

```

class Graph {
    public void Graph() {
        String[] Knoten;
        int anzahlKnoten;
        Knoten = new String[20];
        anzahlKnoten = 0;
        Knoten[0] = "A";
        Knoten[1] = "B";
        Knoten[2] = "C";
        Knoten[3] = "D";
        Knoten[4] = "E";
        Knoten[5] = "F";
        Knoten[6] = "G";
        Knoten[7] = "H";
        Knoten[8] = "I";
        Knoten[9] = "J";
        Knoten[10] = "K";
        Knoten[11] = "L";
        Knoten[12] = "M";
        Knoten[13] = "N";
        Knoten[14] = "O";
        Knoten[15] = "P";
        Knoten[16] = "Q";
        Knoten[17] = "R";
        Knoten[18] = "S";
        Knoten[19] = "T";
        anzahlKnoten = 20;
        boolean neuerknoten(String bezetzung) {
            if (anzahlKnoten < 20) {
                int position = anzahlKnoten;
                Knoten[position] = bezetzung;
                anzahlKnoten++;
                return true;
            } else {
                return false;
            }
        }
        boolean neuerknoten(String bezetzung) {
            if (anzahlKnoten < 20) {
                int position = anzahlKnoten;
                Knoten[position] = bezetzung;
                anzahlKnoten++;
                return true;
            } else {
                return false;
            }
        }
        boolean neuerknoten(String knotena, String knotenb) {
            int a = Knotensuchen(knotena);
            int b = Knotensuchen(knotenb);
            if (a > -1 && b > -1) {
                if (a < b) {
                    Knoten[a][b] = knotena;
                } else {
                    Knoten[b][a] = knotenb;
                }
                return true;
            } else {
                int a = Knotensuchen(knotena);
                int b = Knotensuchen(knotenb);
                if (a > -1 && b > -1) {
                    if (a < b) {
                        Knoten[a][b] = knotena;
                    } else {
                        Knoten[b][a] = knotenb;
                    }
                    return true;
                } else {
                    return false;
                }
            }
        }
        int Knotausgeben(String knotena, String knotenb) {
            int a = Knotensuchen(knotena);
            int b = Knotensuchen(knotenb);
            if (a > -1 && b > -1) {
                if (a < b) {
                    System.out.println(Knoten[a][b]);
                } else {
                    System.out.println(Knoten[b][a]);
                }
            } else {
                System.out.println("Keine Knoten gefunden");
            }
            return -1;
        }
        int Knotensuchen(String bezetzung) {
            for (int i = 0; i < anzahlKnoten; i++) {
                if (Knoten[i].equals(bezetzung)) {
                    return i;
                }
            }
            return -1;
        }
        void ausgeben() {
            System.out.println("Adjazenzmatrix:");
            for (int x = 0; x < anzahlKnoten; x++) {
                for (int y = 0; y < anzahlKnoten; y++) {
                    System.out.print(Knoten[x][y]);
                }
                System.out.println();
            }
        }
    }
}

