Aufgabe Objektorientiert: Definieren Sie eine Klasse `Konto`**.**
Diese Klasse muss folgende private Eigenschaften besitzen:

`kontostand` - Aktueller Betrag auf dem Konto, z.B. `400,20 €`. Der Betrag darf negativ sein. `kreditRahmen` – Gibt an, um welchen Betrag das Konto überzogen werden darf.

Definieren Sie einen Konstruktor, der `kontostand` **und** `kreditRahmen` **festlegt. Die Klasse muss folgende Methoden besitzen:**

`public void einzahlen(int betrag )` erhöht `kontostand` um `betrag`. `int verfuegbaresGeld()`◊Liefert maximalen Betrag, der noch abgehoben werden kann. `int abheben(int betrag)` Hebt Geld vom Konto ab. `betrag` gibt an, wieviel Geld abgehoben werden **soll**. Der Rückgabewerte gibt an, wie viel Geld **tatsächlich** abgehoben wurde.

Hinweise: Der `kreditRahmen` hat Einfluss darauf, wieviel Geld verfügbar ist! Ein negativer Kontostand darf maximal soweit sinken, wie es der `kreditRahmen` zulässt. Es kann also beim Aufruf von `abheben(...)` passieren, dass jemand weniger Geld erhält als gewünscht.

a) Definieren Sie eine Klasse `Sperrkonto`**.**
Diese Klasse soll von `Konto` abgeleitet werden und folgende **zusätzliche Methode** enthalten: `public void sperren ()` sperrt das Konto

Wenn das Konto gesperrt ist, soll folgendes gelten:
- Einzahlungen sind weiterhin wie bisher möglich. - Es kann kein Geld mehr abgehoben werden.
- Das verfügbare Geld ist also auch 0.

Überschreiben Sie nur die Methoden aus der Oberklassen, die ihr Verhalten ändern.

Wenn das Konto nicht gesperrt ist, soll das Verhalten der Methoden gleich bleiben. Delegieren Sie dann die Arbeit an die Methoden der Oberklasse mit `super.xyz()`

b) Definieren Sie die Klasse `Buchung`**. Diese Klasse hat folgende Eigenschaften:**

`buchungsbetrag` – so viel Geld soll umgebucht werden `quellkonto` – von diesem `Konto` soll das Geld abgebucht werden `zielkonto` – auf dieses `Konto` soll das Geld umgebucht werden

Die Klasse muss eine Methode haben:

`public void buchen()` der `buchungsbetrag` wird vom `quellkonto` abgehoben und auf das `zielkonto` eingezahlt.

Wenn auf dem `quellkonto` nicht genug Geld verfügbar ist, dann soll eine `UnsupportedOperationException` geworfen werden.

Sie müssen keinen Konstruktor schreiben.

Aufgabe Objektorientiert: Definieren Sie eine Klasse `Person` **mit den Eigenschaften vorname, name alter, groesse. Definieren Sie einen Konstruktor zum Setzen aller Werte.**

a) Definieren Sie eine Klasse `Freundschaft`**. Die Klasse soll die Freundschaft zwischen zwei Personen abbilden, indem sie zwei Personen als Eigenschaften enthält.**

b) Erzeugen Sie ein Objekt, das die Freundschaft zwischen - Fridolin Fröhlich (18 Jahre, 182 cm groß) und
- Hanna Heiter (19 Jahre, 176 cm groß)

abbildet.

c) Gegeben sei das folgende Interface:

```
public interface Comparator<T> {
    /** @param o1 das erste Objekt zum Vergleich
     *  @param o2 das zweite Objekt zum Vergleich
     *  @return einen negative Wert, wenn o1 kleiner als o2
ist,
     *          einen positiven Wert, wenn o1 größer als o2
ist,
     *          den Wert 0, wenn o1 und o2 gleich sind
  int compare(T o1, T o2);
}
```

Implementieren Sie dieses Interface, so dass die Größe von zwei Personen miteinander verglichen wird. Sie können den Klassennamen frei wählen.

