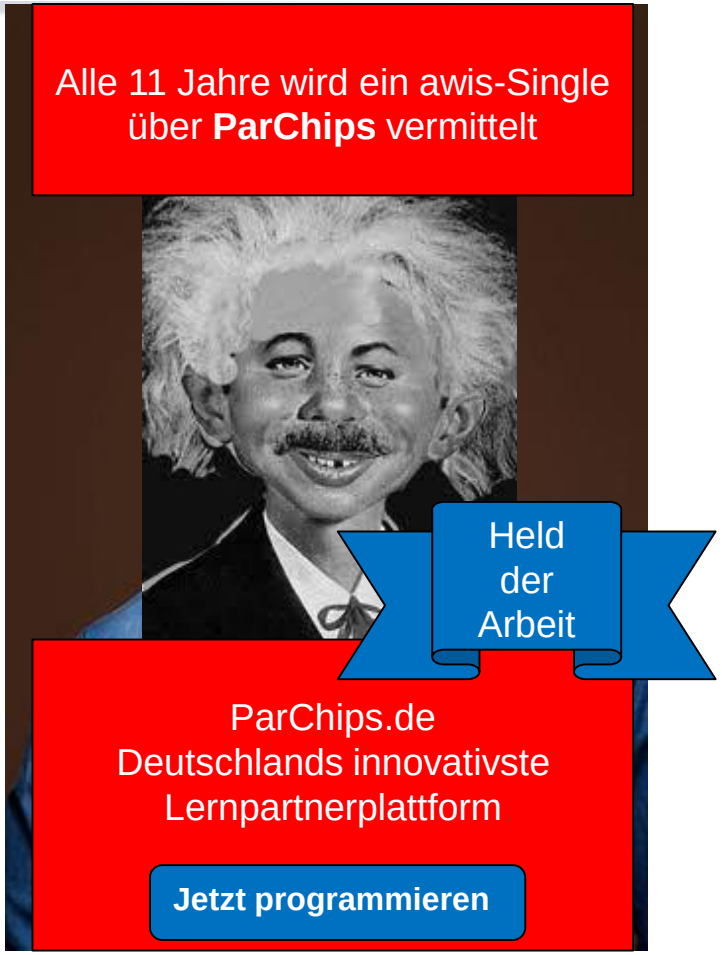
# Aufgaben zur Abgabe als Prüfungsleistung für Prog II

- Allgemeine Regeln und Abgabeformat (Export als Archive File, keine Umlaute, `package mypack;` etc.) wie in Prog I
- Sie haben immer die Möglichkeit, nur Teile der Aufgabe zu bearbeiten
- Falls Anforderungen nicht erfüllt sind, dann bitte im Kopf des Hauptprogramms (Beginning Comment) dokumentieren
  - Beispiel: Anforderung xyz nicht korrekt erfuellt, falls Eingabe = 123

- Sie dürfen Codestücke oder Hinweise aus Internet/Büchern etc. recherchieren, bei umfangreicheren Übernahmen (= mehr als zwei Zeilen übernommen) unbedingt Quellenangabe hinzunehmen!

# ParChips

- Schreiben Sie eine objektorientierte Java-Anwendung, die eine Bildung von Lerngruppen (= zwei Personen) unterstützt
  - Gegeben ist eine Liste von Studierenden
  - Gesucht ist eine Vermittlung eines geeigneten Lernpartners zur Unterstützung in Informatik oder Mathematik, so dass möglichst beide Teilnehmer profitieren
  - Zusätzlich soll das System auch eine geeignete Zuordnung aller Teilnehmer vorschlagen können

