UML - Sequenzdiagramme

Statische vs dynamische UML-Diagramme

Statische Modellierung

Bisher wurde das statische Verhalten eines Softwaresystems in Form von Klassen- und Objektdiagramm betrachtet.

Dynamische Modellierung

Eine Möglichkeit der dynamischen Modellierung wäre das Sequenzdiagramm, bei dem die Interaktion zwischen Objekten in der zeitlichen Abfolge betrachtet wird.

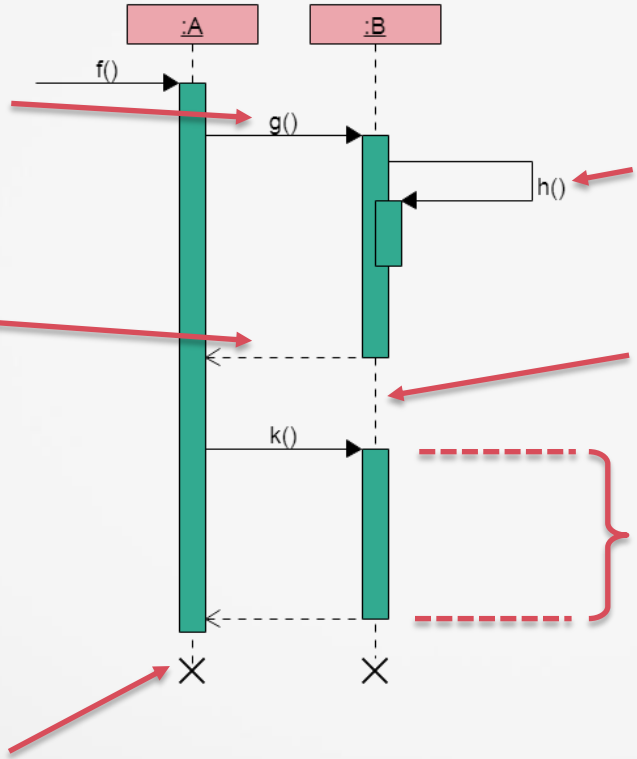
Übersicht anhand eines Beispiels

Objekte

Methodenaufruf
=Nachricht

Rückkehr zur
aufrufenden
Methode

Zeitlicher
Ablauf



Methodenaufruf innerhalb der Klasse=
Nachricht des Objektes an sich selbst:
Rekursion

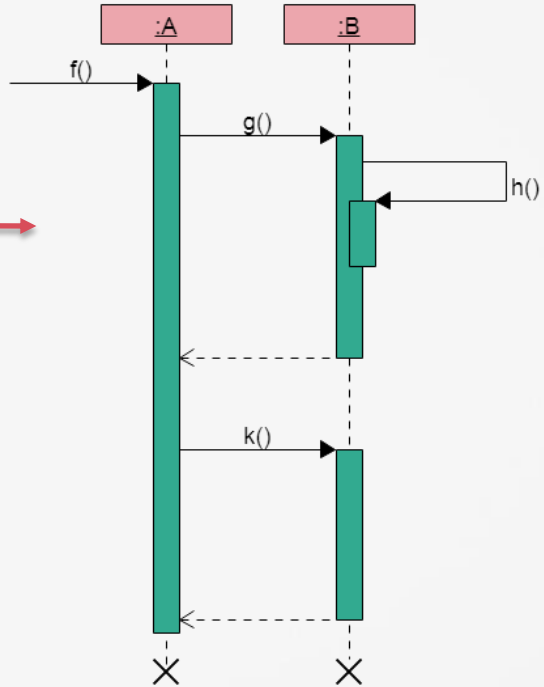
Zeitlinie

Zeitliche Dauer des Methodenaufrufes
von k: Aktivitätsdauer/balken

Löschen des Objektes: Ende der Lebenslinie

Beispiel I: Sequenzdiagram von Methode f

```
class A {
  B b=new B();
  void f() {
    b.g();
    b.k();
  }
}
class B {
  void g() {
    h();
  }
  void h() { }
  void k() { }
}
```

