Name: _____

Beispielaufgabe

Informatik, Leistungskurs

Aufgabenstellung

Am Konrad-Zuse-Gymnasium haben die Schülerinnen und Schüler ihre Kurswahlen für die Einführungsphase bisher durch Ausfüllen eines Wahlbogens erledigt. Der Informatikleistungskurs will dieses Verfahren in Form einer Client-Server-Anwendung realisieren.
Die Schülerinnen und Schüler haben schon eindeutige Benutzernamen und zufällig generierte Passwörter erhalten.
(Die Benutzernamen, die Passwörter, die Fachbezeichnungen und Fachkürzel enthalten dabei keine Leerzeichen.)
Im Gespräch mit der Oberstufenkoordinatorin hat der Informatikleistungskurs die ersten Anforderungen herausgearbeitet. Eine Gruppe von Schülerinnen und Schülern kümmert sich in der Entwicklungsphase um ein Kommunikationsprotokoll und schlägt folgendes vor:

Client sendet an Server	Server sendet an Client
ANMELDEN <Benutzername> <Passwort>	+OK ANGEMELDET <Benutzername> -ERR Fehler bei Anmeldung -ERR bereits angemeldet
FACHANWAHL <Fachkürzel>	+OK GEWAEHLT -ERR nicht angemeldet
FACHABWAHL <Fachkürzel>	+OK ABGEWAEHLT -ERR nicht angemeldet
ALLE_FAECHER	+OK ALLE_FAECHER_DER_SCHULE <Fachkürzel_1> <Fachbezeichnung_1>; <Fachkürzel_2> <Fachbezeichnung_2>; … ; <Fachkürzel_n> <Fachbezeichnung_n>; -ERR nicht angemeldet
ABMELDEN	+OK ABGEMELDET Nach dem Senden trennt der Server die Verbindung.
Anderer Text	-ERR

Tabelle 1: Kommunikationsprotokoll

Name: _____

Abbildung 1 zeigt einen Ausschnitt des Implementationsdiagramms. Eine Dokumentation relevanter Klassen finden Sie im Anhang.

