Aufgabe 1 – modifizierte Bedingungs-/Entscheidungsüberdeckung

Bei dem Verfahren der modifizierten Bedingungs-/Entscheidungsüberdeckung werden für das Prüfen des Ausdrucks `(a | !b) & c & !d` mindestens **fünf** Testfälle benötigt. Geben Sie **nur** in genau solchen **fünf** Zeilen, die eine entsprechende Kombination von Eingabewerten darstellen, das Ergebnis der Auswertung von `(a | !b) & c & !d` an.

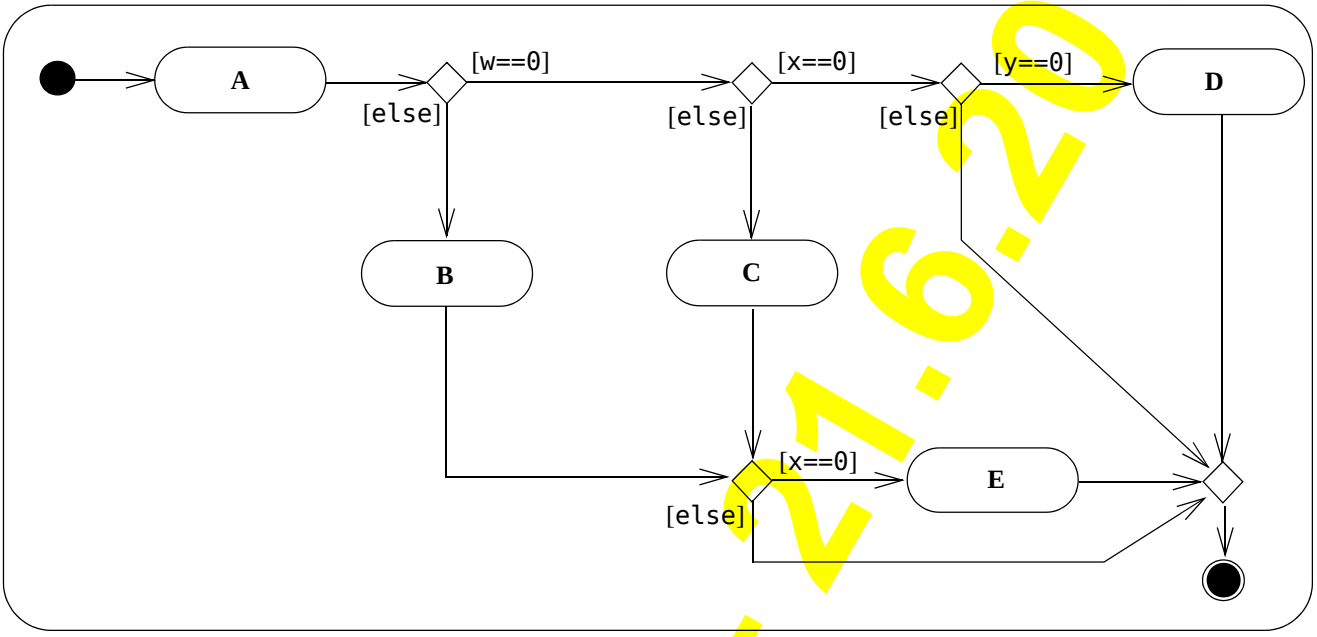
a	b	c	d	(a | !b) & c & !d
false	false	false	false	
false	false	false	true	
false	false	true	false	
false	false	true	true	
false	true	false	false	
false	true	false	true	
false	true	true	false	
false	true	true	true	
true	false	false	false	
true	false	false	true	
true	false	true	false	
true	false	true	true	
true	true	false	false	
true	true	false	true	
true	true	true	false	
true	true	true	true	

Zwei Lösungen sind möglich. Die zu benutzenden Zeilen sind farbig markiert.

In den entsprechenden Zeilen müssen dann true oder false berechnet und angegeben werden.

