Schlange

Folge

Schlange ist ein abstrakter Datentyp, der eine Folge von Elementen mit einem sogenannten Frontelement (vorderstem Element) darstellt. Die Operationen sind das Anstellen am Ende der Schlange, das Entfernen des Frontelements, das Zurückgeben des Frontelements und das Testen, ob die Schlange leer ist.

verkettete Liste, Feld
Operationen

Operationen über einem Feld

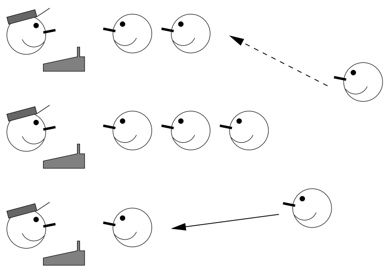
```
package LS8Tools;
import AlgoTools.IO;
public class Schlange {

    private Object[] inhalt;    // Feld fuer Schlangenelemente
    private int head;           // Index fuer Schlangenanfang
    private int count;          // Anzahl Schlangenelemente

    public Schlange (int N) {                   // Konstruktor
        inhalt = new Object[N];      // Platz fuer N Objekte
        head = 0;              // initialisiere Index fuer Anfang
        count = 0;                         // initialisiere Anzahl
    }
    private boolean full () {       // Testet, ob Schlange voll
        return count==inhalt.length;
    }
    public boolean empty () {    // Testet, ob Schlange leer
        return count==0;                       // Anzahl gleich 0?
    }
    public void enq ( Object x ) {       // Fuegt x hinten ein
        if (full ()) IO.error ("in enq: Schlange ist voll!");
        inhalt[ (head+count)%inhalt.length]=x;  // einfuegen
        count++;                          // Anzahl inkrementieren
    }
```

```
public void deq () {              // Entfernt vorderstes Element
        if (empty ()) IO.error ("in deq: Schlange ist leer!");
        head = (head + 1)% inhalt.length;   // neuer Anfang
        count−−;                            // Anzahl dekrementieren
    }
    public Object front () {                // Liefert Element,
        if (empty ()) IO.error ("in front: Schlange ist leer!");
        return inhalt[head];        // welches am Anfang steht
    }
    public int length () {              // Liefert die Anzahl der
        return count;                   // Elemente in der Schlange
    }
    public String toString () {
        int current;                            // Aktueller Index
        int currentCount;               // Aktuelle Elementnummer
        String s;
        current = head;
        currentCount = 0;
        s = new String ("");
        while (currentCount < count) {
            s += ""+ inhalt[current].toString();
            current = (current + 1) % inhalt.length;
            currentCount++;
        }
        return (s);
    }
}
```

Schalterschlange

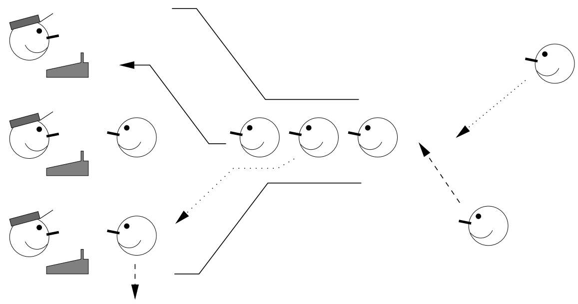


Ein **Mitarbeiter** am Schalter hat eine **Schlange** voller Kunden.

Wenn die Schlange leer ist, meldet er frei,

sonst holt er das Frontelement der Schlange, den nächsten Kunden.

Verteilerschlange



Alle **Mitarbeiter** greifen auf dieselbe **Schlange** voller Kunden zu.

Wenn diese Schlange leer ist, meldet er frei,

sonst holt er das Frontelement der einen Schlange.