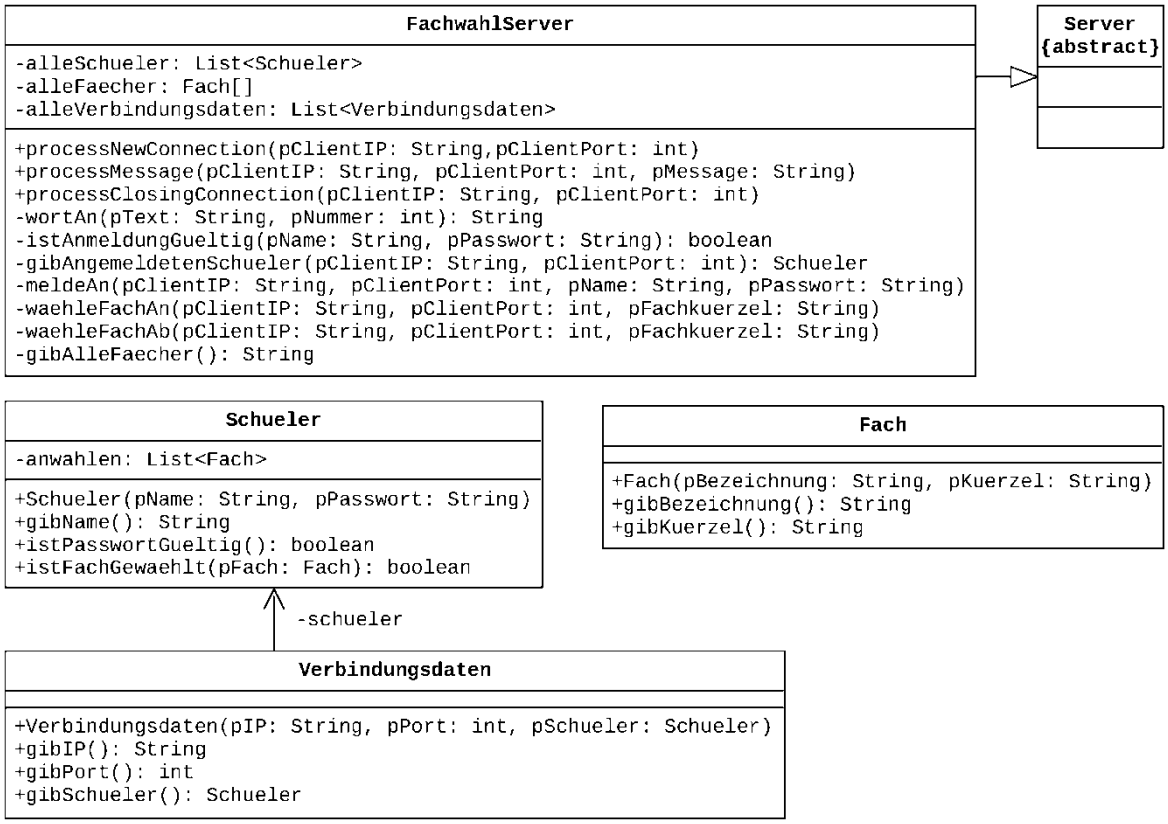

Abbildung 1: Implementationsdiagramm (Teilmodellierung)

a) Folgende beispielhafte Kommunikation ist gegeben:
Der Schüler mit dem Benutzernamen `MaxMustermann` (und seinem Passwort: `gOn1Ce#`) benutzt nun einen Client, um seine Kurswahlen zu tätigen. Er führt dabei folgende Aktionen aus:
   - Er meldet sich mit seinem Benutzernamen und seinem Passwort an, dabei vertippt er sich jedoch bei dem Passwort.
   - Dann meldet er sich mit den korrekten Zugangsdaten an.
   - Er wählt das Fach Deutsch (Fachkürzel: D) an.
   - Er lässt sich die Übersicht aller Fächer ausgeben. (In dieser Programmversion gibt es lediglich folgende Fächer: Deutsch (D), Mathematik (M), Informatik (If), Sport (Sp), Englisch (E), Französisch (F) und Latein (L)
   - Er wählt die Fächer Informatik (If) und Sport (Sp) an.
   - Er wählt das Fach Deutsch (D) wieder ab.
   - Schließlich meldet er sich ab.

Name: _____

Stellen Sie die beispielhafte Kommunikation zwischen dem Client und dem Server nach den Vorgaben aus Tabelle 1 tabellarisch dar.

Erweitern Sie das Kommunikationsprotokoll um mindestens zwei geeignete Fehlermeldungen bei der Abwahl eines Faches.

(10 Punkte)

b) Ein Schüler hat angefangen die Methode `processMessage` in der Klasse `Fachwahlserver` nach den Vorgaben aus dem Kommunikationsprotokoll zu implementieren. Ein Bereich (vgl. Zeile 18) fehlt allerdings noch.

```
 1 public void processMessage(String pClientIP, int pClientPort,
                              String pMessage) {
 2   String befehl = wortAn(pMessage, 1);
 3   String wort2 = wortAn(pMessage, 2);
 4   String wort3 = wortAn(pMessage, 3);
 5   Schueler aktuellAngemeldeterSchueler
         = gibAngemeldetenSchueler(pClientIP, pClientPort);
 6   if (aktuellAngemeldeterSchueler == null) {
 7     if (befehl.equals("ANMELDEN")) {
 8       if (istAnmeldungGueltig(wort2, wort3)) {
 9         meldeAn(pClientIP, pClientPort, wort2, wort3);
10         send(pClientIP, pClientPort, "ANGEMELDET " + wort2);
11       } else {
12         send(pClientIP, pClientPort,
                "-ERR Fehler bei Anmeldung");
13       }
14     } else {
15       send(pClientIP, pClientPort, "-ERR nicht angemeldet");
16     }
17   } else {
18     /* Aufgabe: Vervollständigen Sie die Implementierung in
        *          diesem Teil! */
     }
   }
```

Erläutern Sie die Zeilen 5-17 im Sachzusammenhang!

Vervollständigen Sie die Implementierung ab Zeile 18 gemäß den Vorgaben aus dem ursprünglichen Kommunikationsprotokoll (vgl. Tabelle 1).

Hinweis: Nutzen Sie die vorhandenen Methoden im Anhang.

(12 Punkte)

Name: _____

c) Ein Schüler hat begonnen eine Erweiterung zu programmieren. Dazu hat er zwei neue Methoden implementiert.

