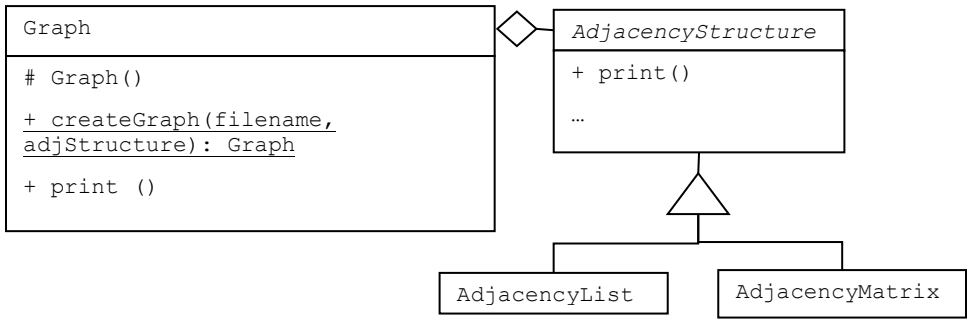
Algorithmen und Datenstrukturen Übung 6

		Späteste Abgabe	Übungen zu Algorithmen für Graphen im WS17
☐	IT 3_1, Wolfgang Auer	06.12.2017, 13:50h	Name _____
☐	IT 3_2, Wolfgang Sandholzer	06.12.2017, 13:50h	Abgegeben am _____

1. Erzeugen eines Graphen (12 Punkte)

Um uns bei den weiteren Übungen auf die algorithmischen Details konzentrieren zu können, wird im Rahmen dieser Übung ein Rahmenwerk für die Verwaltung von Graphen entwickelt.

Realisieren Sie die Klassen `Graph`, `AdjacencyStructure`, `AdjacencyList` und `AdjacencyMatrix` (siehe Klassendiagramm).

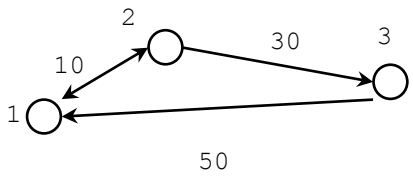


Die Methode `Graph.createGraph(…)` erzeugt eine neue Instanz der Klasse Graph. Die Definition des Graphs wird von der Datei *filename* eingelesen und abhängig vom Parameter `adjStructure` in einer Adjazenzliste bzw. –matrix gespeichert.

Verwenden Sie folgendes Format für die *„Graph-Definitions-Datei“:*

```
Graph     = Vertices Edges.
Vertices  = "V={" node { "," node } "}".
Edges     = "E={" ε | Edge | Edge { "," Edge } "}".
Edge      = "[" startnode "," endnode "," weight "]".
```

Beispiel:




```
V={1,2,3}
E={[1,2,10],[2,1,10],[2,3,30],[3,1,50]}
```

Die Methode `print()` sorgt dafür, dass die jeweilige Adjazenzstruktur in entsprechendem Format am Bildschirm ausgegeben wird:

`Adjazenzmatrix`

	1	2	3
1	0	10	0
2	10	0	30
3	50	0	0

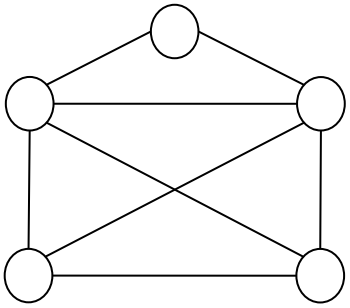
`Adjazenzliste:`

1	->	[2, 10]
2	->	[1, 10], [3, 30]
3	->	[1, 50]

Achten Sie bei der Realisierung auf den geschickten Einsatz von Polymorphismus und dynamischer Bindung!

2. Das Haus vom Nikolaus (8 Punkte)

Prof. Derwisch Mayar von der FH Konstantinopel lässt begeistert die Kreide fallen. Beim Zeichnen des Hauses vom Nikolaus fällt ihm auf, dass es sich einerseits um seinen byzantinischen Landsmann Nikolaus von Myra und andererseits um ein graphentheoretisches Problem handelt, für das er gerne eine Lösung implementieren würde.



Da Prof. Mayar gerade keine Zeit hat – er muss erst die Vita des hl. Nikolaus und sein Wirken beim Konzil von Nizäa nachlesen – ist es angebracht, ihn tatkräftig bei seinem Vorhaben zu unterstützen.

Gesucht ist ein offener Eulerpfad in einem Graphen. Erweitern Sie die Implementierung aus Beispiel 1 um die Suche nach einem offenen (oder, falls dieser existiert, einem geschlossenen) Eulerpfad bzw. bestimmen Sie, ob es diesen Eulerpfad gibt (Zur Bestimmung der Existenz ist keine Pfadsuche im Graphen notwendig).

Testen Sie die Implementierung (die auf dem Graphenframework aus Aufgabe 1 aufsetzt) mit dem Haus vom Nikolaus und anderen Graphen (interessant sind beim händischen Probieren auch 3-dimensionale geometrische Figuren wie der Kuboktaeder).

Hinweis:

- Testen Sie die Klassen ausführlich und vollkommen automatisch
- Achten Sie bei der Implementierung auf die Wiederverwendbarkeit Ihrer Klassen
- Halten Sie sich an die Programmierkonventionen!