Aufgabe 2 – strukturorientierter Test

Gegeben ist das folgende Aktivitätsdiagramm:



Bestimmen Sie Testfälle. Geben Sie jeweils eine ***minimale*** Anzahl von Wertebelegungen für die `int`-Variablen `w`, `x` und `y` an, so dass jeweils eine *möglichst gute* Anweisungsüberdeckung, Zweigüberdeckung und Pfadüberdeckung erreicht werden. Geben Sie für jeden Testfall zusätzlich den Pfad als Folge der durchlaufenen Aktionen an. Die Werte der Variablen `w`, `x` und `y` werden in den Aktionen **A**, **B**, **C**, **D** und **E** ***nicht*** geändert.

a) Anweisungsüberdeckung:

w	x	y	Pfad als Folge der Aktionen
0	0	0	AD
1	0		ABE
0	1		AC

Der Überdeckungsgrad mit den Testfällen unter a) für die Anweisungsüberdeckung ist: **1**

b) Zweigüberdeckung (***nur*** Testfälle aufführen, die nicht unter a) aufgeführt sind):

w	x	y	Pfad als Folge der Aktionen
0	0	1	A

Der Überdeckungsgrad mit den Testfällen unter a) und b) für die Zweigüberdeckung ist: **1**

c) Pfadüberdeckung (***nur*** Testfälle aufführen, die nicht unter a) und b) aufgeführt sind):

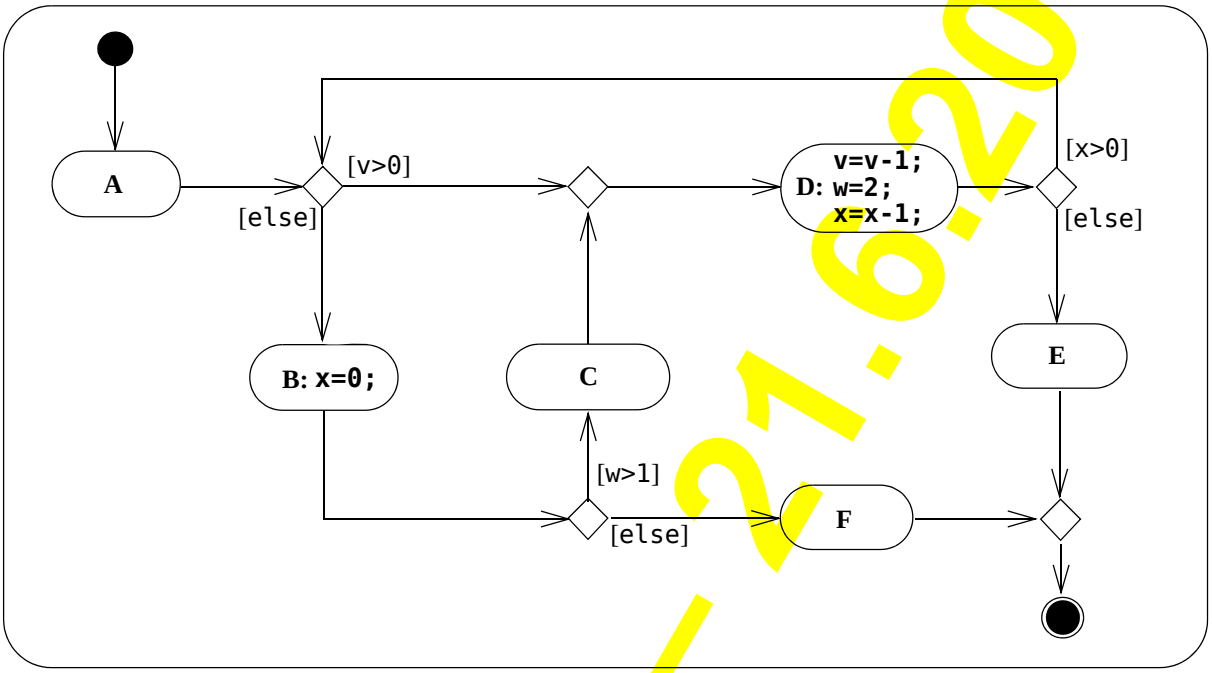
w	x	y	Pfad als Folge der Aktionen
1	1		AB

Der Überdeckungsgrad mit den Testfällen unter a), b) und c) für die Pfadüberdeckung ist: **5/6**

ACE nicht möglich

Aufgabe 3 – strukturorientierter Test

Gegeben ist das folgende Aktivitätsdiagramm:



Alle Aktionen enthalten neben den angeführten Zuweisungen (in **B** und **D**) weitere Operationen, die beim Testen ausgeführt werden sollen, im Detail aber nicht wichtig und daher im Diagramm nicht aufgeführt sind. Insbesondere werden die Werte der Variablen `v`, `w` und `x` ***nur*** durch die angegebenen Zuweisungen geändert.