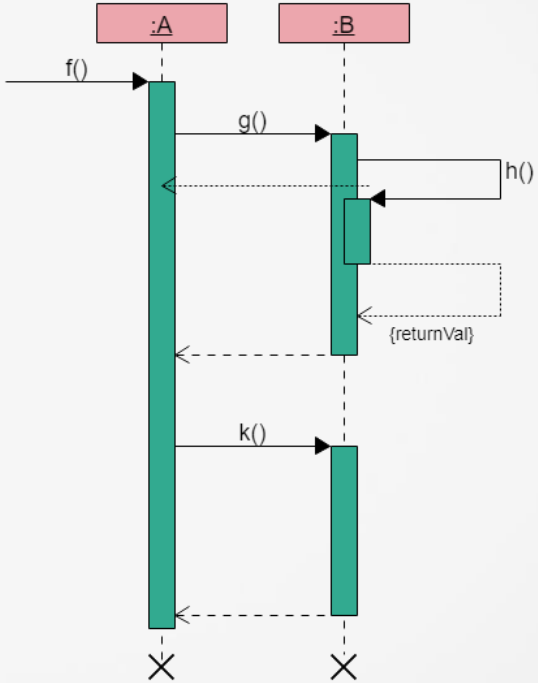


Quellcode

Sequenzdiagramm

Beispiel I: Sequenzdiagram von Methode f

```
class A {
  B b=new B();
  void f() {
    b.g();
    b.k();
  }
}
class B {
  void g() {
    int returnVal=h();
  }
  int h() { }
  void k() { }
}
```



Quellcode

Sequenzdiagramm

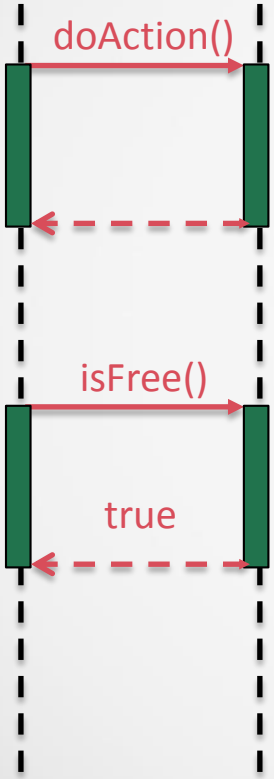
Regeln: Ablauf

- ✓ Sequenzdiagramme beschreiben die Kommunikation zwischen Objekten in einer bestimmten Szene.
- ✓ Es wird beschrieben welche Objekte an der Szene beteiligt sind, welche Informationen (Nachrichten) sie austauschen und in welcher zeitlichen Reihenfolge der Informationsaustausch stattfindet.
- ✓ Sequenzdiagramme enthalten eine implizite Zeitachse. Die Zeit schreitet in einem Diagramm von oben nach unten fort.
- ✓ Die Reihenfolge der Pfeile in einem Sequenzdiagramm gibt die zeitliche Reihenfolge der Nachrichten an.

