Übung 8: Graphen

Abgabetermin: 24.05.2016

Name: _____ **Matrikelnummer:** _____

Gruppe: ☐ **G1** Di 10:15-11:00 ☐ **G2** Di 11:00-11:45 ☐ **G3** Di 12:45-13:30

Aufgabe	Punkte	gelöst	abzugeben schriftlich	abzugeben elektronisch	Korr.	Punkte
Aufgabe 1	12		Simulation			
Aufgabe 2	12		Java-Programm Testfälle und Ergebnisse	Java-Programm Testfälle und Ergebnisse		

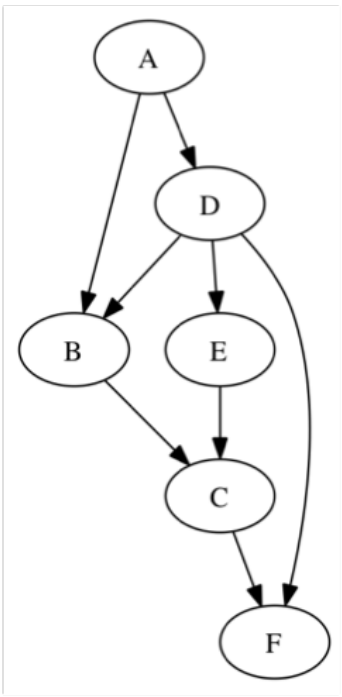
Aufgabe 1: Transitive Hülle mit dem Warshall-Algorithmus bestimmen

Berechnen Sie die transitive Hülle des Graphen mit dem Warshall-Algorithmus. Geben Sie alle Zwischenergebnisse als Adjazenzmatrix an. Markieren Sie neu hinzugekommene Kanten durch Einkreisen.

	A	B	C	D	E	F
A						
B						
C						
D						
E						
F						

Spalte A →

	A	B	C	D	E	F
A						
B						
C						
D						
E						
F						



Spalte B →

	A	B	C	D	E	F
A						
B						
C						
D						
E						
F						

Spalte C →

	A	B	C	D	E	F
A						
B						
C						
D						
E						
F						

Spalte D →

	A	B	C	D	E	F
A						
B						
C						
D						
E						
F						

Spalte E →

	A	B	C	D	E	F
A						
B						
C						
D						
E						
F						

Spalte F →

	A	B	C	D	E	F
A						
B						
C						
D						
E						
F						

Aufgabe 2: Implementierung des Warshall-Algorithmus

Implementieren Sie den Warshall-Algorithmus in Java. Fügen Sie dazu Ihrer Graphs-Klasse die Methode *makeTransitiveHull* hinzu. (Sie müssen für diese Übung die Methoden aus Übung 7 nicht abgeben.)

```
class Graphs {
  // ...
  public static Graph makeTransitiveHull(Graph graph) {...}
}
```

Erstellen Sie eine Adjazenzmatrix mit den Kanten im Graph. Verwenden Sie dafür ein *boolean[][]*-Array. Verwenden Sie als Indizes in die Matrix die Position der Knoten in der *vertices*-Liste des *Graph*-Objekts. Bestimmen Sie dann mit dem Warshall-Algorithmus schrittweise die Kanten der transitiven Hülle. Erstellen Sie als Ergebnis einen *neuen* Graphen der alle Kanten der transitiven Hülle enthält.

Implementierungshinweise:

a) Berücksichtigen Sie, dass bei einer ungerichteten Kante zwischen A und B sowohl A direkt von B als auch B direkt von A erreichbar ist.

b) Die *getEdge(start, end)*-Methode von Graph gibt *null* zurück, wenn keine Kante von *start* nach *end* existiert. Die Methode berücksichtigt auch ungerichtete Kanten von *end* nach *start*.

c) Ignorieren Sie eventuell vorhandene Kantengewichte.

d) Unabhängig davon ob der übergebene Graph gerichtet oder ungerichtet ist: Das Ergebnis von *makeTransitiveHull* soll ein gerichteter Graph sein (das macht Ihre Lösung einfacher).

   Optional: für ein schöneres (grafisches) Ergebnis können Sie zwei gerichtete Kanten A→B und B→A zu einer ungerichteten Kante A–B zusammenfassen.

e) Die *indexOf*-Methode von *Graph.vertices* liefert den Index eines Vertex-Objekts.