Geben Sie eine ***minimale*** Anzahl von Testfällen als Wertebelegungen für die `int`-Variablen v, w und x an, so dass die *vollständige strukturierte Pfadüberdeckung mit* ***k=3*** die ***maximal*** mögliche Überdeckung erreicht.

Geben Sie für jeden Testfall zusätzlich den Pfad als Folge der Bezeichnungen der durchlaufenen Aktionen an.

v	w	x	Pfad als Folge der Aktionen
-1	-1		ABF
-1	1		ABCE
1		1	ADE
2		2	ADDE
3		3	ADDDE
1		2	ADBCDE
2		3	ADDBCDE

Aufgabe 4 – Entwurfsmuster Adapter

Eine Anwendung erwartet ein Objekt, dass zu dem Interface `Expected` kompatibel ist.
Die durch `Expected` vorgegebenen Methoden sollen folgende Wirkung haben:

- Die Methode `length` gibt die Anzahl der in der Datenstruktur abgelegten Objekte zurück.
- Die Methode `add` soll das als Parameter übergebene Objekt dann in die Datenstruktur einfügen, falls es noch nicht in der Datenstruktur verfügbar ist.
- Die Methode `getAll` soll alle Objekte der Datenstruktur in einem Feld zurückgeben.

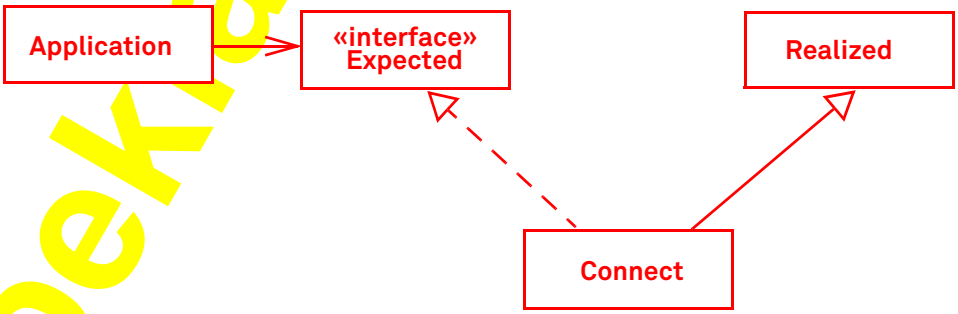
Die Klasse `Realized` liegt bereits als Implementierung mit folgenden Methoden vor:

- Der Konstruktor legt ein neues Objekt der Klasse `Realized` an.
- Die Methode `contains` gibt `true` zurück, falls das Objekt `o` in der Datenstruktur abgelegt ist. Existiert `o` nicht in der Datenstruktur, wird der `false` zurückgegeben.
- Die Methode `add` fügt das als Parameter übergebene Objekt in die Datenstruktur ein. Die Datenstruktur passt sich in ihrer Größe dynamisch an.
- Die Methode `getAll` gibt alle Objekte der Datenstruktur in einem Feld zurück.

```
public interface Expected {
     public abstract int length();
     public abstract void add( Object o );
     public abstract Object[] getAll();
}
```

```
public class Realized {
     public Realized() {...}
     public boolean contains( Object o ) {...}
     public void add( Object o ) {...}
     public Object[] getAll() {...}
}
```

a) Geben Sie ein UML-Klassendiagramm an, bei dem die Klasse `Connect` einen ***Klassenadapter*** bereitstellt, der `Realized` nutzt, um die Anforderungen von `Expected` zu erfüllen.