Regeln: Nachricht, Botschaft

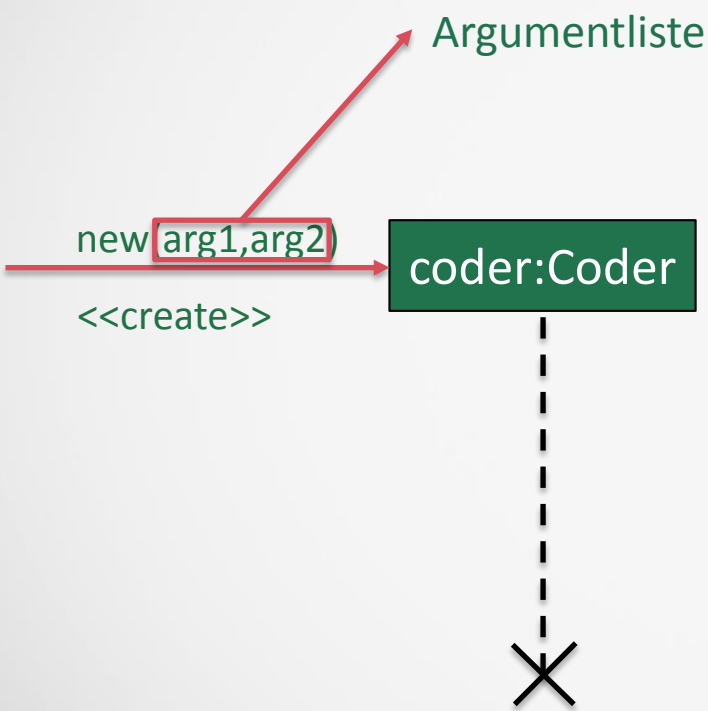
- ✓ Objekte kommunizieren über Nachrichten.
- ✓ Nachrichten werden als Pfeile zwischen den Aktivierungen eingezeichnet.
- ✓ Der Name der Nachricht steht an dem Pfeil.
- ✓ Eine Nachricht liegt immer zwischen einem sendenden und einem empfangenden Objekt.

Regeln: Synchrone Nachricht



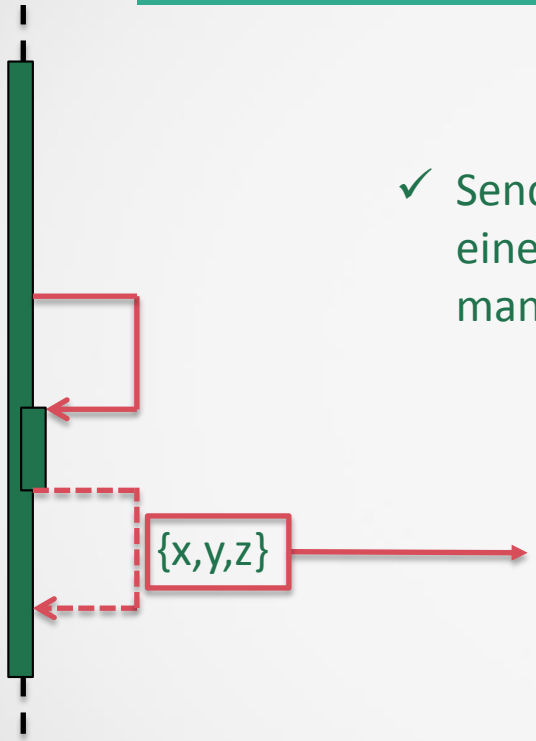
- ✓ Der Pfeil mit der ausgefüllten Spitze bezeichnet einen synchronen Aufruf.
- ✓ Der Aufruf erfolgt von der Quelle zum Ziel, d. h. die Zielklasse muss eine entsprechende Methode implementieren.
- ✓ Die Quelle wartet mit der weiteren Verarbeitung bis die Zielklasse ihre Verarbeitung beendet hat und setzt die Verarbeitung dann fort.
- ✓ Ein Aufruf muss einen Namen haben; in runden Klammern können Aufrufparameter angegeben werden.
- ✓ Der gestrichelte Pfeil ist der Return.
- ✓ Die Bezeichnung des Return mit einem Namen ist optional.

Regeln: Erstellen und Zerstören von Objekten



- ✓ Das Erzeugen eines Objektes wird durch eine Nachricht, die im Kopf des Objekts endet dargestellt (gestrichelte Linie).
- ✓ Das Zerstören (Löschen) eines Objektes wird durch ein x auf der
- ✓ Lebenslinie markiert.

