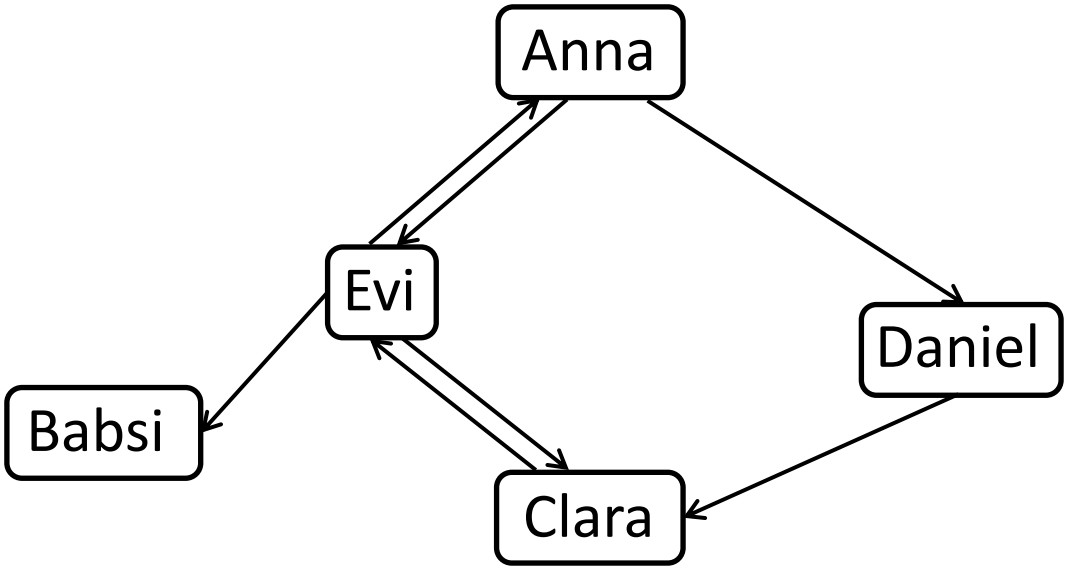
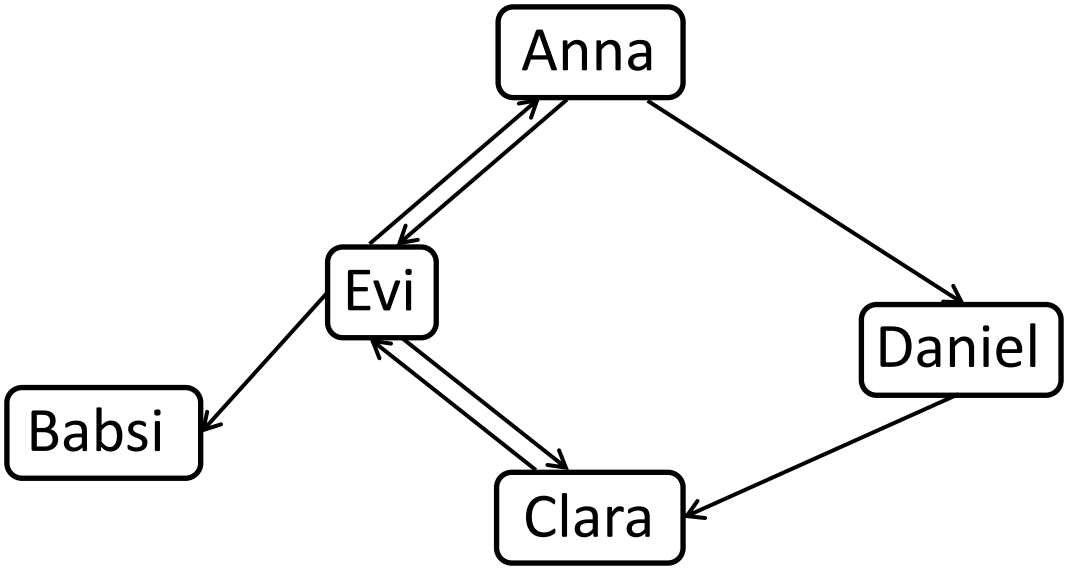
3 Repräsentationen von Graphen



