Name:

Matrikelnummer:

Bearbeitungszeit: 120 min.

Aufgabe 1. Gegeben ist die Klasse `Universitaet`, die Klasse `Ranking` und das Interface `ReverseRanking`. Die Klasse `Ranking` verfügt über einen parameterlosen Konstruktor und die Methode

```
public void sort(Universitaet[] uniliste),
```

die die Liste der Universitäten nach ihrem Ranking sortiert (sodass die beste Universität am ersten Listenplatz steht). Das Interface `ReverseRanking` definiert die Methoden

```
public void sort(Universitaet[] uniliste),
public void sortReverse(Universitaet[] uniliste).
```

Die Methode `sort()` entspricht der gleichnamigen Methode der Klasse `Ranking`, und die Methode `sortReverse()` soll die Liste der Universitäten in umgekehrter Reihenfolge sortieren (sodass die beste Universität am letzten Listenplatz steht).

Schreiben Sie eine Unterklasse `AllRankings` von `Ranking`, die das Interface `ReverseRanking` implementiert.

Aufgabe 2. Gegeben sind die Klassen `Auftrag` und `Simulation`. Die Klasse `Simulation` verfügt über die Methode

```
public double berechneAuslastung(List<Auftrag> reihenfolge),
```

die die Auslastung einer Fabrik berechnet, wenn die Aufträge in der gegebenen Reihenfolge bearbeitet werden.

Schreiben Sie eine Klasse `Reihenfolgeplanung` mit einem parameterlosen Konstruktor und der Methode

```
public List<Auftrag> gibReihenfolge(Simulation sim,
                                    Auftrag[] dieAuftraege),
```

die jene Reihenfolge der Aufträge zurückgibt, für die die Simulation die maximale Auslastung berechnet.

Hinweis: Diese Aufgabe ist am besten rekursiv mittels vollständiger Suche zu lösen, wobei die Bewertung einer möglichen Reihenfolge durch die Methode `berechneAuslastung()` der Klasse `Simulation` erfolgt.

Aufgabe 3. Scheiben Sie eine Klasse `Matching` mit einem parameterlosen Konstruktor und der Methode

```
public String bestMatch(String s1, String s2),
```

die einen längsten String zurückgibt, der als Teil-String sowohl in `s1` also auch in `s2` vorkommt.

Hinweis: Auf die Zeichen (vom Typ `char`) eines Strings `s` kann mit der Methode `s.charAt(i)` zugegriffen werden, wobei `0 <= i < s.length()`.

Beispiele:
`bestMatch("abcabc","bbbccc")` liefert `"bc"`,
`bestMatch("PeterAuer","Antenreiter")` liefert `"ter"`,
`bestMatch("bbb","ccc")` liefert `""`,
`bestMatch("abcabc","abacbc")` liefert `"ab"` oder `"bc"`.