```

```

class Graph {
    public void Graph() {
        String[] Knoten;
        int anzahlKnoten;
        Knoten = new String[20];
        anzahlKnoten = 0;
        Knoten[0] = "A";
        Knoten[1] = "B";
        Knoten[2] = "C";
        Knoten[3] = "D";
        Knoten[4] = "E";
        Knoten[5] = "F";
        Knoten[6] = "G";
        Knoten[7] = "H";
        Knoten[8] = "I";
        Knoten[9] = "J";
        Knoten[10] = "K";
        Knoten[11] = "L";
        Knoten[12] = "M";
        Knoten[13] = "N";
        Knoten[14] = "O";
        Knoten[15] = "P";
        Knoten[16] = "Q";
        Knoten[17] = "R";
        Knoten[18] = "S";
        Knoten[19] = "T";
        anzahlKnoten = 20;
        boolean neuerknoten(String bezetzung) {
            if (anzahlKnoten < 20) {
                int position = anzahlKnoten;
                Knoten[position] = bezetzung;
                anzahlKnoten++;
                return true;
            } else {
                return false;
            }
        }
        boolean neuerknoten(String bezetzung) {
            if (anzahlKnoten < 20) {
                int position = anzahlKnoten;
                Knoten[position] = bezetzung;
                anzahlKnoten++;
                return true;
            } else {
                return false;
            }
        }
        boolean neuerknoten(String knotena, String knotenb) {
            int a = Knotensuchen(knotena);
            int b = Knotensuchen(knotenb);
            if (a > -1 && b > -1) {
                if (a < b) {
                    Knoten[a][b] = knotena;
                } else {
                    Knoten[b][a] = knotenb;
                }
                return true;
            } else {
                int a = Knotensuchen(knotena);
                int b = Knotensuchen(knotenb);
                if (a > -1 && b > -1) {
                    if (a < b) {
                        Knoten[a][b] = knotena;
                    } else {
                        Knoten[b][a] = knotenb;
                    }
                    return true;
                } else {
                    return false;
                }
            }
        }
        int Knotausgeben(String knotena, String knotenb) {
            int a = Knotensuchen(knotena);
            int b = Knotensuchen(knotenb);
            if (a > -1 && b > -1) {
                if (a < b) {
                    System.out.println(Knoten[a][b]);
                } else {
                    System.out.println(Knoten[b][a]);
                }
            } else {
                System.out.println("Keine Knoten gefunden");
            }
            return -1;
        }
        int Knotensuchen(String bezetzung) {
            for (int i = 0; i < anzahlKnoten; i++) {
                if (Knoten[i].equals(bezetzung)) {
                    return i;
                }
            }
            return -1;
        }
        void ausgeben() {
            System.out.println("Adjazenzmatrix:");
            for (int x = 0; x < anzahlKnoten; x++) {
                for (int y = 0; y < anzahlKnoten; y++) {
                    System.out.print(Knoten[x][y]);
                }
                System.out.println();
            }
            System.out.println();
        }
    }
}

```

```

42 int kanteAusgeben(String knotenA, String knotenB)
43 {
44     int a = knotenSuchen(knotenA);
45     int b = knotenSuchen(knotenB);
46
47     if (a > -1 && b > -1) {
48         return kanten[a][b];
49     } else {
50         return -1;
51     }
52 }
53
54 int knotenSuchen(String bezeichnung)
55 {
56     for(int i = 0; i<anzahlKnoten; i++) {
57         if (knoten[i].equals(bezeichnung)){
58             return i;
59         }
60     }
61
62     return -1;
63 }
64
65 void ausgeben() {
66     System.out.println("Adjazenzmatrix:");
67
68     for (int x = 0; x < anzahlKnoten; x++) {
69         for (int y = 0; y < anzahlKnoten; y++) {
70             System.out.print(kanten[x][y]);
71             System.out.print(", ");
72         }
73         System.out.println();
74     }
75 }
76
77 }

```

```

1 class Graph
2 {
3
4     String[] knoten;
5     int anzahlKnoten;
6     int[][] kanten;
7
8     Graph()
9     {
10         knoten = new String[20];
11         kanten = new int[20][20];
12         anzahlKnoten = 0;
13     }
14
15     boolean neuerKnoten(String bezeichnung)
16     {
17         if (anzahlKnoten < 20) {
18             int position = anzahlKnoten;
19             knoten[position] = bezeichnung;
20
21             anzahlKnoten++;
22             return true;
23         } else {
24             return false;
25         }
26     }
27
28     boolean neueKante(String knotenA, String knotenB,
29                         int wert)
30     {
31         int a = knotenSuchen(knotenA);
32         int b = knotenSuchen(knotenB);
33
34         if (a > -1 && b > -1) {
35             kanten[a][b] = wert;
36             return true;
37         }
38
39         return false;
40     }
41

```