

Modulendprüfung**27. Sept. 2007****Wichtig:**

- Als Unterlagen dürfen zwei A4 Seiten (ein Blatt) mit eigenen Notizen (müssen nicht handschriftlich sein) benutzt werden. Weitere Unterlagen sind nicht erlaubt!
- Die Lösungen sind an den entsprechenden Stellen in den Aufgabenblättern einzutragen.
- Halten Sie Ihre Antworten kurz. Es gibt für lange Sätze nicht mehr Punkte.
- Alle Aufgabenblätter sind abzugeben. Es werden keine Zusatzblätter akzeptiert!
- Zeit : 120 Minuten

Viel Glück bei der Arbeit !

Aufgabe 1 (12 Punkte)A. Was bedeutet das Sprachkonstrukt: `atomic S1; ...; Sn; end;` ? (3 Punkte)

B. Wann besteht bei Nebenläufigkeit Bedarf an Synchronisation ? (2 Punkte)

C. Was verstehen Sie unter "Sicherheit" bei nebenläufigen Programmen ? (2 Punkte)

D. Was verstehen Sie unter "Lebendigkeit" bei nebenläufigen Programmen ? (2 Punkte)

E. Zählen Sie drei wichtige Punkte vom Konzept "Semaphor" auf. (3 Punkte)

Aufgabe 2 (11 Punkte)

Gegeben ist der folgende C# Code "Ping-Pong". Es ist eine kleine Thread Demonstration, die zwei Ping-Pong Spieler "simuliert".

```
1 class PingPong
2 {
3     private String whoseTurn = null;
4
5     public bool hit(String opponent) {
6         lock (this) {
7             String x = Thread.CurrentThread.Name;
8             if (whoseTurn == null) {
9                 whoseTurn = x;
10                return true;
11            }
12            if (whoseTurn == "DONE") {
13                return false;
14            }
15            if (opponent == "DONE") {
16                whoseTurn = opponent;
17                Monitor.PulseAll(this);
18                return false;
19            }
20            if (x == whoseTurn) {
21                Console.WriteLine("PING! (" + x + ")");
22                whoseTurn = opponent;
23                Monitor.PulseAll(this);
24            }
25            else {
26                if (!Monitor.Wait(this, 2500)) {
27                    Console.WriteLine("***** TIMEOUT! " + x +
28                    " is waiting for " + whoseTurn + " to play.");
29                }
30            }
31            return true;
32        }
33    }
34 }
35
36 class Player
37 {
38     PingPong myTable;
39     String myOpponent;
40
41     public Player(String opponent, PingPong table) {
42         myTable = table;
43         myOpponent = opponent;
44     }
45
46     public void Play() {
47         while (myTable.hit(myOpponent)) ;
48     }
49 }
50
51
```

Aufgabe 2 Fortsetzung

```

52 class Game
53 {
54     public static void Main()
55     {
56         PingPong table = new PingPong();
57         Thread alice = new Thread(new Player("bob", table).Play);
58         Thread bob = new Thread(new Player("alice", table).Play);
59
60         alice.Name = "alice";
61         bob.Name = "bob";
62
63         alice.Start();
64         bob.Start();
65
66         Thread.Sleep(1000);
67         table.hit("DONE");
68         Thread.Sleep(100);
69     }
70 }
71
72

```

Ordnen Sie, falls dies möglich ist, die folgenden Aussagen über den Thread "alice" den dazugehörigen Programmzeile(n) zu. (10 Punkte)

	Zeile(n)
a. Der Thread befindet sich im new-Zustand.	
b. Der Thread wird in den Ready-Zustand versetzt.	
c. Diese Operation(en) des Threads werden (quasi)parallel abgearbeitet.	
d. Der Thread wird <u>in jedem Fall</u> in den Blocked-Zustand versetzt.	
e. Der Thread kommt vom Running-Zustand in den Objects-Wait-Pool.	
f. Der Thread kommt vom Running-Zustand in den Objects-Lock-Pool.	
g. Der Thread kommt vom Objects-Wait-Pool in den Objects-Lock-Pool.	
h. Der Thread wird durch einen andern Thread (nicht Scheduler) unterbrochen.	
i. Der Thread setzt ein Timeout.	
k. Der Thread wird in den Dead-Zustand versetzt.	

