Aufgabe 1 (10+1+2+2+3+2+2=22 Punkte):
In dieser Aufgabe sollen Sie ein Zahlenschloss nachahmen:
Ein Objekt der Klasse Zahlenschloss soll von mehreren Threads gleichzeitig benutzt werden können. Die Klasse Zahlenschloss besitzt einen Konstruktor mit einem Parameter des Typs int[], über den die Zahlenkombination eingegeben werden kann, bei der sich das Schloss öffnet. Durch die Länge des Felds wird damit implizit festgelegt, wie viele Rädchen zum Einstellen von Zahlen dieses Zahlenschlossobjekt besitzt. Die Zahlenkombination, bei der sich das Schloss öffnet, kann nach dem Erzeugen des Objektes nicht mehr geändert werden. Die Zahnrädchen stehen zu Beginn auf einem Wert, der Ihnen überlassen ist (z.B. alle auf 0). Sie müssen sicherstellen, dass ein Zahnrädchen nur auf Werte zwischen 0 und 9 eingestell werden kann. Sie können statdessen der Einfachheit halber davon ausgehen, das die Rädchen auf jede beliebige int-Zahl gedreht werden können.
Ihr Zahlenschloss soll folgende Methoden besitzen:

- public void drehen(int radnummer, int zahl): Mit dieser Methode wird das Rädchen mit der Nummer radnummer auf zahl gedreht. Sie müssen nicht kontrollieren, ob radnummer ein gültiger Wert ist, denn andernfalls wird vermutlich ohnehin eine Ausnahme in Ihrer Methode geworfen.
- public void warten(); Wenn diese Methode aufgerufen wird, wartet der Aufrufer so lange, bis die richtige Zahlenkombination an den Rädchen eingestellt wird und sich das Schloss somit öffnen lässt. Ist diese Kombination zum Zeitpunkt des Aufrufs von warten schon eingestellt, kehrt man aus warten sofort zurück.
- public int lesen(int radnummer): Liefert die momentan am Rädchen mit der Nummer radnummer eingestellte Zahl zurück.
- public int anzahlRädchen(): Liefert die Anzahl der einstellbaren Rädchen zurück.

a) Geben sie eine vollständige Implementierung der Klasse Zahlenschloss an!
(10/7)

b) Wie viele Wartebedingungen benötigt man für die Klasse Zahlenschloss?
(1/1)

c) Wie lautet die andere Bedingung zur Entscheidung, ob notifyAll benötigt wird, und ist diese Bedingung für eine korrekte Implementierung der Klasse Zahlenschloss erfüllt oder nicht? Muss man folglich notifyAll benutzen oder genügt notify?
(2/1)

d) Falls Sie das in ihrer Implementierung in Teilaufgabe a noch nicht schon so programmiert haben, dann geben Sie hier eine optimierte Implementierung eines Teils Ihrer Klasse Zahlenschloss so an, das möglichst selten Threads durch notify oder notifyAll geweckt werden! Falls Sie das aber in Teilaufgabe schon so realisiert haben, dann schreiben Sie bitte einfach “s. Aufgabe a”!
(2/0)

e) Betrachten Sie den folgenden Programmcode eines Threads! zk sei eine Referenz auf ein Zahlenschlossobjekt, das auch von anderen Threads benutzt werden kann:

```
for(int i = 0; i < zk.anzahlRädchen(); i++)
{
      System.out.println(“” + zk.lesen(i));
}
System.out.println();
```

Warum kann eine Zahlenkombination ausgegeben werden die es tatsächlich nie gegeben hat? Geben Sie zur Beantwortung der Frage ein ganz konkretes Beispiel an und keine allgemeinen Erklärungen!
(3/1)

f) Wie können Sie den Programmcode aus Teilaufgabe e ändern, damit nur Zahlenkombinationen ausgegeben werden, die es tatsächlich gegeben hat? Bitte beachten sie das sie außer den vorgegebenen öffentlichen Methoden der Klasse Zahlenschloss keine weiteren öffentlichen Methoden hinzufügen dürfen.
(2/0)

g) Angenommen, es ist nicht die richtige Zahlenkombination eingestellt. Anschließend rufen mehrere Threads die Methode warten auf, um auf das Öffnen des Schlosses zu warten. Nun werde von einem oder mehreren anderne Threads an den Rädchen gedreht, so dass die richtige Zahlenkombination für das Öffnen eingestellt wird. Kehjren alle Threads aus ihrer Methode warten zurück, ganz unabhängig davon was im Folgenden passiert?
(2/0)

Aufgabe 2 (6+10=16 Punkte): Grafische Benutzeroberflächen
Ein Benaknnter erzählt Ihnen, dass er die Absicht hat eine Anwendung zu programmieren, in der gewisse Werte durch Slider angezeigt werden. Wenn sich durch Benutzerinteraktion diese Werte ändern, soll der Slider nicht direkt zum neuene Wert springen, sondern der Slider soll schrittweise (d.h. animiert) zum neuen Wert bewegt werden. Sie fragen Ihren Bekannten, ob er diese Idee schon realisiert habe. Ihr Bekannter antwortet, dass er das zwar schon programmiert, aber noch nicht ausprobiert habe, weil das ja so einfach sei. Daraufhin lassen Sie sich die Klasse mit der actionPerformed-Methode zeigen in der der Slider auf den neuen Wert bewegt wird. Sie sieht so aus:

```
class BadAnimation implements ActionListener
{
	private JSlider slider;

	public BadAnimation(JSlider slider)
	{
		this.slider = slider;
	}

	public void actionPerformed(ActionEvent evt)
	{
		int oldValue = slider.getValue();
		//neuer Wert wird beschafft
		//(wie ist hier nicht von Interesse)
		int newValue = …;
		int delta;
		if(oldValue < newValue)
		{
			delta = 1;
		}
		else
		{
			delta = -1;
		}
		while(oldValue != newValue)
		{
			oldValue += delta;
			slider.setValue(oldValue);
			try
			{
				Thread.sleep(50);
			}
			catch(InterruptecException e) {}
		}
	}
}
```