Neue Methode in der Klasse `Fachwahlserver`:

```
1 public void sendeEtwas(String pClientIP, int pClientPort) {
2   Schueler aktuellAngemeldeterSchueler
        = gibAngemeldetenSchueler(pClientIP, pClientPort);
3   if (aktuellAngemeldeterSchueler != null) {
4     send(pClientIP, pClientPort, "+OK ANTWORT "
           + aktuellAngemeldeterSchueler.gibAntwort());
5   }
6 }
```

Neue Methode in der Klasse `Schueler`:

```
 1 public String gibAntwort() {
 2   boolean[] markierungen = new boolean[4];
 3   for (int i = 0; i < markierungen.length; i++) {
 4     markierungen[i] = false;
 5   }
 6   anwahlen.toFirst();
 7   while (anwahlen.hasAccess()) {
 8     String fachname = anwahlen.getContent().gibBezeichnung();
 9     switch (fachname) {
10       case "Deutsch":     markierungen[0] = true; break;
11       case "Mathematik":  markierungen[1] = true; break;
12       case "Sport":       markierungen[2] = true; break;
13       case "Englisch":    markierungen[3] = true; break;
14       case "Französisch": markierungen[3] = true; break;
15       case "Latein":      markierungen[3] = true; break;
16     }
17     anwahlen.next();
18   }
19   String rueckgabe = "erfuellt";
20   for (int i = 0; i < markierungen.length; i++) {
21     if (!markierungen[i]) {
22       rueckgabe = "nicht erfuellt";
23     }
24   }
25   return rueckgabe;
26 }
```

Analysieren Sie die Methoden und erläutern Sie die Funktionalität im Sachkontext.

(10 Punkte)

Name: _____

d) In dieser Aufgabe soll der `Fachwahlserver` für die Stufenleitung so erweitert werden, dass für alle Fächer die Benutzernamen von den Schülerinnen und Schülern ausgegeben werden, die das jeweilige Fach gewählt haben.
Dazu soll eine neue Methode `sendeAnwahlenFuerAlleFaecher` entwickelt werden. Die Methode hat den folgenden Methodenkopf:

```
public void sendeAnwahlenFuerAlleFaecher(String pClientIP,
                                         int pClientPort)
```

Die Methode sendet an den Benutzer mit der IP-Adresse `pClientIP` und der Port-Nummer `pClientPort` eine Nachricht in Form eines Strings mit einer Aufstellung für alle Fächer und den jeweiligen Benutzernamen von den Schülerinnen und Schülern, die das jeweilige Fach gewählt haben.
Aufbau:

```
+OK Anwahlen:
<Fachbezeichnung_1> <Benutzer_1> <Benutzer_2> … <Benutzer_n'>;
<Fachbezeichnung_2> <Benutzer_1> <Benutzer_2> … <Benutzer_n''>;
…
<Fachbezeichnung_n> <Benutzer_1> <Benutzer_2> … <Benutzer_n'''>;
```

Hinweis: Die zu sendende Nachricht ist einzeilig.

Implementieren Sie diesen Algorithmus in der Methode `sendeAnwahlenFuerAlleFaecher` *der Klasse* `Fachwahlserver`*.*
Dokumentieren Sie Ihre Implementation durch geeignete Kommentare im Quelltext.
(10 Punkte)

e) In der bisherigen Programmversion werden alle Informationen als Klartext übertragen. In der nächsten Programmversion soll die Kommunikation zwischen Client und Server mit einem asymmetrischen Verschlüsselungsverfahren gesichert werden.

Folgende Abbildung stellt die konsequente Anwendung eines asymmetrischen Verschlüsselungsverfahrens zwischen Client und Server dar. Dabei hat sich der Schüler mit dem korrekten Benutzernamen `HeinzNixdorf` und seinem gültigen Passwort `gO#Iee7` bei der Anmeldung vertippt.

Name: _____

Es wird das Kommunikationsprotokoll aus Tabelle 1 zugrunde gelegt.