Zusatzfrage: Was ist die gemeinsame Ressource im Ping-Pong Code ? (1 Punkt)

Aufgabe 3 (12 Punkte)

Die abgebildete Klasse `BoundedBuffer` besitzt die gleiche Funktionalität, wie die bekannte Klasse `BoundedBufferWithSemaphore` aus den Unterlagen, ist aber effizienter und thread-sicher.

Interface `IQueue`

```
1 interface IQueue {
2     void Enqueue(Object x);
3     Object Dequeue();
4 }
```

Klasse `BoundedBuffer`

```
1 public class BoundedBuffer : IQueue
2 {
3     protected object[] buffer;
4     protected int takePtr = 0;
5     protected int putPtr = 0;
6     protected int usedSlots = 0;
7     protected int emptySlots;
8     protected object putMonitor = new object();
9
10    public BoundedBuffer(int capacity) {
11        if (capacity <= 0)
12            throw new ArgumentException();
13        buffer = new Object[capacity];
14        emptySlots = capacity;
15    }
16
17    protected virtual void IncEmptySlots() {
18        lock (putMonitor) {
19            ++emptySlots;
20            Monitor.Pulse(putMonitor);
21        }
22    }
23
24    protected virtual void IncUsedSlots() {
25        lock (this) {
26            ++usedSlots;
27            Monitor.Pulse(this);
28        }
29    }
30
31    protected void Insert(Object x) {
32        --emptySlots;
33        buffer[putPtr] = x;
34        if (++putPtr >= buffer.Length)
35            putPtr = 0;
36    }
37
38    protected Object Extract() {
39        --usedSlots;
40        Object old = buffer[takePtr];
41        buffer[takePtr] = null;
42        if (++takePtr >= buffer.Length)
43            takePtr = 0;
44        return old;
45    }
46 }
```

Aufgabe 3 Fortsetzung

Fortsetzung Klasse BoundedBuffer

```
47     public virtual void Enqueue(Object x) {
48         if (x == null)
49             throw new ArgumentException();
50         lock (putMonitor) {
51             while (emptySlots <= 0) {
52                 try {
53                     Monitor.Wait(putMonitor);
54                 }
55                 catch (ThreadInterruptedException ex) {
56                     Monitor.Pulse(putMonitor);
57                     throw ex;
58                 }
59             }
60             Insert(x);
61         }
62         IncUsedSlots();
63     }
64
65     public virtual Object Dequeue() {
66         Object old = null;
67         lock (this) {
68             while (usedSlots <= 0) {
69                 try {
70                     Monitor.Wait(this);
71                 }
72                 catch (ThreadInterruptedException ex) {
73                     Monitor.Pulse(this);
74                     throw ex;
75                 }
76             }
77             old = Extract();
78         }
79         IncEmptySlots();
80         return old;
81     }
82 }
```

Die folgenden Fragen beziehen sich auf die Klasse BoundedBuffer.

A. Welches sind die gemeinsamen Ressourcen der Klasse BoundedBuffer ? (3 Punkte)

B. Warum müssen die `protected` Methoden `Insert` und `Extract` nicht mit einem `lock`-Konstrukt synchronisiert sein ? (2 Punkte)

Aufgabe 3 Fortsetzung

- C. Warum sind die öffentlichen Methoden `Enqueue` und `Dequeue` mit dem `lock`-Konstrukt nicht vollständig (von der ersten bis zur letzten Zeile) synchronisiert ? (2 Punkte)

- D. Welche Probleme könnten entstehen, wenn die öffentlichen Methoden `Enqueue` und `Dequeue` vollständig synchronisiert wären ? Geben Sie auch den Grund der Probleme an. (2 Punkte)

- E. Was passiert, wenn ein Thread die Methode `Enqueue` an einem vollen `Bounded Buffer` aufruft und danach `interrupted` wird ? Begründen Sie Ihre Antwort. (3 Punkte)

Aufgabe 4 (13 Punkte)