Regeln: Aufruf von Methoden der eigenen Klasse

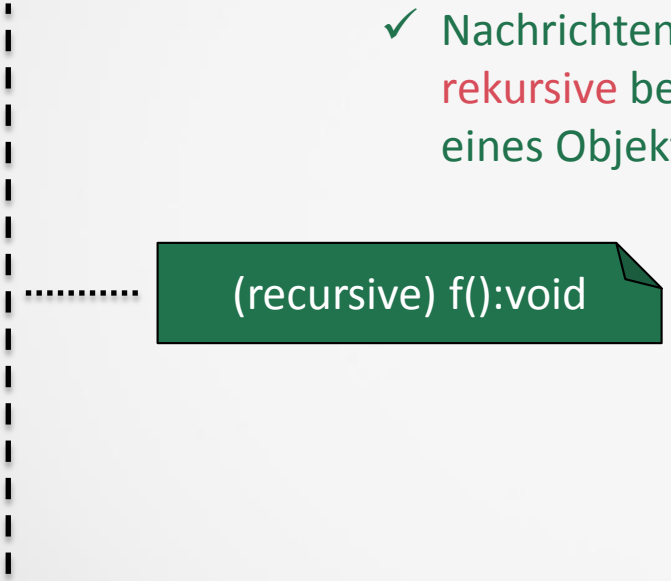


- ✓ Sendet ein Objekt eine Nachricht an sich selbst, so ruft es eine Methode auf, die es selbst implementiert. Dies nennt man „Rekursion“ bei einem Objekt!

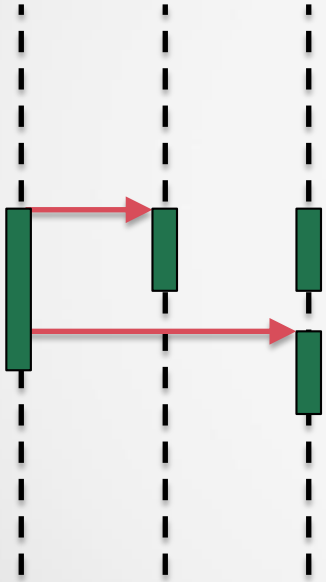
Mehrere Rückgabewerte sind NICHT möglich!

Regeln: Rekursiver Aufruf

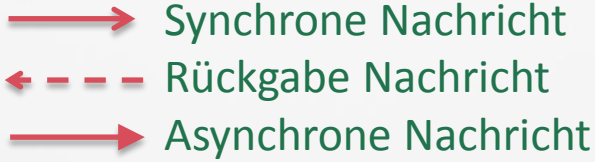
- ✓ Nachrichten eines Objektes an sich selbst werden oft als rekursive bezeichnet, hier ruft aber lediglich eine Methode eines Objektes eine Methode des gleichen Objektes auf!