d) Definieren Sie eine Unterklasse von `Freundschaft` **mit der zusätzlichen Methode** `public Person vergleichen (Comparator <Person> c)`

Die Methode liefert die `Person` **zurück, die beim Vergleich mit dem** `Comparator c` **als größer gilt. Wenn beide Personen gleich sind, ist es egal, welche Person als Resultat geliefert wird.**

Aufgabe Objektorientiert:
Definieren Sie eine **abstrakte Klasse** `Tier` mit den Methoden

- `void setzeGroesse (int g)`
- `int getGroesse()`
- `boolean groesserAls(Tier t)`

Ihr Code:

```
// Größe in cm setzen
// Größe abfragen
// Größe von 2 Tieren vergleichen
```

DefinierenSiedieKlasse`Elefant`als **konkreteAbleitung**derKlasse`Tier`. Die Klasse `Elefant` bietet zusätzlich die Methode `void trompeten()` an. Die Funktion gibt „Töröö“ auf der Standardausgabe `System.out` aus.

Ihr Code:

Hinweis: Folgender Code sollte z.B. sinnvoll ablaufen:

```
Elefant grosserElefant = new Elefant ();
```

```
Elefant kleinerElefant = new Elefant ();
grosserElefant.setzeGroesse (330);
kleinerElefant.setzeGroesse (180);

if (grosserElefant.groesserAls(kleinerElefant) ) { System.out.println("Die
Welt ist in Ordnung");

    grosserElefant.trompeten();
}
```

Aufgabe Objektorientiert: Definieren drei Klassen: Heißgetränk, Kaffee und Tee.

Leiten Sie die spezielleren Klassen von der allgemeineren Klasse ab. Definieren Sie eine gemeinsame Eigenschaft und eine gemeinsame Methode in der Oberklasse. Definieren Sie für jede Unterklasse jeweils eine zusätzliche (spezielle) Eigenschaft und Methode.

Sie dürfen sich beliebige Eigenschaften und Methoden ausdenken!

Aufgabe Objektorientiert: Definieren Sie eine Klasse `Produkt` **zur Repräsentation von Produkten in einem Geschäft.**

Die Klasse muss folgende Eigenschaften besitzen: Produktname, Produktpreis.
Die Klasse muss einen Konstruktor besitzen, der Produktnamen und Produktpreis initialisiert.
Die Klasse muss folgende Methoden besitzen:

- `setPreis (...)` ◊ setzt den Preis neu
- `getPreis ()` ◊ liefert den aktuellen Preis zurück

Entscheiden Sie selbst, welche Datentypen und Rückgabewerte sinnvoll sind. Schützen Sie interne Eigenschaften vor direktem Zugriff!

a) Definieren Sie eine Unterklasse `ErweitertesProdukt`**, die von** `Produkt` **abgeleitet wird.**

Die Unterklasse muss als zusätzliche Eigenschaft einen **Zähler** besitzen, der speichert, wie häufig der Produktpreis geändert wurde. Dieser Zähler soll bei jedem Aufruf von `setPreis(...)` um eins erhöht werden. **Überschreiben** Sie daher in der Unterklasse die Methode `setPreis(...)` wie folgt:

- Jedes Mal wenn sich der Preis ändert, soll der Zähler erhöht werden.
- Delegieren Sie das eigentliche Setzen des neuen Preises an die Methode der Oberklasse.

b) Schreiben Sie eine Klasse `Warenkorb`**.** Dieser Klasse können Objekte vom Typ `Produkt` hinzugefügt werden. Die Klasse soll die hinzugefügten Produkte intern speichern, um den Gesamtpreis aller im Warenkorb enthaltenen Produkte zu berechnen.

Die Klasse muss einen Konstruktor besitzen, der festlegt, wie viele Produkte maximal im Warenkorb gespeichert werden können.

Die Klasse muss folgende Methoden besitzen:

- `addProdukt (Produkt p)` ◊ fügt das Produkt p dem Warenkorb hinzu, wenn noch

Platz im Warenkorb ist

- `getGesamtPreis ()` ◊ liefert den Gesamtpreis von allen Produkten, die sich gerade

im Warenkorb befinden