Die abgebildete Klasse `EchoServer` empfängt Daten eines Clients und sendet diese wieder an den Client zurück, ohne irgendwelche Veränderungen an den Daten vorzunehmen.

```
1 class EchoServer
2 {
3     public static void Main(String[] args) {
4         int port = Convert.ToInt32(args[0]);
5         TcpListener listen = new TcpListener(port);
6         listen.Start();
7         Console.WriteLine("Echo Server auf " +
8             listen.LocalEndpoint.ToString()+" gestartet");
9         BinaryFormatter binFormatter = new BinaryFormatter();
10        while (true) {
11            Console.WriteLine("Warten auf Verbindung...");
12            TcpClient client = listen.AcceptTcpClient();
13            Console.WriteLine("Verbindung mit " +
14                client.Client.RemoteEndPoint);
15            NetworkStream stream = new NetworkStream(client.Client);
16            Object obj = binFormatter.Deserialize(stream);
17            binFormatter.Serialize(stream, obj);
18            client.Close();
19        }
20    }
21 }
```

Aufgabe 4 Fortsetzung

- A. Welche Einschränkungen besitzt die Klasse `EchoServer` ? Geben Sie nur zwei Einschränkungen an. (2 Punkte)

- B. Startet man den `EchoServer` könnte die Ausgabe wie folgt aussehen:

```
Unhandled Exception: System.IndexOutOfRangeException: Index was outside the bounds of the array.
   at Echo.EchoServer.Main(String[] args) in
   C:\MPSS07\SocketAufgaben\EchoServer\EchoServer.cs:line 4
```

Was ist hier passiert ? (1 Punkt)

- C. Startet man den `EchoServer` und nimmt ein Client Kontakt auf, könnte die Ausgabe wie folgt aussehen:

```
Echo Server auf 127.0.0.1:7788 gestartet
Warten auf Verbindung...
Verbindung mit 172.21.1.102:1096

Unhandled Exception: System.Runtime.Serialization.SerializationException: The input stream is not a valid binary format. The starting contents (in bytes) are: 77-77-77-77-77-0D-0A-0D-0A-0D-0A-0D-0A-0D-0A-77-77 ...
...
   at System.Runtime.Serialization.Formatters.Binary.BinaryFormatter.Deserialize(Stream serializationStream)
   at Echo.EchoServer.Main(String[] args) in
   C:\MPSS07\SocketAufgaben\EchoServer\EchoServer.cs:line 16
```

Was ist hier passiert ? (1 Punkt)

- D. Die Ausgabe könnte aber auch so aussehen:

```
Unhandled Exception: System.Runtime.Serialization.SerializationException: End of Stream encountered before parsing was completed.
...
   at System.Runtime.Serialization.Formatters.Binary.BinaryFormatter.Deserialize(Stream serializationStream)
   at Echo.EchoServer.Main(String[] args) in
   C:\MPSS07\SocketAufgaben\EchoServer\EchoServer.cs:line 16
```

Was ist hier passiert ? (1 Punkt)

- E. Welche Korrektur muss ausgeführt werden, damit die Fehler von Aufgabe C und D den `EchoServer` nicht mehr beeinflussen ? Sie müssen keinen Code angeben, eine kurze Beschreibung genügt. (2 Punkte)

Aufgabe 4 Fortsetzung

F. Aus dem Unterricht kennen Sie die Klasse `MyClass`, die für die Demos zur Serialisierung verwendet wurde.

```
1  [Serializable]
2  class MyClass
3  {
4      private int value = 0;
5      private String id = null;
6
7      public MyClass(String id, int value) {
8          this.id = id;
9          this.value = value;
10     }
11
12     public String getId() {
13         return id;
14     }
15
16     public int getValue() {
17         return value;
18     }
19 }
```

Erstellen Sie einen einfachen Client, der den `EchoServer` benutzt, um eine tiefe Kopie eines Objekts der Klasse `MyClass` zu machen. (4 Punkte)

```
class EchoClient
{
    public static void Main() {

    }
}
```

G. Damit der `EchoServer` der dem Client auch antworten kann, muss eine wichtige Randbedingung erfüllt sein. Welche ? (2 Punkte)

Aufgabe 5 (6 Punkte)

Ein Server soll eine Nachricht senden. Am einfachsten ist es, wenn der Server schon nach dem Verbindungsaufbau mit einem Browser ein "Hello" senden würde. Zum Beispiel so:



Ein Server, der eine solche Nachricht senden könnte, sieht so aus:

```

1  class MessageServer : AbstractServer
2  {
3      public static String message(String msg) {
4          return
5              "<html>\n" +
6              " <body>\n" +
7              " <h2 align=\"center\">\n" +
8              " <font face=Arial color=#0000FF>" +
9              msg + ".</font>\n" +
10             " </h2>\n" +
11             " </body>\n" +
12             "</html>";
13     }
14
15     override protected AbstractHandler CreateHandler(Socket client) {
16         return new MessageHandler(client);
17     }
18
19     public static void Main() {
20         new MessageServer().Start();
21     }
22 }

1  class MessageHandler : AbstractHandler
2  {
3      private StreamReader sr;
4      private StreamWriter sw;
5      private String id;
6
7      public MessageHandler(Socket client) : base (client) {
8          NetworkStream nws = new NetworkStream(client,true);
9          sr = new StreamReader(nws);
10         sw = new StreamWriter(nws);
11         id = client.RemoteEndPoint.ToString().Split(':')[0];
12     }
13
14     override protected bool ReadRequest() {
15         String request = sr.ReadLine();
16         Console.WriteLine(request);
17         string[] tokens = ((string)request).Split(' ');
18         return (tokens.Length >= 2 && tokens[0] == "get");
19     }
20 }

```

Aufgabe 5 Fortsetzung

```

21     override protected void CreateResponse() {
22         Console.WriteLine(id);
23         String msg = MessageServer.message("Hello "+id);
24         sw.WriteLine("HTTP/1.0 200 OK");
25         sw.WriteLine("Content-Length: " + msg.Length);
26         sw.WriteLine("Content-Type: text/html");
27         sw.WriteLine();
28         sw.WriteLine(msg);
29         sw.Flush();
30     }
31 }

```

Nachdem Sie den MessageServer ohne Laufzeitfehler gestartet haben, möchten Sie diesen testen. Von einem zweiten Rechner starten Sie einen Web-Browser, geben die Adresse ein und dann passiert folgendes:



Warum findet der Browser den MessageServer nicht ? Geben Sie die drei verschiedenen Fehlerquellen an. Geben Sie für die drei verschiedene Fehlerquellen, die Korrektur an. Falls die Korrektur im C#-Code erfolgen müsste, geben Sie an, wo im Code Sie was ändern würden. Es ist aber kein Code zu schreiben. (6 Punkte)

Fehler:

Korrektur:

Fehler:

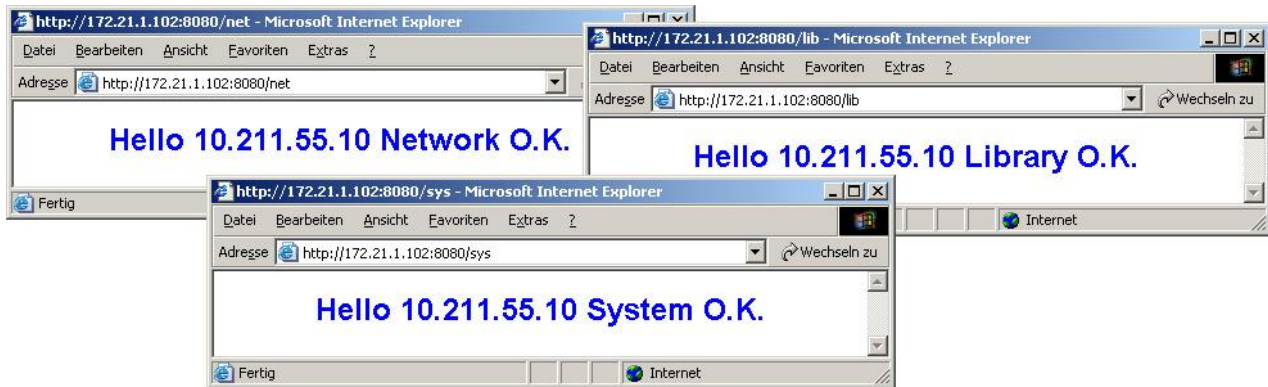
Korrektur:

Fehler:

Korrektur:

Aufgabe 6 (12 Punkte)