a) Sie betrachten die Klasse und erzählen Ihrem Bekannten, dass das nicht so funktioniert, wie er es sich vorgestellt hat. Daraufhin fragt er: “Was passiert denn, wenn ich das ausführe?” Wie lautet die richtige Antwort auf diese Frage?
(6/0)

b) Ihr Bekannter ist nun ziemlich enttäuscht, dass seine schönen animierten Slider nicht funktionieren. Helfen Sie Ihrem Bekannten! Geben Sie einen Programmtext an, der so funktioniert wie erwünscht! Auf derfolgenden Seite ist noch weiterer Platz für eventuell zusätzlichen benötigte Klassen.
(10/0)

```
class BetterAnimation extends Thread {
  private JSlider slider;
  private int newValue;
  private int oldValue;

  public BetterAnimation(JSlider slider, int newValue) {
    this.slider = slider;
    this.newValue = newValue;
    oldValue = slider.getValue();
  }

  public void run() {
    int delta;
    if(oldValue < newValue) {
      delta = 1;
    }
    else {
      delta = -1;
    }
    while(oldValue != newValue) {
      oldValue += delta;
      slider.setValue(oldValue);
      try {
        Thread.sleep(50);
      }
      catch(InterruptedException e) {}
    }
  }
}

class BadAnimation implements ActionListener {
  private JSlider slider;
  private JTextField tf;

  public BadAnimation(JSlider slider, JTextField tf) {
```

```
    this.slider = slider;
    this.tf = tf;
  }

  public void actionPerformed(ActionEvent evt) {
    int oldValue = slider.getValue();
    //neuer Wert wird beschafft
    //(wie ist hier nicht von Interesse)
    int newValue = Integer.parseInt(tf.getText());
    BetterAnimation animation = new BetterAnimation(slider, newValue);
    animation.start();
  }
}
```

Aufgabe 3 (6+3+4+3=16 Punkte):
Im Lehrbuch wurde im Zusammenhang mit der Socket-Schnittstelle für TCP ein Beispiel entwickelt, in dem der Server für die Bearbeitung eines Auftrags länger braucht. Wenn man ihm ihm über eine aufgebaute TCP-Verbindung eine ganze positive Zahl in Form eines Strings sendet, dann verzögert er seine Antwort um die angegeben Zahl in Sekunden und liefert danach als Echo das was man ihm gesendet hat. Diese Anwendung wurde als Beispiel herangezogen für eine Parallelisierung eines TCP-Servers.

a) Erklären Sie, was statische und dynamische Parallelität jeweils bedeuten und erklären Sie wie man einen TCP-Server mit statischer und dynamischer Parallelität mit Hilde der, Socket-Schnittstelle jeweils implementieren kann! Es muss kein Programmcode angegeben werden aber die wesentlichen Apekte müssen genannt werden.
(6/3)

b) Der im Buch angegebene Client führt nach dem Verbindungsaufbau in einer Schleide aus: Kommando senden - Antwort empfangen. Eine Bekannte erzählt Ihnen nun, dass sie die Idee hätte, die Laufzeit des Clients zu reduzieren, indem sie einen Client entwickelt, der in einer ersten Schleife mehrere Kommandos direkt hintereinander sendet und dann in einer zweiten Schleife die Antworten empfängt. Sie antworten ihr, dass ihre Idee zu keiner Laufzeitreduktion führen würde. Daraufhin widerspricht Ihre Bekannte Ihnen und erinnert daran, dass der Server mit statischer oder dynamischer Parallelität ausgestattet sei. Was antworten sie ihr? Stimmen sie ihr zumindest für eine der beiden Formen von Parallelität zu?
(3/1)

c) Erläutern Sie wie die Datenstromorientierung von TCP gerade beim Cleint Ihrer Bekannten ohne spezielle Maßnahemn ein Problem für den Server werden könnte! Wie kann das Problem gelöst werden? Geben Sie zwei unterschiedliche Lösungsmöglichkeiten an! Es ist kein Programmcode verlangt.
(4/2)

d) Was müssten sie ändern, damit es bei der Nutzung des Clients Ihrer Bekannten (senden - senden - … - empfangen - empfangen - empfangen - …= auf jeden Fall zu einer Laufzeitreuktion gegenüber dem Standard-Cleint aus dem Buch (senden - empfangen - senden - empganegn - …) kommen würde? (3/0)
Aufgabe 4 (4+2=6 Punkte)
Folgende RMI-Schnittstelle und folgende RMI-Klasse seien gegeben:

```
import java.rmi.*;

public interface Hello extends Remote
{
      public String hello(String name) throws RemoteException;
}

---------------------------------------------------------------

import java.rmi.*;
import java.rmi.server.*;

public class HelloImpl extends UnicastRemoteObject impements Hello
{
      public String hello(String name) throws RemoteException
      {
            return “hello” + name;
      }
}
```

a) Was ist am oben gegebenen Programmcode falsch? (4/4)

b) Welche Art von Paramezerübergabe liegt beim Aufruf der RMI-Methode hello vor: Wertübergabe oder Referenzübergab? Begründen sie Ihre Antwort! (2/2)