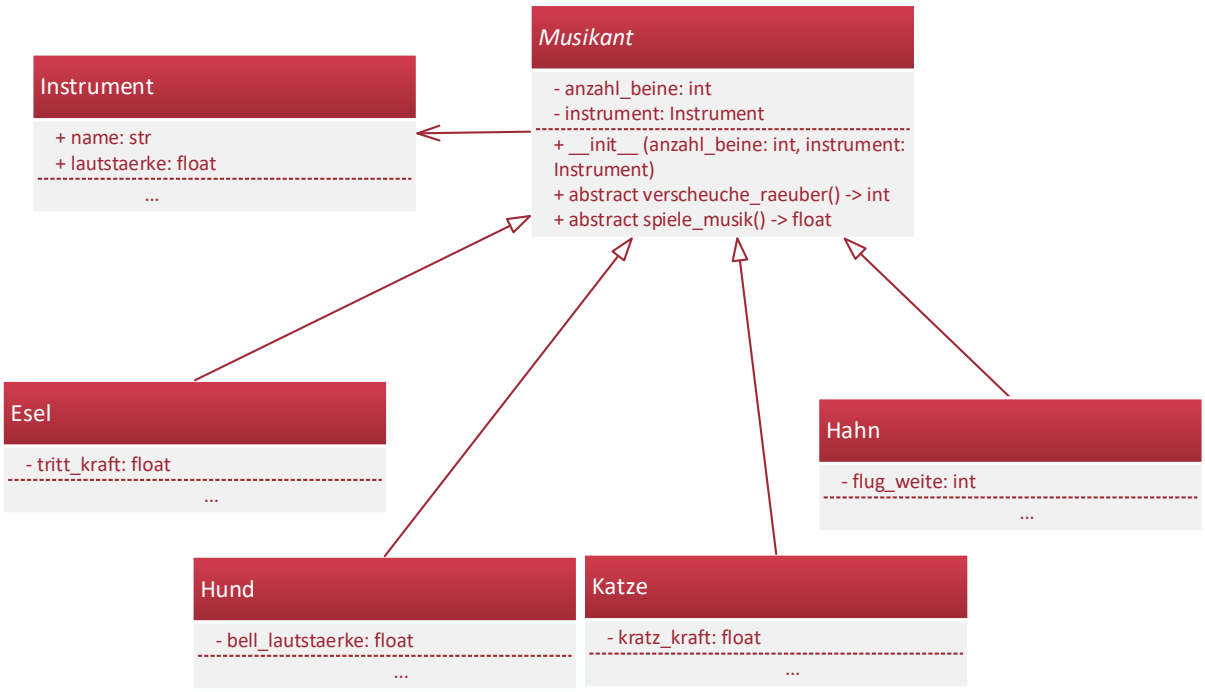
Die Bremer Stadtmusikanten

Eine grobe Übersicht der wichtigsten Klassen sehen Sie im folgenden Klassendiagramm:



Erstellen Sie die Klasse **Instrument** und ein __init__ um die beiden Attribute zu initalisieren.

Erstellen Sie eine abstrakte Klasse **Musikant** und fügen Sie die beiden Eigenschaften aus dem Klassendiagramm hinzu. (private und get Property) Implementieren Sie __init__ zum Initialisieren (der beiden Eigenschaften).

Erstellen Sie in der Klasse **Musikant** die beiden abstrakten Methoden aus dem Klassendiagramm:

Überschreiben Sie die __**repr**__ Methode und erstellen Sie einen String der die Rückgabewerte der abstrakten Methoden *verscheuche_raeuber* (Rückgabewert *1*) und *spiele_musik* (Rückgabewert *2*) auf folgende Art aufbaut:

> Verscheucht:*1*, Musiziert:*2* (statt *1* bzw. *2* die Rückgabewerte einsetzen)
>
> z.B: Verscheucht: 4, Musiziert: 6.0

Erstellen Sie die Klassen **Esel, Hund, Katze** und **Hahn,** die sich von der abstrakten Klasse Musikant ableiten. Erstellen Sie für jede Klasse _init_ um die Memberattribute der Basisklasse und die neuen Memberattribute zu initialisieren. (private und get property)

Überschreiben Sie für die Klassen **Esel, Hund, Katze** und **Hahn** die __**repr**__ Methode und fügen sie **zu Beginn des Strings** den Namen der Klasse und den Wert der neuen Memberattribute (der abgeleiteten Klasse) hinzu und danach den bestehenden String aus der Basisklasse. (Verwenden Sie die Implementation der Basisklasse)