Mitarbeiter

```
import AlgoTools.IO;
import LS8Tools.Schlange;                              // ADT Schlange
class Mitarbeiter {
    private String kunde;                   // aktuell bedienter Kunde
    private Schlange warteschlange;                   // Warteschlange
                            // Konstruktor fuer Schalterschlangen
    public Mitarbeiter () {
        kunde = null;
        warteschlange = new Schlange (100);
    }
                            // Konstruktor fuer Verteilerschlangen
    public Mitarbeiter (Schlange _warteschlange) {
        kunde = null;
        warteschlange = _warteschlange;
    }
    public Schlange gibWarteschlange () {
        return (warteschlange); }
    public boolean frei () { return (kunde == null); }
    public void setzeFrei () { kunde = null; }

    public String toString () {
        if (kunde == null) return ("(frei)");
        return (kunde);
    }
```

```
    public void naechstenKundenBedienen () {
        if (warteschlange.empty ()) kunde = null;
        else {
            kunde = (String) warteschlange.front ();
            warteschlange.deq ();
        }
    }
}
```

Simulation

Eintreten eines Kunden
hinten anstellen (bei Schalterschlange: an der kürzesten Schlange)
Heranholen des ersten Kunden
Bedienen des aktuellen Kunden
Schlangenstand anzeigen

Schalterschlange

Parameter fuer die Warteschlangen-Simulation:
Anzahl Service-Mitarbeiter: 3
Wahrscheinlichkeit fuer Kundenauftreten pro Zeitschritt: 0.9

Zeitpunkt 1:
Mitarbeiter 1: K_1
Mitarbeiter 2: (frei)
Mitarbeiter 3: (frei)

Zeitpunkt 2:
Mitarbeiter 1: K_1
Mitarbeiter 2: K_2
Mitarbeiter 3: (frei)

Zeitpunkt 3:
Mitarbeiter 1: K_1
Mitarbeiter 2: K_2
Mitarbeiter 3: K_3

Zeitpunkt 4:
Mitarbeiter 1: K_1 K_4
Mitarbeiter 2: K_2
Mitarbeiter 3: K_3

Zeitpunkt 5:
Mitarbeiter 1: K_1 K_4
Mitarbeiter 2: K_2 K_5
Mitarbeiter 3: K_3

Zeitpunkt 6:
Mitarbeiter 1: K_1 K_4
Mitarbeiter 2: K_2 K_5
Mitarbeiter 3: K_3 K_6

Zeitpunkt 7:
Mitarbeiter 1: K_1 K_4
Mitarbeiter 2: K_2 K_5
Mitarbeiter 3: K_6

Zeitpunkt 8:
Mitarbeiter 1: K_1 K_4
Mitarbeiter 2: K_2 K_5
Mitarbeiter 3: K_6 K_7

Zeitpunkt 9:
Mitarbeiter 1: K_4 K_8
Mitarbeiter 2: K_2 K_5
Mitarbeiter 3: K_6 K_7

Verteilerschlange

Parameter fuer die Warteschlangen-Simulation:
Anzahl Service-Mitarbeiter: 3
Wahrscheinlichkeit fuer Kundenauftreten pro Zeitschritt: 0.9

Zeitpunkt 1:
Verteilerschlange:
Mitarbeiter 1: K_1
Mitarbeiter 2: (frei)
Mitarbeiter 3: (frei)

Zeitpunkt 2:
Verteilerschlange:
Mitarbeiter 1: K_1
Mitarbeiter 2: K_2
Mitarbeiter 3: (frei)

Zeitpunkt 3:
Verteilerschlange:
Mitarbeiter 1: K_1
Mitarbeiter 2: K_2
Mitarbeiter 3: K_3

Zeitpunkt 4:
Verteilerschlange: K_4
Mitarbeiter 1: K_1
Mitarbeiter 2: K_2
Mitarbeiter 3: K_3

Zeitpunkt 5:
Verteilerschlange: K_4 K_5
Mitarbeiter 1: K_1
Mitarbeiter 2: K_2
Mitarbeiter 3: K_3

Zeitpunkt 6:
Verteilerschlange: K_4 K_5 K_6
Mitarbeiter 1: K_1
Mitarbeiter 2: K_2
Mitarbeiter 3: K_3