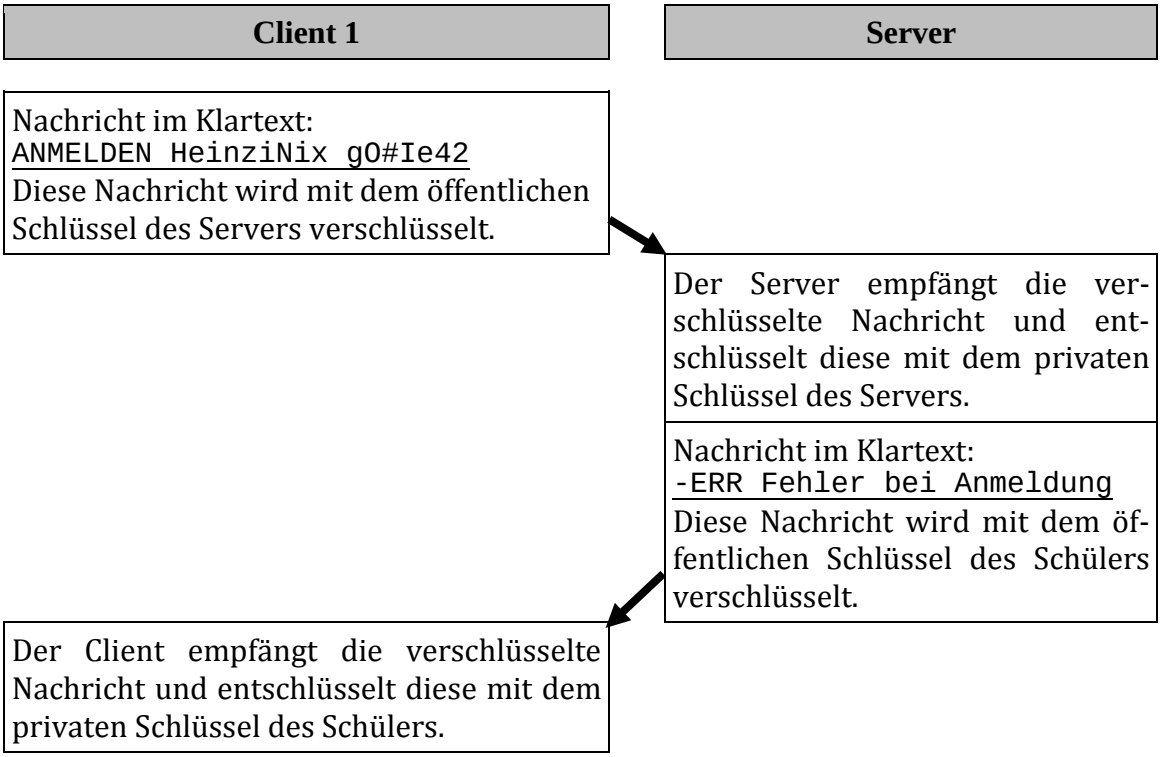

Abbildung 2: Konsequente Anwendung eines asymmetrischen Verschlüsselungsverfahrens zwischen Client und Server

Eine Schülerin behauptet, dass bei der konsequenten Anwendung eines asymmetrischen Verschlüsselungsverfahrens für alle zu übermittelnden Nachrichten ein Problem auftaucht, wenn ein Anmeldeversuch scheitert.

Beurteilen Sie die Aussage der Schülerin zum Problem bei gescheiterten Anmeldeversuchen.

(8 Punkte)

Zugelassene Hilfsmittel:

- Wörterbuch zur deutschen Rechtschreibung
- Taschenrechner (grafikfähiger Taschenrechner / CAS-Rechner)

Name: _____

Anhang

Die Klasse `FachwahlServer`

Diese Klasse ist Unterklasse der Klasse `Server`. Objekte dieser Klassen verwalten die am Server angemeldeten Schülerinnen und Schüler. Dabei stellt sie neben den geerbten Methoden unter anderem folgende Methoden zur Verfügung:

Ausschnitt aus der Dokumentation der Klasse `FachwahlServer`

```
void processClosingConnection(String pClientIP, int pClientPort,
                              String pMessage)
```

Das Verbindungsdaten-Objekt des angemeldeten Schülers mit der IP-Adresse `pClientIP` und der Port-Nummer `pClientPort` wird aus der Liste `alleVerbindungsdaten` gelöscht. Dem Schüler wird die Nachricht `"+OK ABGEMELDET"` gesendet. Die Verbindung zum Client wird getrennt.

Zusätzlich verfügt die Klasse `FachwahlServer` über die folgenden privaten Methoden:

```
String wortAn(String pText, int pNummer)
```

Diese Anfrage liefert das Wort mit der angegebenen Wortnummer `pNummer` in dem angegebenen Text `pText`. Die Wörter sind dabei beginnend mit `1` nummeriert und durch Leerzeichen voneinander getrennt. Falls kein Wort an der angegebenen Wortnummer existiert, wird ein leerer String zurückgegeben.

```
boolean istAnmeldungGueltig(String pName, String pPasswort)
```

Die Anfrage liefert den Wert `true`, wenn es einen Schüler mit `pName` als Benutzernamen und dem dazugehörigen Passwort `pPasswort` gibt. Sonst liefert sie den Wert `false`.

```
Schueler gibAngemeldetenSchueler(String pClientIP, int pClientPort)
```

Wenn es einen angemeldeten Schüler mit der IP-Adresse `pClientIP` und der Port-Nummer `pClientPort` gibt, wird dieses Objekt der Klasse `Schueler` zurückgeliefert und sonst `null`.

```
void meldeAn(String pClientIP, int pClientPort, String pName,
             String pPasswort)
```

Der Schüler mit dem Benutzernamen `pName` und mit dem dazugehörigen Passwort `pPasswort` wird mit der IP-Adresse `pClientIP` und der Port-Nummer `pClientPort` angemeldet. Ein neues Verbindungsdaten-Objekt mit den Informationen `pClientIP`, `pClientPort` und der Assoziation auf das zugehörige Schüler-Objekt wird initialisiert und in der Liste `alleVerbindungsdaten` verwaltet.

