```
class nrw_graph

    __init__(self)
           Es wird ein leerer ungerichteter Graph erzeugt
           Knoten und Kanten können mit beliebigen
           Attributen versehen werden.

    alleKanten(self)
           liefert eine Liste aller Kanten.

    alleKnoten(self)
           liefert eine Liste aller Knoten.

    alleNachbarknoten(self, start)
           liefert eine Liste alle Knoten, die über eine
           (ungerichtete) Kante mit dem angegebenen Knoten
           verbunden sind.

    besucheKnoten(self, node)
        der angegebene Knoten wird als besucht markiert.

    faerbeKante(self, start, ziel, farbe)
           Die angegebene Kante wird mit der angegebenen
           Farbe gefärbt.

    fuegeKanteHinzu(self, start, ziel, **args)
           eine neue Kante (ggf. mit weiteren Attributen)
           zwischen den angegebenen Knoten wird dem
           Graphen hinzugefügt. Die Endknoten werden ggf.
           neu erzeugt.

    fuegeKnotenHinzu(self, knoten, **args)
           ein neuer Knoten (ggf. mit weiteren Attributen)
           wird dem Graphen hinzugefügt.

    getKantenAttribut(self, start, ziel, attr)
           liefert den durch 'attr' spezifizierten
           Attribut-Wert der Kante

    getKantenAttribute(self, start, ziel)
           liefert ein Dictionary der Attribute der Kante.

    getKantenFarbe(self, start, ziel)
           liefert die Farbe der Kante, sofern vorhanden;
           ansonsten "?"
```

```
getKantenGewicht(self, start, ziel)
       liefert das Gewicht der Kante, sofern
       vorhanden; ansonsten "?"

getKantenMitAttribut(self, attr)
       liefert eine Liste aller Kanten mit dem
       angegebenen Attribut.

getKnotenAttribut(self, node, attrName)
       liefert den durch 'attrName' spezifizierten
       Attribut-Wert des Knotens.

getKnotenAttribute(self, node)
       liefert ein Dictionary der Attribute des
       Knotens.

getKnotenMarke(self, node)
       Die Markierung des angegebenen Knoten wird
       geliefert.

getKnotenMitAttribut(self, attr)
       liefert eine Liste aller Knoten mit dem
       angegebenen Attribut

gewichteKante(self, start, ziel, gewicht)
       Die angegebene Kante wird mit dem angegebenen
       Gewicht versehen.

graphEinlesen(self, dateiname, sep=',', header=0)
       Nachdem ein neuer leerer Graph erzeugt wurde,
       werden hiermit neue gewichtete Kanten zugefügt.
       Die angegebene Datei enthält neben der
       Überschriftzeile pro Zeile eine Kante,
       dargestellt mit - durch Kommata getrennt -
       Namen der Endknoten sowie das Gewicht der
       Kante.

kanteExists(self, start, ziel)
       liefert genau dann True, wenn der Graph eine
       Kante zwischen den angegebenen Enden besitzt.

kanteHatAttribut(self, start, ziel, attr)
       liefert genau dann True, wenn die spezifizierte
       Kante das angegebene Attribut hat.
```

knotenExists(self, node)
liefert genau dann True, wenn ein Knoten mit dem angegebenen Namen existiert.

knotenHatAttribut(self, node, attr)
liefert genau dann True, wenn der angegebene Knoten das angegebene Attribut hat.

knotenIstBesucht(self, node)
Es wird True geliefert genau dann, wenn der angegebene Knoten als besucht markiert wurde.

markiereKnoten(self, node, marke)
Der angegebene Knoten erhält die angegebene Marke.

setKantenAttribut(self, start, ziel,
attrName, attrWert)
Die angegebene Kante erhält unter dem angegebenen AttributNamen den angegebenen Wert. Ggf. wird ein neues Attribut mit dem angegebenen Wert erzeugt.

setKnotenAttribut(self, node, attrName, attrWert)
Der angegebene Knoten erhält unter dem angegebenen AttributNamen den angegebenen Wert. Ggf. wird ein neues Attribut mit dem angegebenen Wert erzeugt.

verlasseKnoten(self, node)
der angegebene Knoten wird als unbesucht markiert.