Adjazenzmatrix

	A	B	C	D	E
A					
B					
C					
D					
E					

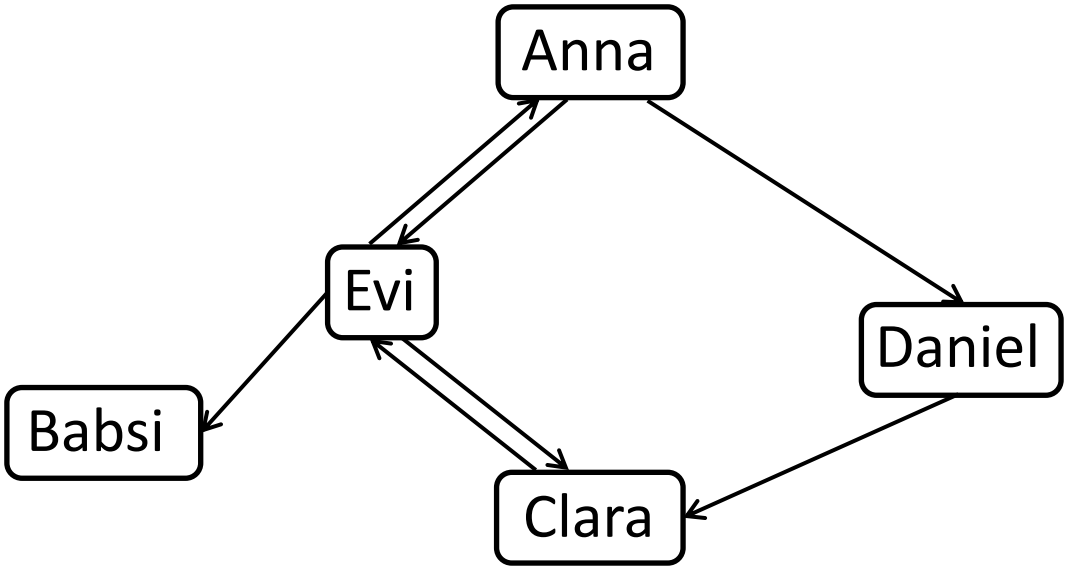
3 Repräsentationen von Graphen



Adjazenzmatrix

	A	B	C	D	E
A				X	X
B					
C					X
D			X		
E	X	X	X		

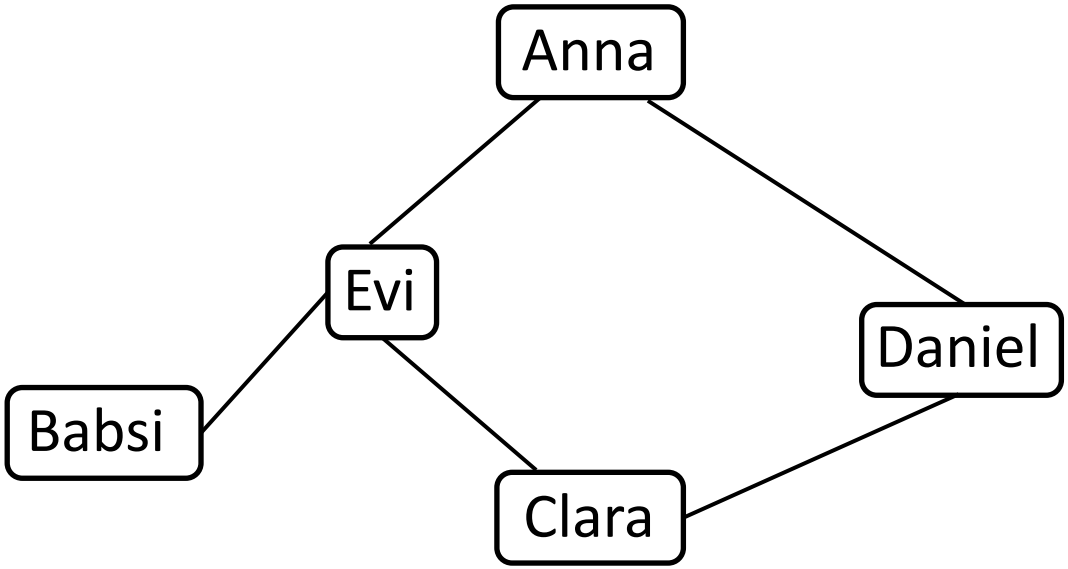
3 Repräsentationen von Graphen



Adjazenzmatrix

	A	B	C	D	E
A	f	f	f	w	w
B	f	f	f	f	f
C	f	f	f	f	w
