Lösung Aufgabe 4

1.1

```
import java.io.File;

public class Beobachter extends Thread {
	private File file;
	private long zeit;
	public Beobachter(String filename) {
		super();
		file = new File(filename);
		zeit=file.lastModified();
	}
	@Override
	public void run() {
		//super.run();
		while (!this.isInterrupted()){
			long neuzeit=file.lastModified();
			if (neuzeit>zeit){
				System.out.println("Datei "+file.getName()+" wurde 
geändert");
				zeit=neuzeit;
			}
			try {
				Thread.sleep(10000);
			} catch (InterruptedException e) {
				this.interrupt();
			}
		}
	}
	public static void main(String[] args) {
		Beobachter beo=new Beobachter("file.txt");
		beo.start();
		String eingabe;
		Scanner cs=new Scanner(System.in);
		do {
			eingabe=cs.nextLine();
		}while(!eingabe.equals("stop"));
		cs.close();
		beo.interrupt();
	}
}
```

1.2

//bank

```
//import java.util.concurrent.locks.Condition; //ReentrantLock
//import java.util.concurrent.locks.ReentrantLock; //ReentrantLock

public class Bank {
//	static ReentrantLock lock=new ReentrantLock();//ReentrantLock
//	static Condition condition=lock.newCondition();//ReentrantLock

	private double kontostand;

	public Bank(double kontostand) {
		super();
```

```java
			this.kontostand = kontostand;
		}

		public double getKontostand() {
			return kontostand;
		}

		public synchronized void ab(double betrag) {
				//lock.lock();//ReentrantLock
					if (kontostand-betrag>=0) {
						kontostand = kontostand-betrag;
					}
					else {
						try {
							wait();
							//condition.await();//ReentrantLock
						} catch (InterruptedException e) {
							Thread.currentThread().interrupt();
						}
					}
				//lock.unlock();//ReentrantLock
		}
		public synchronized void zu(double betrag) {
			//lock.lock();//ReentrantLock
				kontostand = kontostand+betrag;
				notifyAll();
				//condition.signalAll();//ReentrantLock
			//lock.unlock();//ReentrantLock
			}

}

// Grandchild
public class Grandchild extends Thread {
		Bank bank;
		String name;
		double kontostand;
		int wartezeit;
		public Grandchild(Bank bank, String name, double kontostand, int wartezeit)
{
			super();
			this.bank = bank;
			this.name = name;
			this.kontostand = kontostand;
			this.wartezeit = wartezeit;
		}
		@Override
			public void run() {
			//super.run();
			double betrag=Math.random()*10;
//			while (bank.getKontostand()>betrag){
			while (!this.isInterrupted()){
//				synchronized(bank){
					bank.ab(betrag);
					kontostand+=betrag;
					System.out.println(name+": ab "+betrag+" rest
"+bank.getKontostand());
					//}
//				}
				try {
					Thread.sleep(wartezeit);
```

```java
                Thread.yield();
            } catch (InterruptedException e) {
                //e.printStackTrace();
                this.interrupt();
            }
            betrag=Math.random()*10;
        }
    }
}

// Grandfather
import java.util.Scanner;

public class Grandfather {

    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);

        Bank bank=new Bank(500);
        Grandchild e1,e2,e3;
        e1=new Grandchild(bank,"Udo",0,500);
        e2=new Grandchild(bank,"Ulla",0,100);
        e3=new Grandchild(bank,"Uwe",0,250);
        e1.start();e2.start();e3.start();
        double betrag;
        betrag=s.nextDouble();
        while (betrag>0){
//              synchronized(bank){
                bank.zu(betrag);
//              }
                betrag=s.nextDouble();
        }
        if (bank.getKontostand()<5){
            //System.exit(0);
            e1.interrupt();e2.interrupt();e3.interrupt();
        }
    }

}
```