Regeln: Asynchrone Nachricht

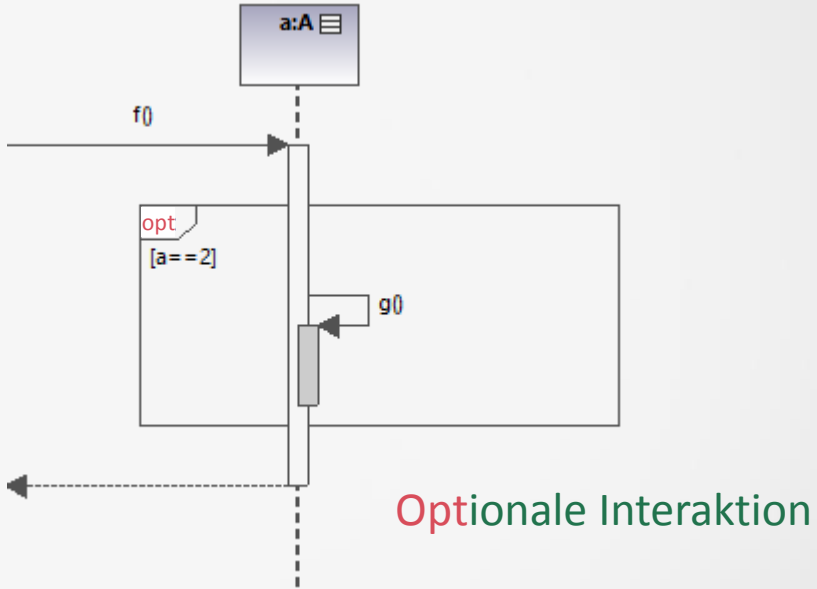


- ✓ Mit einer offenen Pfeilspitze werden asynchrone Nachrichten gekennzeichnet.
- ✓ Der Aufruf erfolgt von der Quelle zum Ziel.
- ✓ Die Quelle wartet mit der Verarbeitung nicht auf die Zielklasse, sondern setzt ihre Arbeit nach dem Senden der Nachricht fort.
- ✓ Asynchrone Aufrufe werden verwendet, um parallele Threads zu modellieren.



Beispiele: Bedingungen I

```
class A {
  void f() {
    int a=0;
    if (a==2)
       g();
  }
  void g() {
  }
}
```



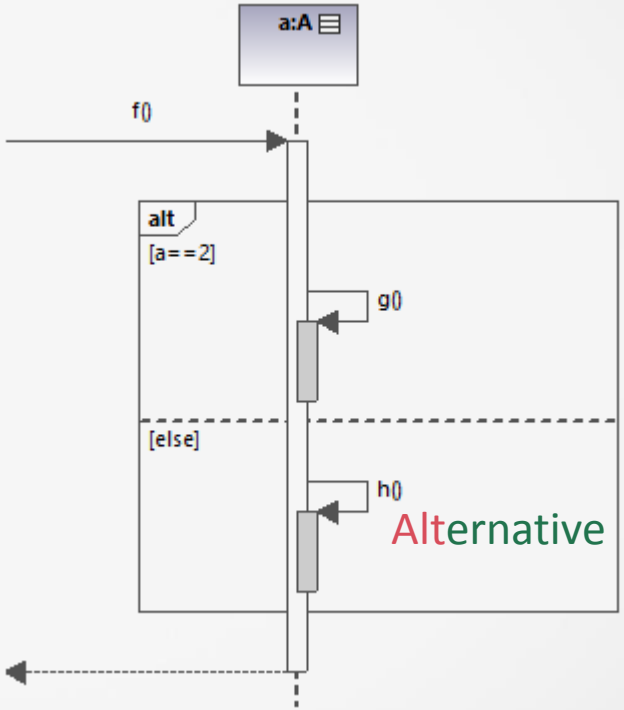
Optionale Interaktion

Quellcode

Sequenzdiagramm

Beispiele : Bedingungen II

```
class A {
  void f() {
    int a=0;
    if (a==2)
       g();
    else
       h();
  }
  void g() {
  }
  void h() {
  }
}
```