D	f	f	w	f	f
E	w	w	w	f	f

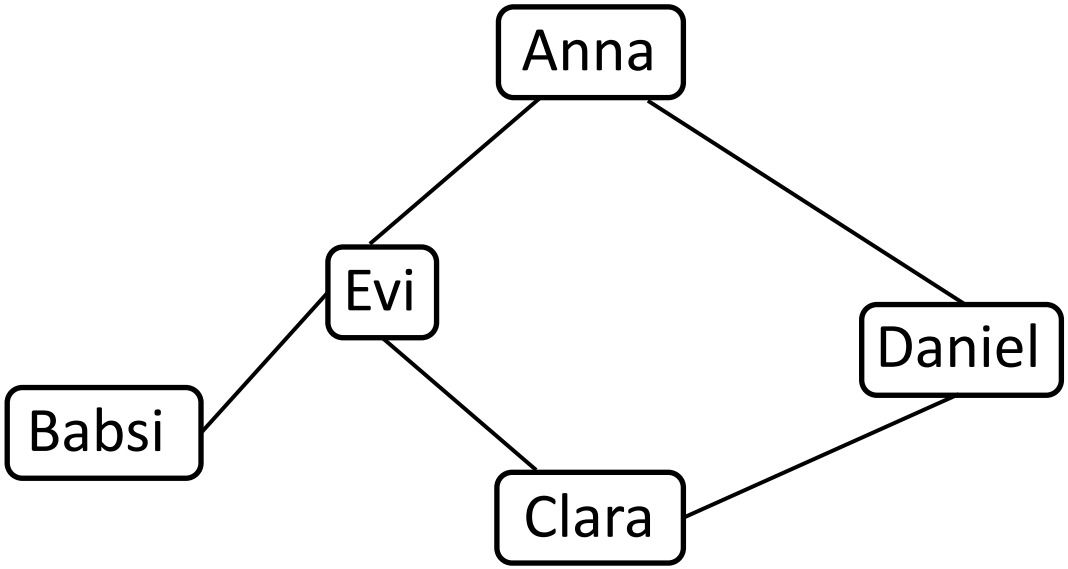
3 Repräsentationen von Graphen



Adjazenzmatrix

	A	B	C	D	E
A					
B					
C					
D					
E					

3 Repräsentationen von Graphen



Adjazenzmatrix

	A	B	C	D	E
A				X	X
B					X
C				X	X
D	X		X		
E	X	X	X		

Die Adjazenzmatrix eines ungerichteten Graphen ist symmetrisch.