3.

```
import java.io.File;
import java.util.Scanner;
import java.util.concurrent.Callable;

public class FileEvaluator implements Callable<Long> {

	String fileName;

	public FileEvaluator(String fileName) {
		super();
		this.fileName = fileName;
	}

	@Override
	public Long call() throws Exception {
		Scanner sc=new Scanner(new File(fileName));
		String line;
		long anz=0;
		while(sc.hasNext())
		{
			line=sc.nextLine();
			if (line.split(" ")[0].equals("not")) anz++;
		}
		sc.close();
		return anz;
	}
}

import java.util.ArrayList;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.Future;

public class MainClass {

	public static void main(String[] args) {
		String[] files={"Server1.txt","Server2.txt","server3.txt"};
		ArrayList<Future<Long>> list=new ArrayList<Future<Long>>();

	    ExecutorService exec = Executors.newFixedThreadPool(10);

	    for (String name:files )list.add(exec.submit(new FileEvaluator(name)));
		long sum=0;
	    try {
			for (Future<Long> erg:list){
				System.out.println(erg.get());
				sum+=erg.get();
			}
		} catch (InterruptedException e) {
			  e.printStackTrace();
		} catch (ExecutionException e) {
			  e.printStackTrace();
		}
	    exec.shutdown();
	    System.out.println(sum);
	}
}
```

4.

```
public class Account {
	private int number;
	private int balance;
	public Account(int number, int balance) {
		super();
		this.number = number;
		this.balance = balance;
	}
	public int getNumber() {
		return number;
	}
	public void setNumber(int number) {
		this.number = number;
	}
	public int getBalance() {
		return balance;
	}
	public void setBalance(int balance) {
		this.balance = balance;
	}
}

public class Transaction {
		private int from;
		private int to;
		private int amount;
		public Transaction(int from, int to, int amount) {
			super();
			this.from = from;
			this.to = to;
			this.amount = amount;
		}
		public int getFrom() {
			return from;
		}
		public int getTo() {
			return to;
		}
		public int getAmount() {
			return amount;
		}
}

import java.util.HashMap;
import java.util.concurrent.ArrayBlockingQueue;

public class Bank implements Runnable {
	public ArrayBlockingQueue<Transaction> transactionQueue;
	public HashMap<Integer,Account> accounts;

	public Bank(ArrayBlockingQueue<Transaction> transactionQueue) {
		super();
		this.transactionQueue = transactionQueue;
		accounts=new HashMap<Integer,Account>();

	}
```

```java
	@Override
	public void run() {
		while (!Thread.currentThread().isInterrupted()){
			try {
				Transaction t=transactionQueue.take();
				int betrag=t.getAmount();
				int from=t.getFrom();
				Account afrom=accounts.get(from);
				int to=t.getTo();
				Account ato=accounts.get(to);
				if (afrom!=null &&ato!=null){
					afrom.setBalance(afrom.getBalance()-betrag);
					ato.setBalance(ato.getBalance()+betrag);
				}
			} catch (InterruptedException e) {
				Thread.currentThread().interrupt();
			}
		}
	}
}

import java.io.InputStreamReader;
import java.util.Scanner;
import java.util.concurrent.ArrayBlockingQueue;

public class TransactionHandler {

	public static void main(String[] args) {
		ArrayBlockingQueue<Transaction> transactionQueue=new
ArrayBlockingQueue<Transaction>(100);
		Bank bank=new Bank(transactionQueue);
		bank.accounts.put(1,new Account(1,1000));
		bank.accounts.put(2,new Account(2,1000));
		bank.accounts.put(3,new Account(3,1000));
		bank.accounts.put(4,new Account(4,1000));
		Thread t=new Thread(bank);
		transactionQueue.add(new Transaction(1,2,100));
		transactionQueue.add(new Transaction(1,3,200));
		transactionQueue.add(new Transaction(1,4,100));
		transactionQueue.add(new Transaction(1,5,100));
		t.start();

		for (Account a: bank.accounts.values()){
			System.out.println(a.getNumber()+ " "+a.getBalance());
		}
		Scanner eing=new Scanner(new InputStreamReader(System.in));
		int betrag=eing.nextInt();
		while (betrag!=0){
			int from=eing.nextInt();
			int to=eing.nextInt();
			transactionQueue.add(new Transaction(from,to,betrag));
			betrag=eing.nextInt();
		}
		eing.close();
		for (Account a: bank.accounts.values()){
			System.out.println(a.getNumber()+ " "+a.getBalance());
		}
		t.interrupt();
	}
}
```