Nachdem der `MessageServer` nun richtig funktioniert und vom Browser gefunden wird, soll der Server etwas verfeinert werden. Wir wollen den entfernten Rechner spezifisch nach dem "Wohlbefinden" abfragen und zwar beispielsweise mit diesen Aufrufen:



- A. Wie müsste die oben beschriebene Funktionalität realisiert werden. Geben Sie die drei wichtigsten Aktionen an, welche Sie im `MessageServer` implementieren würden. Es ist kein Code zu schreiben. (3 Punkte)

- B. Da nun der `MessageServer` seine "Geheimnisse" fremden Internet-Benutzern offen legen könnte, wollen wir den Server noch etwas weiter verfeinern. Er soll nämlich nur bestimmten Benutzern antworten. Wie machen Sie das? Welche Methode/Eigenschaft werden Sie unter anderem dazu benutzen? Es ist kein Code zu schreiben. (3 Punkte)

- C. Der `MessageServer` ist nur eine Hilfsapplikation, die auf dem entfernten Rechner läuft. Wie stellen Sie sicher, dass diese Hilfsapplikation so wenig Ressourcen des Rechners wie möglich verbraucht und trotzdem jede Benutzeranfrage jederzeit beantwortet. Es ist kein Code zu schreiben. (3 Punkte)

Aufgabe 6 Fortsetzung

- D. Zum Schluss noch dies. Plötzlich gibt unser MessageServer den folgenden Laufzeitfehler aus.

```
System.NullReferenceException: Object reference not set to an instance of an object.  
at SocketAufgaben.MessageHandler.ReadRequest() in  
C:\MPSS07\SocketAufgaben\MessageServer\MessageHandler.cs:line 17  
at ServerPattern.AbstractHandler.Run() in  
C:\MPSS07\SocketAufgaben\ServerPattern\AbstractHandler.cs:line 13
```

Der Server läuft zwar weiter, aber trotzdem: Was ist passiert ? (3 Punkte)

Anhang C# versus Java

Ausnahmebehandlung (Exceptions)	
C#	Java
<pre>try { ... throw Exception; ... } catch (Exception e) { ... } finally { ... }</pre>	<pre>try { ... throw Exception; ... } catch (Exception e) { ... } finally { ... }</pre>
Exception - SystemException - IOException - FileNotFoundException - ... - ... - ApplicationException	Throwable - Exception - IOException - FileNotFoundException - ... - ... - RuntimeException - Error

Threads	
C#	Java
<pre>class MyClass { void P() { ... thread actions ... } } ... MyClass my = new MyClass(); Thread t = new Thread(new ThreadStart(my.P)); t.Start();</pre>	<pre>class MyThread extends Thread { public void run() { ... thread actions ... } } ... Thread t = new MyThread(); t.start();</pre>
<ul style="list-style-type: none"> • Beliebige Methode kann als Thread gestartet werden • Keine Unterklasse von Thread nötig (nicht möglich) 	<ul style="list-style-type: none"> • Thread-Aktionen müssen in einer run-Methode stecken • Unterklasse von Thread nötig, bzw. Implementierung von Runnable

Thread-Synchronisation	
C#	Java
<pre>class Buffer { int[] data; public void Put(int x) { lock(this) { ... } } }</pre>	<pre>class Buffer { int[] data; public void put(int x) { synchronized(this) { ... } } }</pre>
<ul style="list-style-type: none"> • Monitor.Wait(this); • Monitor.Pulse(this); • Monitor.PulseAll(this); 	<ul style="list-style-type: none"> • wait(); • notify(); • notifyAll();

Anhang zu Aufgabe 3**Klasse RingBufferArray**

```

1  class RingBufferArray
2  {
3      protected Object[] array;
4      protected int front = 0;
5      protected int rear = 0;
6      private Object putLock = new Object();
7      private Object getLock = new Object();
8
9      public RingBufferArray(int n) {
10         array = new Object[n];
11     }
12
13     public void Put(Object x) {
14         lock (putLock) {
15             array[rear] = x;
16             rear = (rear + 1) % array.GetLength(0);
17         }
18     }
19
20     public Object Get() {
21         lock (getLock) {
22             Object x = array[front];
23             array[front] = null;
24             front = (front + 1) % array.GetLength(0);
25             return x;
26         }
27     }
28 }