> *Name der Klasse* *Memberattribut*: *String von Musikant __repr__ *
>
> z.B: Esel 2.3: Verscheucht: 4, Musiziert: 6.0

Implementieren Sie die **verscheuche_raeuber** Methoden – Diese liefern als Rückgabewert wie viele Räuber ein Musikant verscheuchen kann. Runden Sie den Rückgabewert mittels **math.floor** ab:

- **Esel**: Der Esel kann Trittkraft * Beinanzahl viele Räuber verscheuchen.
- **Hund**: Der Hund verscheucht die Räuber mit lauten Geräuschen und verwendet je nachdem was lauter ist entweder sein Bellen oder sein Instrument.
- **Katze**: Die Katze kratzt die Räuber und verscheucht sie damit. (z.B. Kraft 3.8 verscheucht 3 Räuber) Wenn die Katze verletzt ist und nur 3 Beine hat kann sie nur halb so stark kratzen und wenn die Katze nur 2 Beine (oder weniger) hat kann sie nur 1 Räuber verscheuchen.
- **Hahn**: Wie viele Räuber der Hahn verscheucht ist abhängig von seiner Flugweite und der Lautstärke seines Instruments.
    - Bei jeder Flugweite kleiner als 2 verscheucht er Lautstärke viele Räuber.
    - Bei Flugweiten zwischen 2 und 6:
        - Flugweite 2 -> 6 Räuber
        - Flugweite 3 -> 5 Räuber
        - Flugweite 4 -> 4 Räuber
        - Flugweite 5 -> 3 Räuber
        - Flugweite 6 -> 2 Räuber
    - Bei jeder Flugweite größer als 6 verscheucht er 1 Räuber.

math.floor bei jedem Tier nicht vergessen!

Implementieren Sie die **spiele_musik** Methoden – der Rückgabewert liefert die Lautstärke.

- **Esel** und **Katze** sind so laut wie ihr Instrument.
- Die **Hund** Lautstärke ist der mittlere Wert zwischen Instrument- und Belllautstärke. (Lautstärke Bellen 10, Lautstärke Instrument 20 -> Ergebnislautstärke ist 15)
- Der **Hahn** kräht zusätzlich zu seinem Instrument, weshalb sich seine Instrumentlautstärke um 2 erhöht. Da er auch noch durch die Luft fliegt wird seine Lautstärke durch die Flugweite dividiert.

Erstellen Sie eine Klasse **Quartett**, die Musikanten in einer privaten Liste speichern kann, initialisieren Sie diese im __init__. Erstellen Sie weiters eine **add(Musikant m: Musikant)** Methode um einen Musikanten in die Liste einzufügen.

Die restlichen Punkte können Sie in beliebiger Reihenfolge implementieren:

Erstellen Sie in der Klasse **Quartett** eine Methode **ist_quartett() -> bool** die überprüft ob das Quartett vollständig ist. (= enthält 4 Mitglieder)

Erstellen Sie in Klasse **Quartett** eine Methode **gemeinsam_raeuber_verscheucht() -> int** die aufsummiert wie viele Räuber durch das Quartett verscheucht werden können.

Erstellen Sie in Klasse **Quartett** eine Methode **double durchschnittliche_lautstaerke() -> float**, welche die durchschnittliche Lautstärke des Quartetts als Rückgabewert liefert. (Die Methode spiele_musik verwenden)

Erstellen Sie in der Klasse **Quartett** folgende Methode
get_musikanten_in_lautstaerke_bereich(von: float, bis: float) -> List[Musikant]
die in einer Liste alle Musikanten zurückliefert deren Lautstärke (spiel_musik) zwischen den beiden Grenzen liegt.

Erstellen Sie in der Klasse **Quartett** folgende Methode
get_anzahl_musikanten_mit_bein_anzahl() -> Map[int, int]
die zählt wie viele Musikanten es jeweils mit einer spezifischen Nummer an Beinen gibt. Im Dict sind die Schlüssel die Anzahl der Beine und der Wert die Anzahl der Tiere mit entsprechend vielen Beinen. (z.B. Eintrag **2 - 3** – **Schlüssel** 2 steht für 2 beinige Tiere; Wert 3 gibt an, dass 3 Tiere entsprechend viele Beine haben)