Wichtige Hinweise:

1. Sie müssen die Aufgaben A1, A2 und A3 bearbeiten.

2. In Ihrem Ruby-Prüfungsprojekt finden Sie vier Rubyscripte ***a1.rb,a2.rb, a3.rb***. Schreiben Sie alle Klassen, Module und Methoden einer Aufgabe in das Script, das zu dieser Aufgabe gehört und ***NUR!*** in dieses Script.

3. Schreiben Sie ***in jedes Script, das Sie bearbeiten, Ihre Matrikelnummer***.

4. Zu erreichen sind ***76 Pkt.*** Es genügen 50%, um die Klausur zu bestehen.

A1 Einzelaufgaben (Punkte 26 Pkt)

1. Gegeben die iterative Definition der Näherungsformel für $\ln(x)\ für\ x > 0.5$

   $sum(x,n) = \sum_{i=1}^{n} \frac{(x-1)^i}{i*x^i}$, für x $>$ 0.5

   Berechnen Sie die Formel iterativ. Erzeugen Sie einen ***ArgumentError*** für ungültige Werte.

   Beispielaufrufe entnehmen Sie bitte den RUnit Tests

   (***8 Pkt***)

2. Schreiben Sie eine rekursive Methode ***rek_count_type(type)*** für die Klasse ***Array***, die für ein beliebig geschachteltes Array die Anzahl der Elemente von Typ ***type*** zählt und die Anzahl zurückgibt. Es ist nicht erlaubt die Methoden ***flatten*** oder ***flatten!*** zu verwenden. (***10 Pkt***)

   Beispielaufruf mit Ergebnis:

```
[1,2,3.0,["a", [4.0,"b"], [[[5]]]], ["c",[]]].rek_count_type(Numeric)
# => 5

[1,2,3.0,["a", [4.0,"b"], [[[5]]]], ["c",[]]].rek_count_type(Array)
# => 7
```

3. Gegeben ein ***Hash***, der Kundennummern (Zeichenketten) auf Arrays von Bestellwerten (Zahlen) abbildet. Schreiben Sie die Methode ***max_bestellwert(ein_hash)***, der die Kundennummer mit der größten Summe an Bestellwerten zurückliefert. **Hinweis**: Verwenden Sie die Methode ***max*** für den ***Hash.*** Sie können sich eine Hilfsmethode schreiben, um die Einzelbestellwerte der Arrays aufzuaddieren. (***8 Pkt***)

   Beispielaufruf mit Ergebnis:

```
@bestell_hash = {"knr1" => [12,12,12.78,12],
                 "knr2" => [24,56,788.9, 30],
                 "knr3"=>[1,1,1,1,4]}

max_bestellwert(@bestell_hash)) # => "knr2",
```

Hinweise:

- Die Methodenskelette finden Sie im Rubyscript ***a1.rb***
- Verwenden Sie diese Methodenskelette für die Implementierung Ihrer Lösung.
- ***a1test.rb*** enthält die Unit-Tests zu den Aufgaben.

A2 Objekte vergleichen (Punkte 26Pkt / 4 Zusatzpunkte)

Gegeben folgende Definition der Klasse ***Adresse:***

```
class Adresse

  attr_reader :strasse, :hnr, :plz, :ort

  def initialize(strasse,hnr,plz,ort)
    @strasse = strasse
    @hnr = hnr
    @plz = plz
    @ort = ort
  end
end
```

Sie sollen die Klasse ***Adresse*** erweitern, so dass

1. Adressen sortiert werden können. Implementieren Sie eine Ordnungsrelation für Adressen, so dass Adressen zuerst nach ***plz***, dann nach ***ort***, dann nach ***strasse*** und dann nach ***hnr*** anordnet werden.
2. die Methoden ***<, <=, >, >=, between?()*** auf Adressen anwendbar sind.
3. gleiche Adressen in Arrays wiedergefunden werden.
4. beim Einfügen in eine Menge keine zwei gleichen Adressen eingefügt werden können.
5. Ergänzen Sie die Methode ***sortiere_absteigend_nach_strasse_hnr(adresse_ary)*** , die ein Array von Adressen absteigend sortiert, zuerst nach ***strasse*** und dann nach ***hnr.*** (4 Zusatzpunkte)

Die Klasse ***Adresse*** ist im Rubyscript ***a2.rb*** vorbereitet. Die zugehörigen RUnit-Tests finden Sie in ***a2test.rb***.

A3 Abstrakte Klassen und Objektrekursion (Punkte 24 Pkt / 6 Zusatzpunkte)

Sie sollen ein Klassenmodell für die Mitarbeiter eines Unternehmens entwerfen. ***Mitarbeiter*** sind entweder ***Angestellte*** oder ***Manager***.