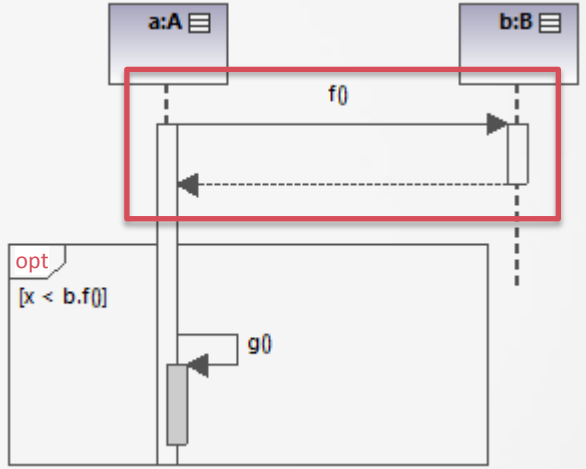


Quellcode

Sequenzdiagramm

Beispiele : Bedingungen III

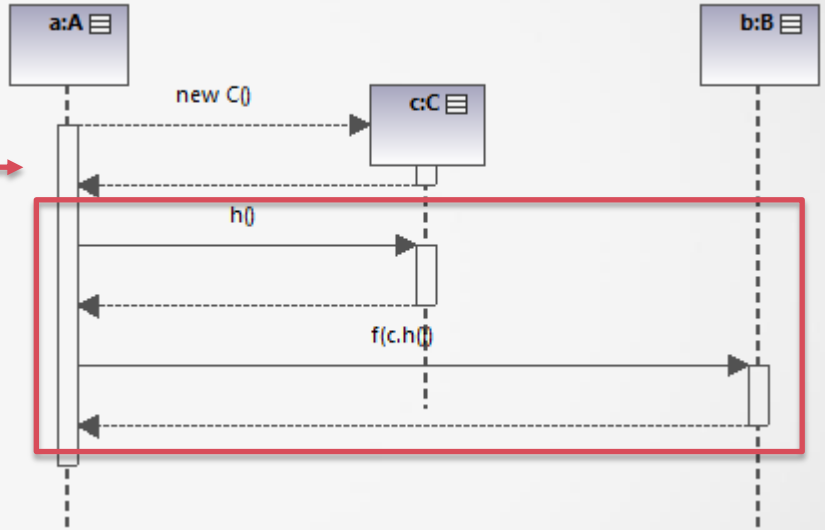
```
public class A {
  public void
start(B b, int x) {
    int a=0;
    if ( x < b.f() )
      g();
  }
  public void g(){
  }
}
public class B {
  public  int f() {
    return 0;
  }
}
```



Funktionsaufruf in der Bedingung

Beispiele : Verschachtelte Funktionsaufrufe

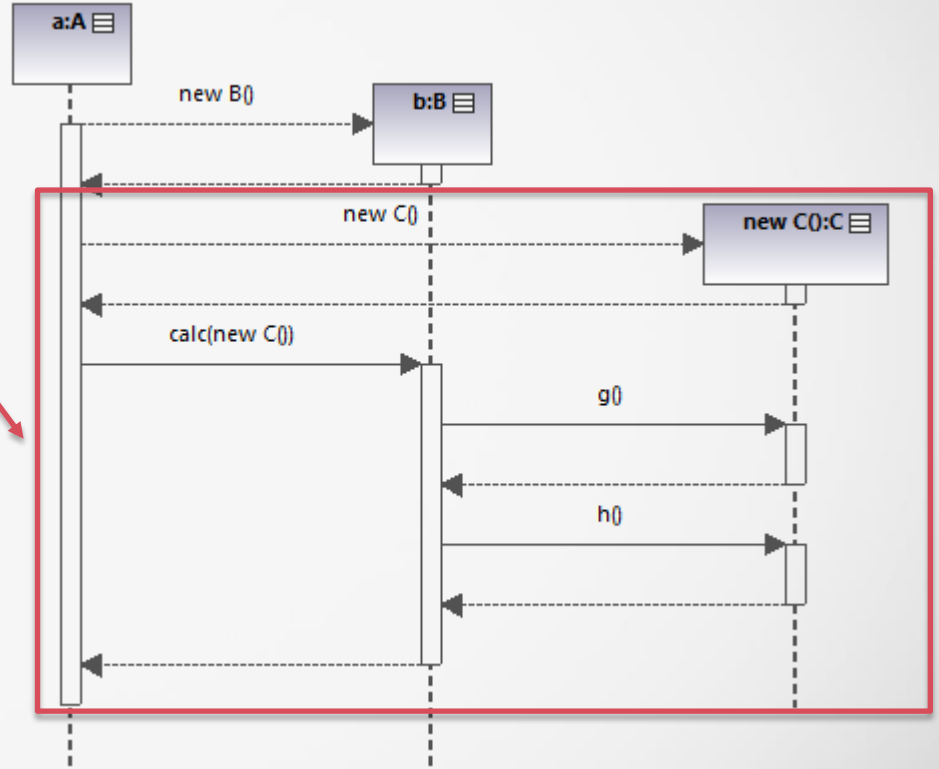
```
public class A {
  public void start(B b, int x) {
    C c=new C();
    int a=b.f(c.h());
  }
}
public class B {
  public  int f(int x) {
    return 0;
  }
}
public class C {
  public  int h() {
    return 0;
  }
}
```