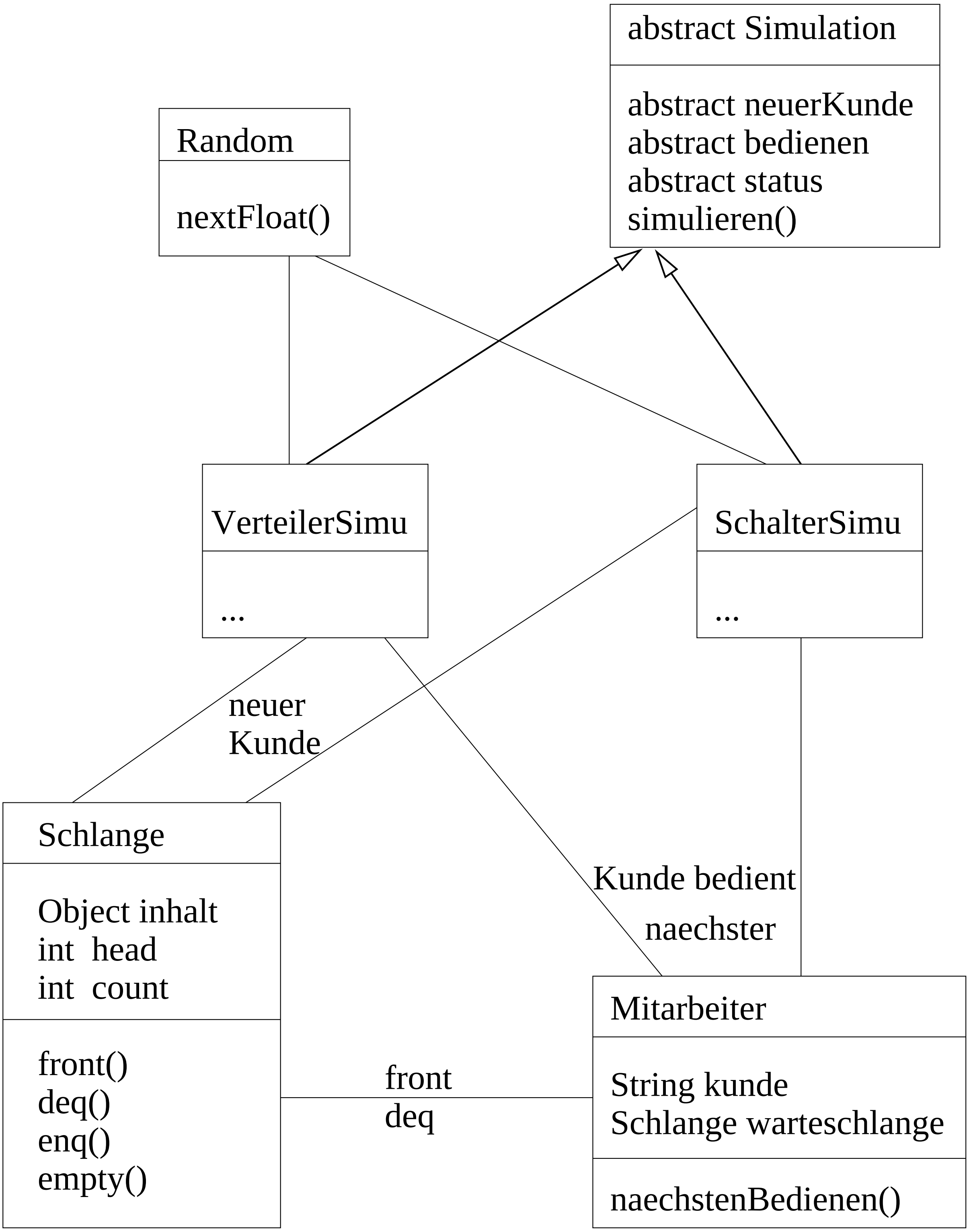
Zeitpunkt 7:
Verteilerschlange: K_5 K_6
Mitarbeiter 1: K_1
Mitarbeiter 2: K_2
Mitarbeiter 3: K_4