3 Repräsentationen von Graphen

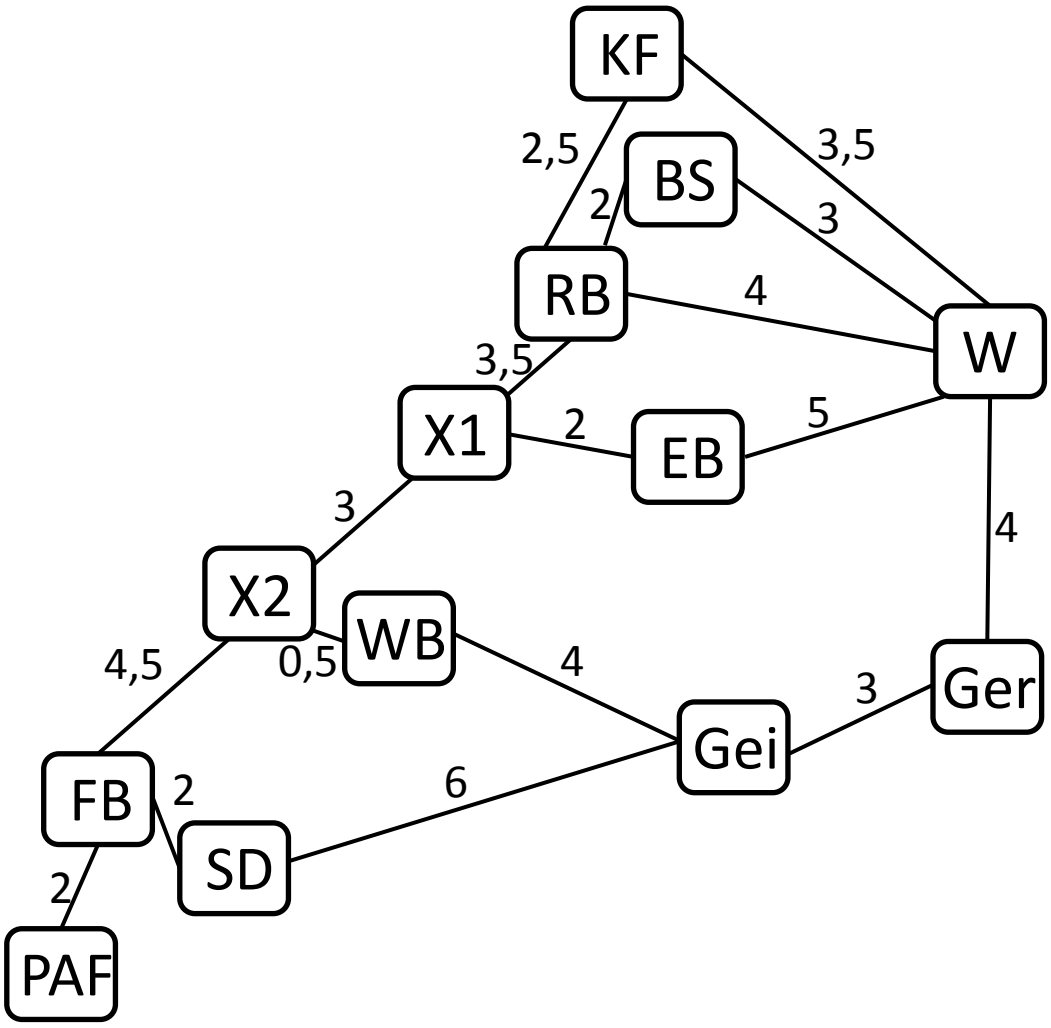
Knotenfeld

knotenindex	0	1	2	3	4
inhalt	A	B	C	D	E

Adjazenzmatrix

	0	1	2	3	4
0				X	X
1					X
2				X	X
3	X		X		
4	X	X	X		

Die Adjazenzmatrix eines ungerichteten Graphen ist symmetrisch.

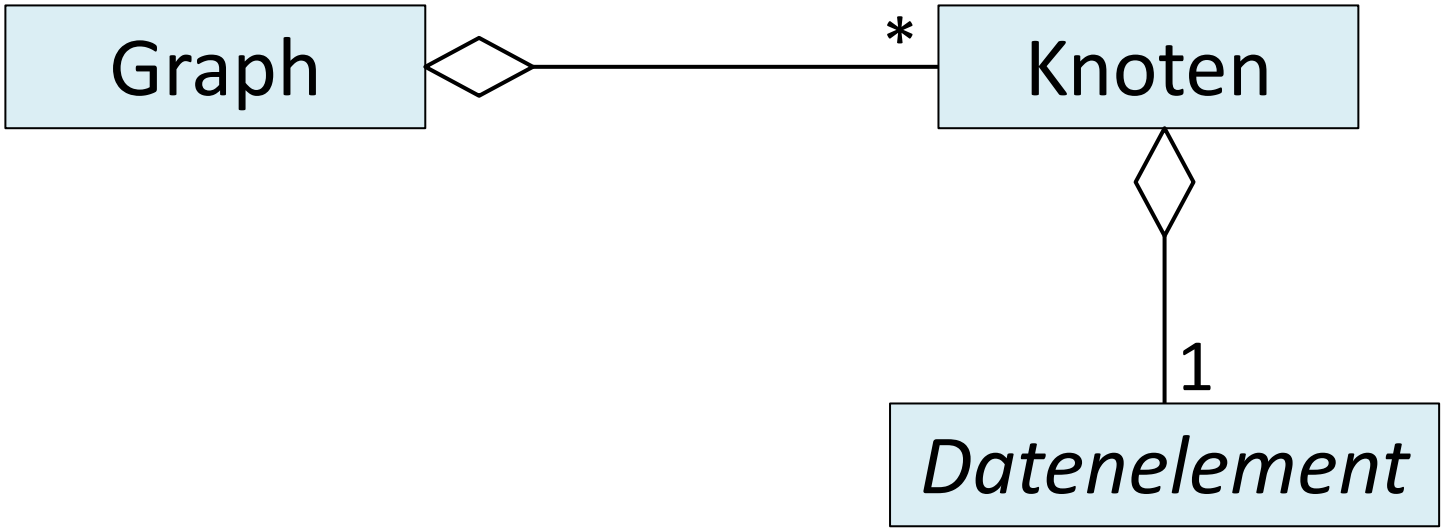

KF
BS
RB
W
X1
EB
X2
WB
Ger
Gei
FB
SD
PAF
2,5
3,5
2
3
4
3,5
2
5
3
4
4,5
0,5
4
3
2
6
2

	BS	EB	FB	GEI	GER	KF	PAF	RB	SD	W	WB	X1	X2
BS													
EB													
FB													
GEI													
GER													
KF													
PAF													
RB													
SD													
W													
WB													
X1													
X2													