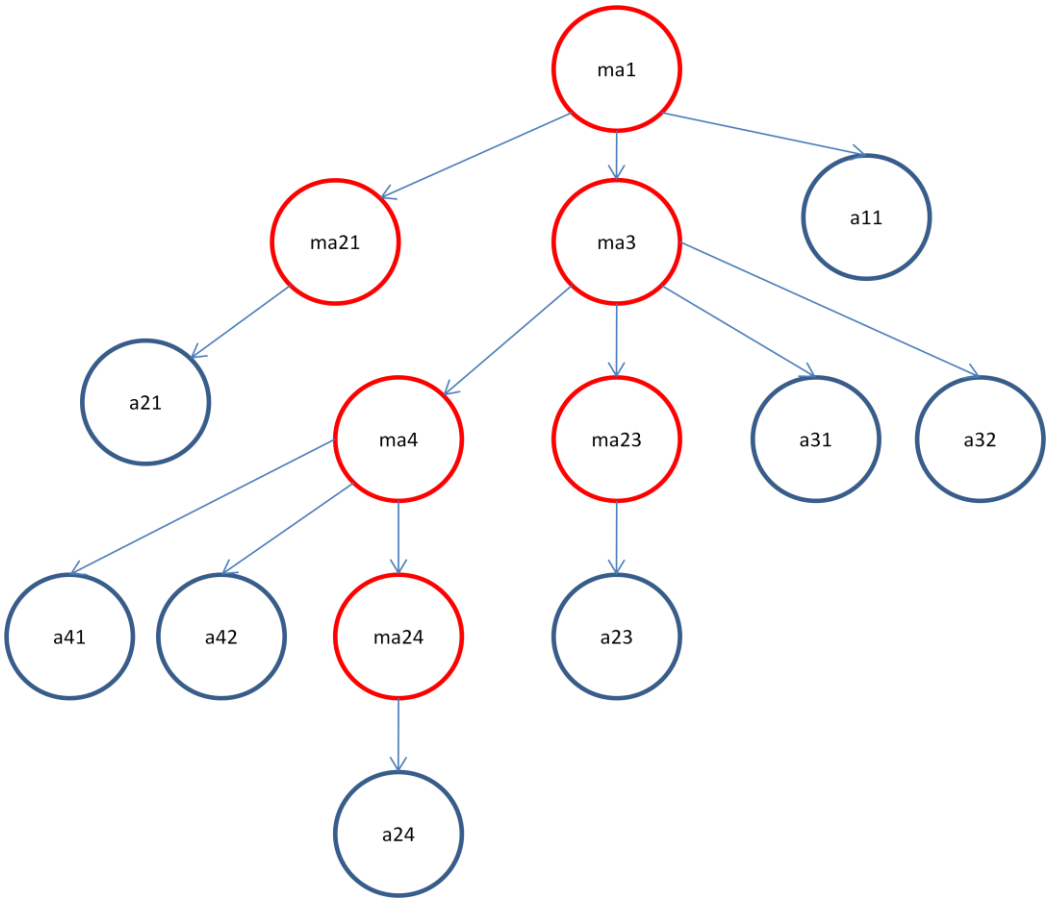
1. Alle ***Mitarbeiter*** haben einen Namen, eine Personalnummer und ein monatliches Basisgehalt. Diese Eigenschaften werden übergeben, wenn die Objekte erzeugt werden.
2. Mit der Methode ***neuer_mitarbeiter(ma)*** erhält der Manager neue ***Mitarbeiter***.
3. Alle ***Mitarbeiter*** haben ein Jahresgehalt, das sich aus dem Basisgehalt und einem Bonus berechnet.
    a. Der Bonus für ***Angestellte*** ist ein monatliches Basisgehalt.
    b. Der Bonus für ***Manager*** berechnet sich aus allen Boni der Mitarbeiter seiner Abteilung, dividiert durch Anzahl der Mitarbeiter, multipliziert mit dem Faktor ***1,8***.
4. Alle Angestellten haben eine Berichtsebene. (**Zusatzpunkte**)
    a. Für ***Angestellte*** ist diese Ebene 0.
    b. Für ***Manager*** ist die Berichtsebene das Maximum der Berichtsebenen der direkten Mitarbeiter plus 1.

Zu lösen:

- Entwerfen Sie ein Klassenmodell für diese Aufgabenstellung. Achten Sie auf abstrakte Klassen.
- Identifizieren Sie die abstrakten Methoden und implementieren Sie diese, wie Sie es in Vorlesung und Praktikum gelernt haben.
- Überschreiben (implementieren) Sie an den passenden Stellen in der Klassenhierarchie die Methoden:
    - ***jahres_gehalt()***
    - ***neuer_mitarbeiter(ma)***
    - ***bonus()***
    - ***berichts_ebene() (6 Zusatzpunkte)***
- Achten Sie in der Initialisierung darauf, dass die Initialisierung gemeinsamer Bestandteile in der Superklasse erfolgt.

Hinweise:

- Schreiben Sie **alle** Klassen und Methoden in das Script ***a3.rb.***
- Nennen Sie die Klassen ***Mitarbeiter, Angestellte, Manager.***
- Nennen Sie die Methoden der Klassen wie oben angegeben.
- Die zugehörigen RUnit-Tests finden Sie in ***a3test.rb.***
- Die Graphik zu den Testdaten zeigt die nächste Seite.


ma1
ma21
ma3
a11
a21
ma4
ma23
a31
a32
a41
a42
ma24
a23
a24