Zeitpunkt 8:
Verteilerschlange: K_5 K_6 K_7
Mitarbeiter 1: K_1
Mitarbeiter 2: K_2
Mitarbeiter 3: K_4

Zeitpunkt 9:
Verteilerschlange: K_6 K_7 K_8
Mitarbeiter 1: K_5
Mitarbeiter 2: K_2
Mitarbeiter 3: K_4

Zeitpunkt 10:
Verteilerschlange: K_6 K_7 K_8 K_9
Mitarbeiter 1: K_5
Mitarbeiter 2: K_2
Mitarbeiter 3: K_4

Skizze der Modellierung



Realisierung in JAVA

```
import java.util.Random;            // Zufallszahlengenerator
import AlgoTools.IO;
import Mitarbeiter;        // Klasse fuer simulierte Mitarbeiter
abstract class Simulation {
    int anzahlMitarbeiter; // Anzahl Mitarbeiter (= Schalter)
    Mitarbeiter[] mitarbeiter;    // Feld mit allen Mitarbeitern
    float kundenWahrscheinlichkeit;
    int kundennummer;             // laufende Kundennummer
    Random random;                     // Zufallszahlengenerator

    public Simulation () {
        IO.println ("Parameter:");
        anzahlMitarbeiter = IO.readInt ("Anzahl Mitarbeiter:
");
        kundenWahrscheinlichkeit = IO.readFloat
("Wahrscheinlichkeit fuer Kundenauftreten pro Zeitschritt: ");
        IO.println ();
        mitarbeiter = new Mitarbeiter[anzahlMitarbeiter];
        kundennummer = 0;
        random = new Random (1998);
    }
```

```
abstract void zufaelligNeuenKundenErzeugen ();

    abstract void
zufaelligBedienungAbschliessenUndWartendeBedienen ();
    abstract void statusAnzeigen (int zeitpunkt);

    public void simulieren () {
        int m; // Index des aktuellen Mitarbeiters
        int zeitpunkt; // aktueller Zeitpunkt
        for (zeitpunkt = 1; zeitpunkt <= 10; zeitpunkt++)
{
            zufaelligNeuenKundenErzeugen ();

zufaelligBedienungAbschliessenUndWartendeBedienen ();
            statusAnzeigen (zeitpunkt);
        }
    }
}
```

