Programmieren 2
Übung Semesterwoche 1

Aufgabe 1a

```
public class Konto
{
    //Klassenattribut
    private static int count;

    //Attribute
    private int no;
    private double saldo;
    private double rate;

    public Konto()
    {
       createKonto();
       this.saldo = 0.0;
       this.rate = 0.0;
    }

    public Konto(double Saldo, double Rate)
    {
       createKonto();
       this.saldo = Saldo;
       this.rate = Rate;
    }

    //Methoden
    private void createKonto()
    {
        this.no = count++;
    }

    public void payIn(double value)
    {
        saldo = saldo + value;
    }

    public void payOut(double value)
    {
        saldo = saldo - value;
    }

    public int getNo()
    {
        return no;
    }

    public double getSaldo()
    {
        return saldo;
    }

    public void setSaldo(double Saldo)
    {
        this.saldo = Saldo;
    }

    public double getRate()
    {
        return rate;
```

```
    }

    public void setRate(double Rate)
    {
        this.rate = Rate;
    }

    public void print()
    {
        System.out.println("Nr: " + no);
        System.out.println("Saldo: " + saldo);
    }

}
```

Aufgabe 1b

```
public class Spar extends Konto
{
    //Attribute
    private double maxOut;

    public Spar(double maxOut)
    {
        super();
        this.maxOut = maxOut;
    }

    public Spar(double Saldo, double Rate, double maxOut)
    {
        super(Saldo, Rate);
        this.maxOut = maxOut;
    }

    public static void main(String[] args)
    {
        Spar s = new Spar(1000.0, 2.0, 500.0);
    }

    //Methoden
    public void payOut(double value)
    {
        if(value <= maxOut)
        {
            super.payOut(value);
        }
    }

    public double getSaldo()
    {
        return super.getSaldo();
    }

    public void setSaldo(double Saldo)
    {
        super.setSaldo(Saldo);
    }

    public void print()
    {
        super.print();
        System.out.println("maxOut: " + this.maxOut);
    }

}
```

Aufgabe 1c

```
import java.util.*;

public class Bank
{
    String name;
    ArrayList<Konto> accountList;

    public Bank(String Name)
    {
      this.name = Name;
      accountList = new ArrayList<Konto>();
    }

    public void openKonto()
    {
        accountList.add(new Konto());
    }

    public void openKonto(double Saldo, double Rate)
    {
        accountList.add(new Konto(Saldo, Rate));
    }

    public void openSpar(double maxOut)
    {
        accountList.add(new Spar(maxOut));
    }

    public int noOfAccounts()
    {
        int counter = 0;
        Iterator<Konto> it = accountList.iterator();
        while(it.hasNext())
        {
            counter++;
        }
        return counter;
    }

    public void printAccounts()
    {
        Iterator<Konto> it = accountList.iterator();
        while(it.hasNext())
        {
            it.next().print();
        }
    }

    public void printFund()
    {
        double sum = 0.0;
        Iterator<Konto> it = accountList.iterator();
        while(it.hasNext())
        {
            sum += it.next().getSaldo();
        }
        System.out.println("Fund: "+sum);
    }
}
```

Aufgabe 2a

```
public class ListNode
{
    private Konto konto;
    private ListNode next;

    public ListNode(ListNode n, Konto k)
    {
        next = n;
        konto = k;
    }

    public void setNext(ListNode n)
    {
        next = n;
    }

    public ListNode getNext()
    {
        return next;
    }

    public void setKonto(Konto k)
    {
        konto = k;
    }

    public Konto getKonto()
    {
        return konto;
    }

}
```

Aufgabe 2b

```
public class LinkedList
{
    private ListNode head;

    public LinkedList()
    {
        head = null;
    }

    public boolean isFound(Konto k)
    {
        ListNode actualNode = head;
        while((actualNode != null) && (k.getNo() != actualNode.getKonto().getNo()))
        {
            actualNode = actualNode.getNext();
        }
        if (actualNode == null)
        {
            return false;
        }
        else
        {
            return true;
        }
    }

    public boolean insert(Konto k)
    {
        if(!isFound(k))
        {
            head = new ListNode(head, k);
            return true;
        }
        else
        {
            return false;
        }
    }

    public int length()
    {
        int counter = 0;
        ListNode actualNode = head;
        while(actualNode != null)
        {
            actualNode = actualNode.getNext();
            counter++;
        }
        return counter;
    }
}
```