```

Klasse BoundedBufferWithSemaphor

```

1  class BoundedBufferWithSemaphor : IQueue
2  {
3      protected Semaphore empty;
4      protected Semaphore full;
5      protected RingBufferArray buf;
6
7      public BoundedBufferWithSemaphor(int size) {
8         buf = new RingBufferArray(size);
9         empty = new Semaphore(size,size);
10        full = new Semaphore(0,size);
11    }
12
13    public void Enqueue(Object x) {
14        empty.WaitOne();
15        buf.Put(x);
16        full.Release();
17    }
18
19    public Object Dequeue(){
20        full.WaitOne();
21        Object x = buf.Get();
22        empty.Release();
23        return x;
24    }
25 }

```

Anhang zu Aufgabe 4**Socket-Kommunikation**

Sockets für Client und Server

Erzeugen eines Client Sockets:

```
TcpClient theClient;
try {
    theClient = new TcpClient(host,port);
}
catch (ArgumentOutOfRangeException) {
    // Port ausserhalb des erlaubten Bereichs
}
catch (SocketException) {
    // Fehler beim Zugriff auf den Socket
}
```

Informationen über die Socket-Verbindung

```
try {
    theClient = new TcpClient(host,port);
    Socket theSocket = theClient.Client;
    //...
    theSocket.LocalEndPoint; // Host + Port lokal (hier des Clients)
    theSocket.RemoteEndPoint; // Host + Port entfernt (hier des
                                // Servers)
    theSocket.ProtocolType; // Socket Protocol (hier Tcp)
    //...
    theClient.Close();
}
catch (Exception e) {
    // Fehler...
}
```

Erzeugen eines Server-Sockets:

```
try {
    TcpListener listen = new TcpListener (port);
    listen.Start();
    while (...) {
        TcpClient client = listen.AcceptTcpClient();
        //...Kommunikation mit Klient
        client.Close();
    }
}
catch (Exception) {
    // Server an diesem Port kann nicht erstellt werden
}
```

Anhang zu Aufgabe 5+6

Klasse AbstractServer

```
1 namespace ServerPattern
2 {
3     public abstract class AbstractServer
4     {
5         private Boolean running = true;
6         public const int DEFAULTPORT = 4711;
7         public const string DEFAULTHOST = "localhost";
8
9         protected void Run(Object port) {
10             IExecutor executor = CreateExecutor();
11             Socket listen = CreateServerSocket((int)port);
12             while (running) {
13                 AbstractHandler handler =
14                     CreateHandler(listen.Accept());
15                 executor.Execute(handler.Run);
16             }
17         }
18
19         virtual protected IExecutor CreateExecutor() {
20             return new PlainThreadExecutor();
21         }
22
23         protected Socket CreateServerSocket(int port) {
24             TcpListener listener = new TcpListener(port);
25             listener.Start();
26             return listener.Server;
27         }
28
29         public void Start() {
30             this.Start(0);
31         }
32
33         public void Start(int port) {
34             Thread server = new Thread(Run);
35             if (port <= 0)
36                 server.Start(DEFAULTPORT);
37             else
38                 server.Start(port);
39         }
40
41         abstract protected AbstractHandler
42             CreateHandler(Socket client);
43     }
44 }
```

Anhang zu Aufgabe 5+6

Klasse AbstractHandler

```
1 namespace ServerPattern
2 {
3     public abstract class AbstractHandler
4     {
5         private Socket client;
6
7         public AbstractHandler(Socket client) {
8             this.client = client;
9         }
10
11        public void Run() {
12            try {
13                if (ReadRequest()) {
14                    CreateResponse();
15                }
16                client.Close();
17            }
18            catch (Exception e) {
19                Console.Error.WriteLine(e);
20            }
21        }
22
23        abstract protected Boolean ReadRequest();
24        abstract protected void CreateResponse();
25    }
26 }
```

Lösungen

Aufgabe 1 (12 Punkte)