Schaltersimulation

```
import java.util.Random;                    // Zufallszahlengenerator
import AlgoTools.IO; // Vornbergers AlgoTools−I/O−Klasse
import LS8Tools.Schlange;                            // ADT Schlange
import Mitarbeiter;         // Klasse fuer simulierte Mitarbeiter

class SchalterSimu extends Simulation {
    public SchalterSimu () {
        super ();          // Aufruf des Oberklassenkonstruktors
        int m;                  // Index des aktuellen Mitarbeiters
        for (m = 0; m < anzahlMitarbeiter; m++)
        mitarbeiter[m] = new Mitarbeiter ();
    }
```

```
private Schlange kuerzesteSchlange () {
        int m;              // Index des aktuellen Mitarbeiters
        int bedienterKunde;          // 0, falls Mitarbeiter frei,
                   // 1, falls er einen Kunden bedient, der zur
                 // Laenge seiner Warteschlange hinzugezaehlt
                                              // werden muss
        Schlange s;
        Schlange kuerzesteSchlange;
        int minSchlangenlaenge;

        minSchlangenlaenge =
(mitarbeiter[0].gibWarteschlange()).length() + 2;
        kuerzesteSchlange = null;
        for (m = 0; m < anzahlMitarbeiter; m++) {
            s = mitarbeiter[m].gibWarteschlange();
            if (mitarbeiter[m].frei()) bedienterKunde = 0;
            else bedienterKunde = 1;
            if (s.length () + bedienterKunde <
minSchlangenlaenge) {
                minSchlangenlaenge = s.length () +
bedienterKunde;
                kuerzesteSchlange = s;
            }
        }
        return (kuerzesteSchlange);
    }
```

```
void zufaelligNeuenKundenErzeugen () {
        double r;    // Zufallszahl fuer (Nicht)Auftreten eines
neuen Kunden
        Schlange warteschlange;   // kuerzeste Warteschlange
        r = random.nextFloat (); // Ziehe Zufallszahl aus 0..1
        if (r <= kundenWahrscheinlichkeit)  {  // Neuer
Kunde?
            kundennummer++;                       // Falls ja:
            warteschlange = kuerzesteSchlange ();
            warteschlange.enq ("K_"+ kundennummer);
            // an Warteschlange anhaengen
        }
    }
```

```
void
zufaelligBedienungAbschliessenUndWartendeBedienen ()
{
        int m;                 // Index des aktuellen Mitarbeiters
        double r;              // Zufallszahl fuer Auftreten Kunde
        for (m = 0; m < anzahlMitarbeiter; m++)  {
            // Fuer jeden Mitarbeiter:
            r = random.nextFloat ();         // Zufallszahl 0..1
            if (r <= 0.15f) mitarbeiter[m].setzeFrei();       //
Kunde bedient?
            if
((!((mitarbeiter[m].gibWarteschlange()).empty()))
            && (mitarbeiter[m].frei()))
            mitarbeiter[m].naechstenKundenBedienen();     //
Wartenden bedienen
        }
    }
```

```
void statusAnzeigen (int zeitpunkt) {
        int m;              // Index des aktuellen Mitarbeiters

        IO.println ("Zeitpunkt "+ zeitpunkt + ":");
        for (m = 0; m < anzahlMitarbeiter; m++) {
            IO.print ("Mitarbeiter "+ (m+1) + ": "+
mitarbeiter[m]);
            IO.println (""+
mitarbeiter[m].gibWarteschlange());
        }
        IO.println ();
    }
}
```

Verteilersimulation

```
import java.util.Random;              // Zufallszahlengenerator
import AlgoTools.IO; // Vornbergers AlgoTools–I/O–Klasse
import LS8Tools.Schlange;                        // ADT Schlange
import Simulation;
import Mitarbeiter;        // Klasse fuer simulierte Mitarbeiter
class VerteilerSimu extends Simulation {
    private Schlange warteschlange;
    public VerteilerSimu () {
        super ();        // Aufruf des Oberklassenkonstruktors
        int m;                // Index des aktuellen Mitarbeiters
        warteschlange = new Schlange (100);
        for (m = 0; m < anzahlMitarbeiter; m++)
        mitarbeiter[m] = new Mitarbeiter (warteschlange);
    }
    void zufaelligNeuenKundenErzeugen () {
        double r;                  // Zufallszahl Auftreten Kunde

        r = random.nextFloat ();             // Zufallszahl 0..1
        if (r <= kundenWahrscheinlichkeit) {     //Kunde?
            kundennummer++;                         // Falls ja:
            warteschlange.enq ("K_"+ kundennummer);
            // an Warteschlange anhaengen
        }
    }
```

```
void
zufaelligBedienungAbschliessenUndWartendeBedienen()
{
        int m;                // Index des aktuellen Mitarbeiters
        double r;             // Zufallszahl fuer Auftreten Kunde

        for (m = 0; m < anzahlMitarbeiter; m++)  {
            r = random.nextFloat ();
            if (r <= 0.15f) mitarbeiter[m].setzeFrei();
            if ((!(warteschlange.empty())) &&
                                      (mitarbeiter[m].frei()))
            mitarbeiter[m].naechstenKundenBedienen();
        }
    }
    void statusAnzeigen (int zeitpunkt)  {
        int m;                // Index des aktuellen Mitarbeiters

        IO.println ("Zeitpunkt "+ zeitpunkt + ":");
        IO.println ("Verteilerschlange: "+ warteschlange);
        for (m = 0; m < anzahlMitarbeiter; m++)
        IO.println ("Mitarbeiter "+ (m+1) + ": "+
mitarbeiter[m]);
        IO.println ();
    }
 }
```