Aufgabe 4 – Entwurfsmuster Adapter (Fortsetzung)

b) Geben Sie die Java-Implementierung einer Klasse `Connect` an, die einen ***Klassenadapter*** bereitstellt, der `Realized` nutzt, um die Anforderungen von `Expected` zu erfüllen.

```
public interface Expected {
     public abstract int length();
     public abstract void add( Object o );
     public abstract Object[] getAll();
}
```

```
public class Realized {
     public Realized() {...}
     public boolean contains( Object o ) {...}
     public void add( Object o ) {...}
     public Object[] getAll() {...}
}
```

```
public class Connect extends Realized implements Expected {

    private int size;

    public int length() {
        return size;
    }

    public void add(Object o) {
        if (!contains(o)) {
            super.add(o);
            size++;
        }
    }

}
```

alternativ OHNE zusätzliches Attribut size:

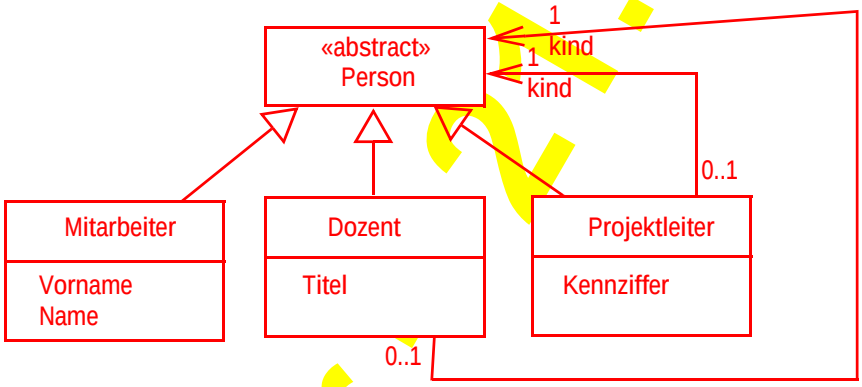
```
public int length() {
    return getAll().length;
}
```

Aufgabe 5 – Entwurfsmuster Dekorierer

Die Rollen einer an einer Universität tätigen Person sollen durch ein ***Dekorierer***-Muster modelliert werden, das die im Folgenden beschriebenen Beziehungen zwischen verschiedenen Rollen geeignet umsetzt.

- Eine Person hat als Mitarbeiter einen Vornamen und einen Namen.
- Eine Person kann verschiedene Vorlesungen als Dozent halten. Jede Vorlesung hat einen Titel.
- Eine Person kann verschiedene Projekte als Leiter durchführen. Jedes Projekt hat eine Kennziffer.

a) Geben Sie ein geeignetes UML-Klassendiagramm für das Dekorierer-Muster an. Geben Sie für die Attribute ***keine*** Datentypen an.



b) Für das oben modellierte Dekorierer-Muster soll eine Methode `int anzahlVorlesungen()` angelegt werden. Die Methode `anzahlVorlesungen` soll für einen mit dem Dekorierer-Muster beschriebene Person die Zahl der von ihr gehaltenen Vorlesungen liefern. Skizzieren Sie für jede der von Ihnen vorgesehenen Klassen entweder als informale Beschreibung oder als Java-Programmcode, wie die Methode `anzahlVorlesungen` implementiert werden müsste.

in der Klasse Mitarbeiter: `int anzahlVorlesungen() { return 0; }`

in der Klasse Dozent: `int anzahlVorlesungen() { return 1 + kind.anzahlVorlesungen(); }`

in der Klasse Projektleiter: `int anzahlVorlesungen() { return kind.anzahlVorlesungen(); }`

Aufgabe 6 – Entwurfsmuster Strategie

Die Klasse `Data` und das Interface `Strategy` setzen gemeinsam das ***Strategie***-Muster um.

```
public interface Strategy {
     int apply( int x, int y );
}
```

```
public class Data {
     private int[] arr;
     private Strategy s;
     public Data( int n, Strategy s ) {
          arr = new int[n];
          this.s = s;
     }
     public void manipArr() {
          for ( int k=0; k<arr.length; k++ ) {
               arr[k] = s.apply( arr[k], k );
          }
     }
}
```

a) Implementieren Sie eine Strategie `Init`, so dass bei Anwendung dieser Strategie die Methode `manipArr` den Wert jedes `arr[i]` auf den Wert `2*i*w` setzt.
Der **`int`**-Wert `w` soll beim Erzeugen eines jeden `Init`-Objektes angegeben werden können.

```
public class Init implements Strategy {
     private int w;
     public Init(int w) { this.w = w; }
     public int apply(int x, int y) {
          return 2*y*w;
     }
}
```