Innerer Funktionsaufruf wird vor dem äußeren Aufruf dargestellt! Ebenso: beiden Funktionen sind in der gleichen Klasse!

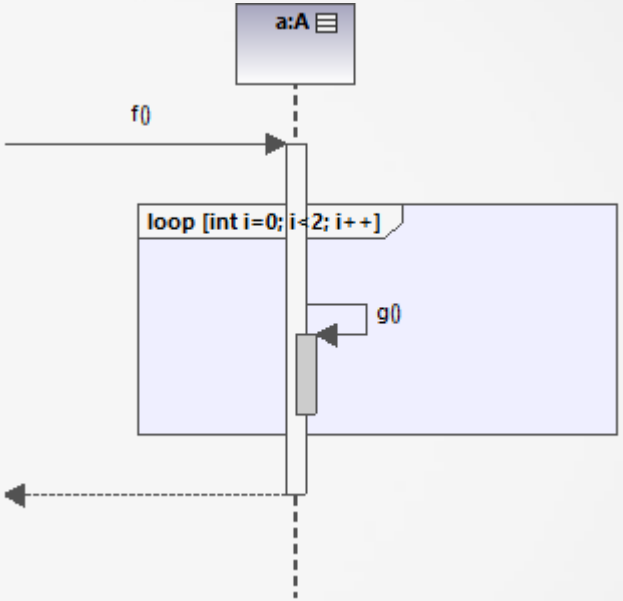
Beispiele : Erzeugen von neuen Objekten bei Funktionsaufruf

```
public class A {
  public void start() {
    B b=new B();
    nt x=b.calc(new C());
  }
}
public class B {
  public  int calc(C c) {
    return c.g()+c.h();
  }
}
public class C {
  public  int g() {
    return 0;
  }
  public  int h() {
    return 0;
  }
}
```



Beispiele : Wiederholungen

```
class A {
  void f() {
    for (int i=0; i<2;i++)
      g();
  }
  void g() {
  }
}
```



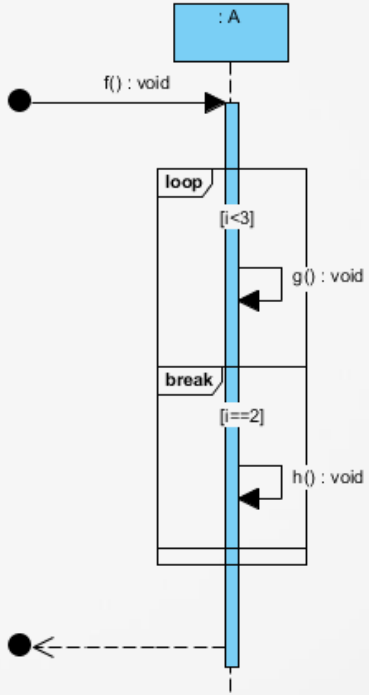
Quellcode

Sequenzdiagramm

Beispiele : Break in Wiederholungen

```
class A {
  void f() {
    int i=0;
    while (i<3){
      g();
      if (i==2) {
        h();
        break;
      }
      i++;
      }
   }
  void g() {
  }
  void h() {
  }
}
```