A.	S1 bis Sn werden <u>atomar</u> abgearbeitet. Das heisst, <u>kein Zwischenschritt</u> , <u>kein Zwischenergebnis</u> und <u>kein Zwischenzustand</u> ist in einem anderen parallel dazu abgearbeiteten Programmteil sichtbar.	3 Punkte
B.	Wenn <u>parallel arbeitende Programmteile</u> auf eine <u>gemeinsame genutzte Ressource</u> zugreifen wollen.	2 Punkte
C.	Sicherheit: Keine <u>Verklemmung (Deadlock)</u> durch <u>gegenseitige Zugriffe (Interferenzen)</u> in <u>kritischen Bereichen</u> .	2 Punkte
D.	Lebendigkeit: <u>Kein Livelock</u> . Jeder Programmteil erhält eine <u>faire Chance ausgeführt zu werden</u> . Chance kann <u>unconditionally fair</u> , <u>weak fair</u> oder <u>strong fair</u> sein.	2 Punkte
E.	Semaphor Einsatz für den <u>wechselseitigen Ausschluss</u> Semaphor ist eine <u>geschützte Instanz (Zähler)</u> Semaphor stellt die <u>Operationen P und V</u> und eine <u>Initialisierung</u> zur Verfügung	max. 3 Punkte

Aufgabe 2 (11 Punkte)

Je richtige und vollständige Zuordnung 1 Punkt

a. Der Thread befindet sich im new-Zustand.	57
b. Der Thread wird in den Ready-Zustand versetzt.	63
c. Diese Operation(en) des Threads werden (quasi)parallel abgearbeitet.	47 (6..32)
d. Der Thread wird in jedem Fall in den Blocked-Zustand versetzt.	-
e. Der Thread kommt vom Running-Zustand in den Objects-Wait-Pool.	26
f. Der Thread kommt vom Running-Zustand in den Objects-Lock-Pool.	6
g. Der Thread kommt vom Objects-Wait-Pool in den Objects-Lock-Pool.	26 (17,23)
h. Der Thread wird unterbrochen.	-
i. Ein Timeout wird gesetzt.	26
k. Der Thread wird in den Dead-Zustand versetzt.	48

Zusatzfrage

String whoseTurn in Objekt PingPong table	1 Punkt
-------------------------------------------	---------

Aufgabe 3 (15 Punkte)

A. Welches sind die gemeinsamen Ressourcen der Klasse `BoundedBuffer` ?

<code>protected final Object buffer[];</code>		1 Punkt
<code>protected int takePtr = 0;</code>	<code>\</code>	1 Punkt
<code>protected int putPtr = 0;</code>	<code>/</code>	1 Punkt
<code>protected int emptySlots;</code>	<code>\</code>	
<code>protected int usedSlots;</code>	<code>/</code>	1 Punkt

B. Warum müssen die `protected` Methoden `Insert` und `Extract` nicht synchronisiert sein ?

Weil sie nur innerhalb der <u>Methoden <code>Enqueue</code> und <code>Dequeue</code> im <code>synchronized</code> Blockes aufgerufen</u> werden.	2 Punkte
--------------------------------------------------------------------------------------------------------------------------------------------------	----------

C. Warum sind die öffentlichen Methoden `put` und `take` nicht synchronisiert ?

Weil nur die <u>kritischen Abschnitte</u> je durch <u>eigene <code>synchronized</code> Blöcke (lock-Pool Objekte)</u> geschützt sind. <u>Höhere Parallelität</u> und dadurch <u>bessere Performance</u> .	2 Punkte
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	----------

D. Welche Probleme könnten entstehen, wenn die öffentlichen Methoden `Enqueue` und `Dequeue` synchronisiert wären ? Geben Sie auch den Grund der Probleme an.

Deadlocks durch Nested Monitore	1 Punkt
Verschachtelung entsteht durch <code>synchronized</code> <code>putlock</code> und <code>takelock</code> Blöcke, sowie den Methoden <code>incUsedSlots</code> und <code>incEmptySlots</code> mit <code>synchronized</code> Blöcken	1 Punkt

E. Was passiert, wenn ein Thread die Methode `Enqueue` an einem vollen `Bounded Buffer` aufruft und danach `interrupted` wird ? Begründen Sie Ihre Antwort.