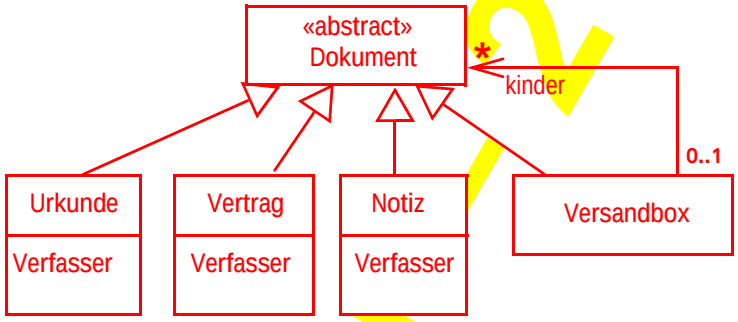
b) Implementieren Sie eine Strategie `SetSum`, so dass bei Anwendung dieser Strategie die Methode `manipArr` den Wert jedes `arr[i]` auf die Summe der Werte von einschließlich `arr[0]` bis einschließlich `arr[i]` setzt.

```
public class SetSum implements Strategy {
     private int sum;
     public int apply(int x, int y) {
          sum +=x;
          return sum;
     }
}
```

Aufgabe 7 – Entwurfsmuster Kompositum

a) Zeichnen Sie ein Klassendiagramm für ein ***Kompositum***-Muster, das die im Folgenden beschriebenen Beziehungen zwischen verschiedenen Dokumenten geeignet umsetzt:

> Eine Akte kann Urkunden, Verträge, Notizen oder andere Akten enthalten.
> Jede Urkunde, jeder Vertrag und jede Notiz hat genau einen Verfasser.



b) Für das oben modellierte Kompositum-Muster soll eine Methode `int anzahlVerfasser()` angelegt werden. Die Methode `anzahlVerfasser` soll für ein Dokument die Zahl der an ihm beteiligten Verfasser liefern. Skizzieren Sie für jede der von Ihnen vorgesehenen Klassen entweder als informale Beschreibung oder als Java-Programmcode, wie die Methode `anzahlVerfasser` implementiert werden müsste.

Die Aufgabe ist etwas unglücklich, da in der Realität ein Verfasser mehrere Dokumente erstellt haben könnte. Letztlich ergibt sich aber die Lösung eindeutig als Zählen der Dokumente, da über die int-Rückgabe keine andere Information transportiert werden kann.

in der Klasse Urkunde:
```
public int anzahlVerfasser() {return 1;}
```

in der Klasse Vertrag:
```
public int anzahlVerfasser() {return 1;}
```

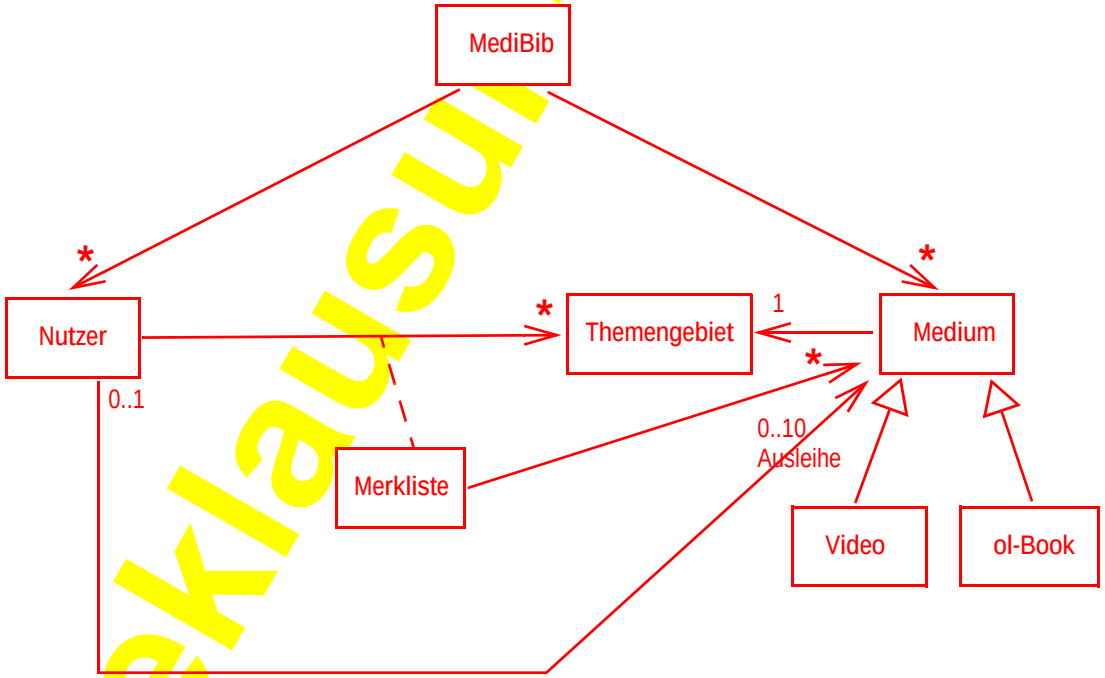
in der Klasse Notiz:
```
public int anzahlVerfasser() {return 1;}
```