	BS	EB	FB	GEI	GER	KF	PAF	RB	SD	W	WB	X1	X2
BS	0							2		3			
EB		0								5		2	
FB			0				2		2				4,5
GEI				0	3				6		4		
GER				3	0					4			
KF						0		2,5		3,5			
PAF			2				0						
RB	2					2,5		0		4		3,5	
SD			2	6					0				
W	3	5			4	3,5		4		0			
WB				4							0		0,5
X1		2						3,5				0	3
X2			4,5								0,5	3	0

Klassenkarte

Graph
Knoten[] knoten boolean[][] adjazenzmatrix int maxAnzahl int anzahl
Graph(int m) knotenEinfuegen(Knoten k) kanteEinfuegen(int i, int j) knotenEntfernen(Knoten k) kanteEntfernen(int i, int j) knotenindexSuchen(Knoten k) adjazenzmatrixAusgeben() knotenlisteAusgeben()



Der Knoten kennt bei Implementierung mit einer Adjazenzmatrix seine Nachbarn nicht!

S. 103 / 1 und 2

- Implementiere die Klasse Ortschaft (Name, Abkürzung), die abstrakte Klasse Datenelement und die Klasse Knoten.
- Implementiere die Klasse Graph (ohne Knotenentfernen)
- Teste das Programm in einer Testklasse

```
public class Ortschaft extends Datenelement {

    private String name;
    private String abkuerzung;

    public Ortschaft(String n, String a){
        name = n;
        abkuerzung = a;
    }

     public void datenAusgeben(){
         System.out.println(name +", " + abkuerzung);
     }
}
```

```
public abstract class Datenelement{
    public abstract void datenAusgeben();
}

public class Knoten{

    private Datenelement inhalt;

    public Knoten(Datenelement inh){
        inhalt = inh;
    }

    public Datenelement inhaltGeben(){
        return inhalt;
    }
}
```

```
public class Graph{

   private Knoten[] knoten;
   private boolean[][] adjazenzmatrix;
   int maxAnzahl;
   int anzahl;

   public Graph(int m){
      knoten = new Knoten[m];
      adjazenzmatrix = new boolean[m][m];
      maxAnzahl = m;
      anzahl = 0;
    }
```

```
public void knotenEinfuegen(Knoten k){
        if(anzahl < maxAnzahl){
            knoten[anzahl]=k;
            anzahl++;
        }
        else {
            System.out.println("Graph voll belegt!");
        }
    }
```

```
public void kanteEinfuegen(int i, int j){
      if (i<anzahl && j<anzahl) {
         adjazenzmatrix[i][j] = true;
      }
      else {
         System.out.println("Kante kann nicht eingefuegt
                                        werden!");
      }
   }
```

Bei ungerichtetem Graph kann auch gleich

```
         adjazenzmatrix[j][i] = true;
```

ergänzt werden.

```
public void kanteEntfernen(int i, int j){
    if (i<anzahl && j<anzahl && adjazenzmatrix[i][j]) {
        adjazenzmatrix[i][j] = false;
    }
    else {
        System.out.println("Hier ist keine Kante
                                    vorhanden!");
    }
}
```

```
public int knotenindexSuchen(Knoten k){
      int index = -1;
      int zaehler = 0;
      while (index < 0 && zaehler < anzahl){
         if (knoten[zaehler].equals(k)){
            index = zaehler;
         }
         zaehler++;
      }
      if (index<0) {System.out.println("Knoten nicht
                                          vorhanden!");}
      return index;
   }
```

```
public void adjazenzmatrixAusgeben(){
        System.out.println("Adjazenzmatrix:");
        System.out.print("  ");
        for(int i=0; i<anzahl; i++){
            System.out. print(i + " ");
        }
        System.out.println();
        for (int i=0; i<anzahl; i++){
            System.out. print(i + " ");
            for (int j=0; j<anzahl; j++){
                if (adjazenzmatrix[i][j]) System.out.print("X ");
                else System.out.print("- ");
            }
            System.out.println();
        }
    }
```

```
public void knotenlisteAusgeben(){
    for(int i=0; i<anzahl; i++){
        System.out.print("Knoten "+ i + " enthaelt: " );
        knoten[i].inhaltGeben().datenAusgeben();
        System.out.println();
    }
}
```