Quellcode

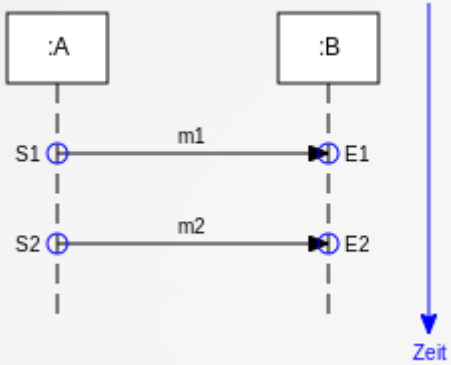


Sequenzdiagramm

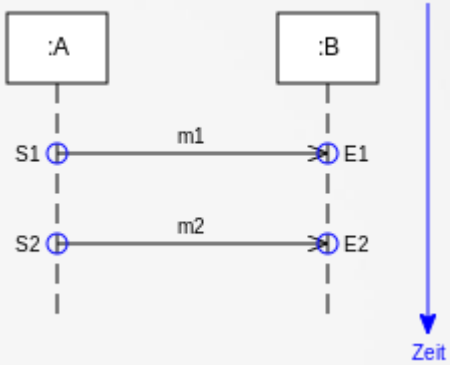
Regeln XI: Übersicht – Verzweigungen und “Schleifen”

Operator	Zweck
alt	If – then - else
opt	If – then
break	Ausnahme Interaktion - break
loop	Wiederholende Interaktion: Es kann nicht zwischen kopf- oder fussgesteuerter Wiederholungsanzweisung unterschieden werden

Synchrone vs asynchrone Nachrichten

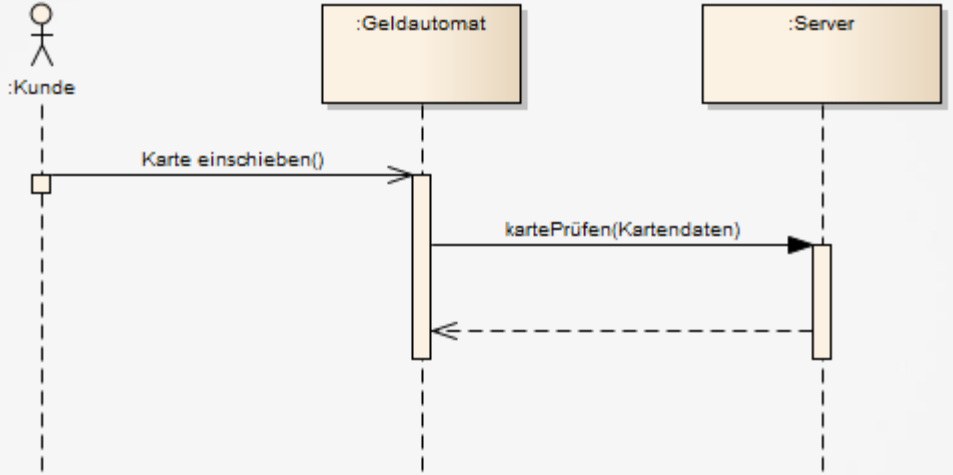


Reihenfolge:
S1,E1,S2,E2



Mögliche Reihenfolgen:
S1, E1, S2, E2
S1, S2, E1, E2
S1, S2, E2, E1

Sequenzdiagramme ohne Java: Beispiel I



Beispiel: Geldautomat

Beispiel IIa: Quellkode I

```
public class Gruppe {
   private List < Person > personen;
   public Gruppe() {
      personen = new Vector < Person > ();
   }
   public void addPerson(Person person) {
      personen.add(person);
   }
   public void removePerson(Person person) {
      personen.remove(person);
   }
}
```

Beispiel IIb: Quellkode II

```
public class SDExample{
   public static void main(String args[]) {
      Person p1 = new Person("p", "1");
      Person p2 = new Person("p", "2");
      Gruppe gruppe = new Gruppe();
      gruppe.addPerson(p1);
      gruppe.addPerson(p2);
      gruppe.removePerson(p1);
   }
}
```

Beispiel IIc: Sequenzdiagram