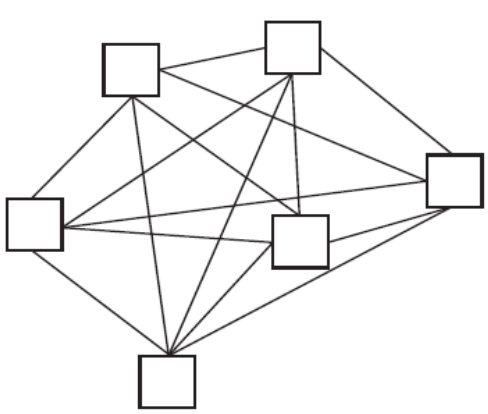


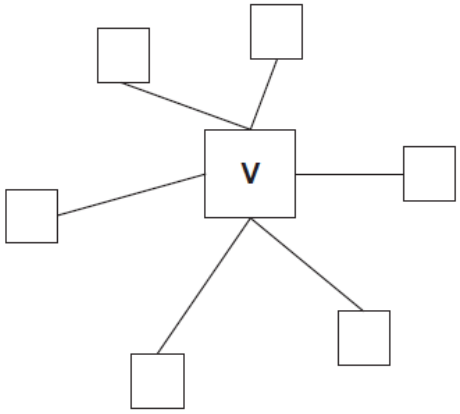
# Anforderungen

- Ein Studierender (= Klasse `KonkreterStudi`) hat vier Eigenschaften: `name` (= `String`), Informatik-Kenntnisse (`int`) und Mathematik-Kenntnisse (`int`), das sind jeweils Zahlen von 0 (= keine Kenntnisse) bis 9 (= sehr gute Kenntnisse), `partner`
  - Der Name muss eindeutig sein, d.h. keine zwei haben denselben Namen
- Die Implementierung soll dem Mediator-Pattern folgen: Es gibt einen Vermittler, der geeignete Lernpartner sucht.
  - Quelle: Siehe Beschreibung in OLAT aus J. Goll „Architektur- und Entwurfsmuster der Softwaretechnik“ S. 181ff (ohne Pattern auch möglich, aber 3 Punkte weniger); andere Quelle: https://en.wikipedia.org/wiki/Mediator_pattern
  - Der Vermittler versucht, den Gesamtnutzen zu optimieren
  - Beispiel:
    - StudA hat Informatik-Kenntnisse 2 und Mathematik-Kenntnisse 8,
    - StudB hat Informatik-Kenntnisse 6 und Mathematik-Kenntnisse 5,
    - Dann profitiert StudA mit 6 – 2 = 4 Punkten und StudB mit 8 – 5 = 3 Punkten;
    - Der Gesamtnutzen ist 4 + 3 = 7; negative Punkte werden nicht berücksichtigt, d.h. 0 ist der niedrigste Wert

# Mediator Pattern

- Bei Änderungen eines Objekts benachrichtigt das Objekt den Vermittler. Der Vermittler wiederum benachrichtigt die anderen Objekte über die erfolgte Änderung.
  - Vorteil: Lose Kopplung von Objekten, d.h. Objekte können hinzukommen oder wegfallen ohne große Nebenwirkungen
  - Einsatzbeispiele: Chat-Anwendungen oder GUI-Elemente, die sich gegenseitig beeinflussen





# Testfälle

Folgende drei Testfälle sind mittels `JUnit` mitzuliefern (Klasse `Testfaelle`)

- 1. Testfall:
  - Voraussetzung 3 Studierende sind bei der Klasse `KonkreterVermittler` registriert (ohne Partnerzuweisung)
    - A, Inf 0 Punkte, Mathe 10 Punkte
    - B, Inf 3, Mathe 7
    - C, Inf 6, Mathe 6
  - Dann kommt ein 4. Studi (D, Inf 10, Mathe 1) hinzu, der eine Partnerzuweisung wünscht; dieser erhält den Partner zugewiesen, der den höchsten Gesamtnutzen mit sich bringt
  - Beide Partner werden über das Ergebnis informiert
  - Testfall vergleicht für Objekt Studi D, ob der erwartete Partner ermittelt wurde
  - Ausgabe während des Testfalls auf der Konsole:

```
Ausgabe registrierte Studierende: [[name=A, fitnessInf=0, fitnessMathe=10], [name=B, fitnessInf=3, fitnessMathe=7],
[name=C, fitnessInf=6, fitnessMathe=6]]
KonkreterStudi D moechte zugeteilt werden und informiert den Vermittler
KonkreterVermittler berechnet Zuteilung
KonkreterStudi ist informiert, zugeteilter Partner von D ist: A
KonkreterStudi ist informiert, zugeteilter Partner von A ist: D
```