in der Klasse Akte:
```
public int anzahlVerfasser() {
        int sum=0;
        for( Dokument kind : kinder ) {
                sum += kind.anzahlVerfasser();
        }
        return sum;
}
```

Aufgabe 8 – Klassendiagramm

Erstellen Sie ein Klassendiagramm für die Entwicklung einer Medienbibliothek (MediBib), die die folgenden Eigenschaften modellieren soll. Geben Sie für die Attribute ***keine*** Datentypen an.

- MediBib verwaltet Nutzer und Medien.
- Jeder Nutzer besitzt einen Namen und eine Nutzernummer.
- Jedes Medium hat einen Titel.
- Spezielle Medien sind Videos mit einer Format-Angabe und OnlineBooks mit einer Seitenzahl.
- Jedem Medium ist genau ein Themengebiet zugeordnet.
- Jeder Nutzer kann sich für beliebig viele Themengebiete interessieren.
- Für jedes Themengebiet ist jedem Nutzer eine Merkliste mit einer beliebigen Zahl Medien zugeordnet.
- Jeder Nutzer kann bis zu 10 Medien ausleihen. Ein Medium kann immer nur an einen Nutzer ausgeliehen werden.

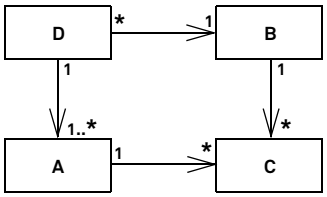


Attribute müssen noch ergänzt werden

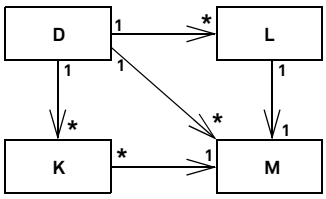
Aufgabe 9 – Objektdiagramm

Gegeben sind die folgenden fünf Klassendiagramme. Geben Sie rechts neben jedem dieser Klassendiagramme ein Objektdiagramm an, das **genau fünf** Objekte enthält, von denen mindestens eines zur Klasse `D` gehört.

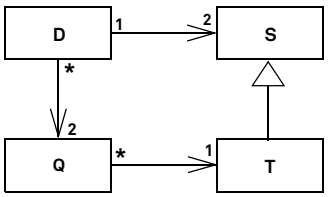
Die Kanten müssen in der Klausur gezeichnet werden.
Hier sind nur alle denkbaren Kombinationen von Objekten aufgeführt.



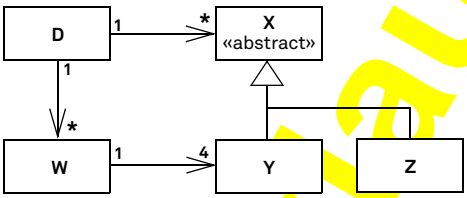
D AABC
D AAAB
D ABCC



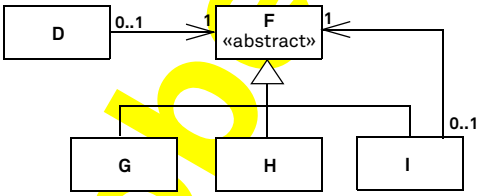
D DDDD
D DDLM
D LMLM
D DKML



D QQTT
D QSTT



D DDDD
D DDDZ
D DDZZ
D DZZZ
D ZZZZ



D IIIG
D IIIH
(Dekorierer)

auch erlaubt sind nach dem Klassendiagramm nicht zusammenhängende Objektdiagramme wie DG und drei einzelne G-Objekte: G G G usw.