Beim Betreten der Methoden <code>Enqueue</code> wird der Thread in den <code>wait-Pool</code> von <code>putMonitor</code> gesetzt.	1 Punkt
Beim <code>interrupt</code> der Thread aus dem <code>wait-Pool</code> entfernt, ohne ein Element zu entfernen	1 Punkt
dadurch kann ein anderer Thread aus dem <code>wait-Pool</code> befreit werden (Zeile 54)	1 Punkt
schliesslich auf Zeile 55 die <code>ThreadInterruptedException</code> ausgelöst.	1 Punkt

Aufgabe 4 (13 Punkte)

A. zwei Nennungen

<ul style="list-style-type: none"> • Der EchoServer ist blockierend, d.h. er kann nur immer <u>einen Client zur gleichen Zeit</u> bearbeiten. • Der EchoServer ist nicht skalierbar • Der EchoServer ist nicht stabil, er stürzt bei einem Fehler ab • Es können nur C# Objekte verarbeitet werden. • Der EchoServer behandelt nur TCP-Sockets 	max. 2 Punkte
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------

B.

Der EchoServer wurde ohne Programmargumente (Port) gestartet..	1 Punkt
----------------------------------------------------------------	---------

C.

Der Client liefert keine C# Objekte.	1 Punkt
--------------------------------------	---------

D.

Der Client bricht die Verbindung gleich nach dem Aufbau wieder ab.	1 Punkt
--------------------------------------------------------------------	---------

E.

<p>Die <u>Exceptions</u> müssen zusätzlich <u>bei der while-Schleife abgefangen</u> werden.</p> <p>Zum Beispiel mit folgendem Code (nicht verlangt)</p> <pre> while (true) try { //...Codezeilen 11..18 } catch (Exception e) { //... } </pre>	2 Punkte
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	----------

F.

Punkte	Code
1	class EchoClient
	{
	public static void Main() {
1	TcpClient client = new TcpClient("localhost", 7788);
	BinaryFormatter binFormatter = new BinaryFormatter();
1	NetworkStream stream = new NetworkStream(client.Client);
1	binFormatter.Serialize(stream, new MyClass("Köln", 4711));
1	stream.Flush();
	MyClass t = (MyClass)binFormatter.Deserialize(stream);
	Console.WriteLine(t.getId() + "(" + t.getValue() + ")");
	}
	}
4	Total

G.

Der Server muss die zu serialisierende Klasse kennen	2 Punkte
------------------------------------------------------	----------

Aufgabe 5 (6 Punkte)

Je Fehler und Korrektur 2 Punkte. Max. 6 Punkte

Fehler: Server hört am Standardport (4711) auf Verbindungsanfragen

Korrektur: Konstruktor mit Port-Parameter von AbstractServer überschreiben

Fehler: Browser stellt falsche Verbindungsanfrage

Korrektur: Verbindungsanfrage von Browser: <http://172.21.1.104:4711>

Fehler: Server hört nicht auf http-Anfragen (Fehler in readRequest)

Korrektur: Zeile 19 in readRequest: GET

Fehler: Fire Wall aktiv auf Server oder Client Seite

Korrektur: Fire Wall richtig konfigurieren

Fehler: Keine Netzwerkverbindung (Grund: Kabel nicht angeschlossen, Netzwerkkarte defekt / falsch konfig., Router defekt / falsch konfig., Telefonzentrale defekt / überlastet)

Korrektur: Netzwerkverbindung reparieren

Aufgabe 6 (12Punkte)

Punkte

A. • URL in http-Anfrage herausfiltern	1
• mit URL Aktion in Liste suchen und ausführen	1
• in <code>CreateResponse</code> den entsprechenden String senden	1
B. • Client Socket überprüfen mit Hilfe der Eigenschaft <code>client.RemoteEndPoint</code> . Methode <code>ReadRequest</code> wird nur bei bekannten Socket-Verbindungen <code>true</code> .	3
• SSL Verbindung mit Client und Client-Zertifikat verlangen	
• Benutzer Login in der Methode <code>ReadRequest</code> durchführen.	
C. Executor einsetzen mit	1
1 Thread (minimale Systembelastung) und	1
großem Buffer (viele Benutzeranfragen möglich)	1
D. Client nimmt Verbindung auf und	1
bricht Socket-Verbindung wieder ab,	1
ohne eine http-Anfrage zu senden	1
Total	12