# Testfälle

- 2. Testfall
  - 4 Studierende mit den vorigen festen Daten sind registriert (alle ohne Partnerzuweisung)
  - Es wird eine Ausgabe der Präferenzmatrix auf der Konsole vorgenommen, die die Nutzen für alle Beteiligten anzeigt (siehe unten Ausgabe)
  - Der Vermittler erzeugt eine gute (oder zur Not auch weniger gute) Partnerzuweisung für alle Studierende, die Zuweisung wird auf der Konsole wie unten ausgegeben
  - Der Testfall vergleicht, ob der gewünschte Gesamtnutzen erreicht wurde

Ausgabe registrierte Studierende: [[name=A, fitnessInf=0, fitnessMathe=10], [name=B, fitnessInf=3, fitnessMathe=7], [name=C, fitnessInf=6, fitnessMathe=6], [name=D, fitnessInf=10, fitnessMathe=1]]
Praeferenzmatrix: [[0, 3, 6, 10], [3, 0, 3, 7], [4, 1, 0, 4], [9, 6, 5, 0]]

Zuordnung: A zu D; B zu C;
Gesamtnutzen: 23

# Testfälle

- 3. Testfall
  - 26 Studierende von A bis Z mit Zufallszahlen von 0..9 (reproduzierbare Zufallsreihe) sollen zugeordnet werden
    - Bei Initialisierung mit Zufallszahlen die Klasse `java.util.Random` verwenden mit einem festen Seed 123 (= Saat, Initialisierung)

```
Random rand = new Random(123);
int pseudoZufall = rand.nextInt(10);
```

  - Die Zuordnung soll einen hohen Gesamtnutzen liefern, aber wenn dies aus Zeitmangel/Komplexität etc. nicht möglich ist, dann ist eine einfache Zuteilung (A↔B; C↔D usw.) auch akzeptabel (minus 3 Punkte)
  - Falls ein hoher Gesamtnutzen angestrebt wird: Es ist bei einer großen Anzahl von Kollegen im Allgemeinen nicht mehr möglich, das Optimum zu finden, deshalb überlegen oder recherchieren Sie (mit Quellenangabe!) eine brauchbare Näherungslösung; diese soll eine Laufzeit von 10 Sekunden auf einem aktuellen PC nicht überschreiten
  - ✔ Ausgabe und Prüfung wie im vorigen Testfall

# Weitere Anforderungen

- ✓ Die Anzahl der Teilnehmer bei einer vollständigen Zuteilung ist gerade, damit kein Partner übrigbleibt, das Programm darf sich auf eine gerade Anzahl verlassen
- ✓ Die Zuordnungs-Tabelle muss eine `HashMap` sein
- Achten Sie auf typsichere Listen und Operationen
- Bezeichnung der Klassen: `Vermittler`, `KonkreterVermittler`, `Kollege`, `KonkreterStudi`, `Testfaelle`, `Zuord` (= zur Ermittlung einer möglichst guten Zuordnung), weitere Klassen möglich
  - Ein neu angelegter Kollege wird nach der Erzeugung bei einem Vermittler explizit registriert mittels Vermittler-Methode ✓`registriereKollege(Kollege k)`
  - Ein Kollege, der zugeteilt werden möchte, informiert den Vermittler mittels Kollegen-Methode ✓`zuteilen()`, die automatisch die Vermittler-Methode ✓`aenderungAufgetreten(Kollege k)` aufruft
- Geben Sie ein Klassendiagramm (nur Klassen und Attribute, keine Methoden) gemäß Notation der Vorlesung mit ab (mit Kardinalitäten)
  - Format: pdf, auch handschriftlich gezeichnet und dann eingescannt möglich