Name: _____

```
void waehleFachAn(String pClientIP, int pClientPort,
                  String pFachkuerzel)
```

Für den angemeldeten Schüler mit der IP-Adresse `pClientIP` und der Port-Nummer `pClientPort` wird das Fach mit dem Kürzel `pFachkuerzel` angewählt.

```
void waehleFachAb(String pClientIP, int pClientPort,
                  String pFachkuerzel)
```

Für den angemeldeten Schüler mit der IP-Adresse `pClientIP` und der Port-Nummer `pClientPort` wird das Fach mit dem Kürzel `pFachkuerzel` abgewählt.

```
String gibAlleFaecher()
```

Alle Fächer werden als String mit Leerzeichen getrennt zurückgegeben. Die Rückgabe ist wie folgt aufgebaut:

```
<Fachkürzel_1> <Fachbezeichnung_1>;
<Fachkürzel_2> <Fachbezeichnung_2>; … ;
<Fachkürzel_n> <Fachbezeichnung_n>;
```

Die Klasse `Schueler`

Objekte dieser Klasse verwalten die Daten einer Schülerin bzw. eines Schülers sowie die bisher angewählten Fächer. Die Klasse stellt unter anderem folgende Methoden zur Verfügung:

Ausschnitt aus der Dokumentation der Klasse `Schueler`

```
Schueler(String pName, String pPasswort)
```

Die Methode erstellt ein neues Objekt der Klasse `Schueler` mit dem Benutzernamen `pName` und dem Passwort `pPasswort`. Die von dem Objekt verwaltete Datensammlung der angewählten Fächer ist anfangs leer. Die Verbindungsdaten eines neuen Teilnehmer-Objekts sind zunächst mit `null` initialisiert und werden erst bei Anmeldung durch einen Client gesetzt.

```
boolean istPasswortGueltig(String pPasswort)
```

Die Anfrage liefert dann `true` zurück, wenn das im Parameter übergebene Passwort `pPasswort` mit dem Passwort des Objekts der Klasse `Schueler` übereinstimmt. Sonst wird `false` zurückgegeben.

```
boolean istFachGewaehlt(Fach pFach)
```

Die Anfrage liefert dann `true` zurück, wenn das im Parameter übergebene Fach `pFach` von dem Schüler bereits gewählt wurde. Sonst wird `false` zurückgegeben.

Name: _____

Die Klasse `Server`

Objekte von Unterklassen der abstrakten Klasse **`Server`** ermöglichen das Anbieten von Serverdiensten, so dass Clients Verbindungen zum Server mittels TCP/IP-Protokoll aufbauen können. Zur Vereinfachung finden Nachrichtenversand und -empfang zeilenweise statt, d. h., beim Senden einer Zeichenkette wird ein Zeilentrenner ergänzt und beim Empfang wird dieser entfernt. Verbindungsannahme, Nachrichtenempfang und Verbindungsende geschehen nebenläufig. Auf diese Ereignisse muss durch Überschreiben der entsprechenden Ereignisbehandlungsmethoden reagiert werden. Es findet nur eine rudimentäre Fehlerbehandlung statt, so dass z.B. Verbindungsabbrüche nicht zu einem Programmabbruch führen. Einmal unterbrochene oder getrennte Verbindungen können nicht reaktiviert werden.

Dokumentation der Klasse `Server`

`Server(int pPort)`

Ein Objekt vom Typ `Server` wird erstellt, das über die angegebene Portnummer einen Dienst anbietet an. Clients können sich mit dem Server verbinden, so dass Daten (Zeichenketten) zu diesen gesendet und von diesen empfangen werden können. Kann der Server unter der angegebenen Portnummer keinen Dienst anbieten (z.B. weil die Portnummer bereits belegt ist), ist keine Verbindungsaufnahme zum Server und kein Datenaustausch möglich.

`boolean isOpen()`

Die Anfrage liefert den Wert `true`, wenn der Server auf Port `pPort` einen Dienst anbietet. Ansonsten liefert die Methode den Wert `false`.

`boolean isConnectedTo(String pClientIP, int pClientPort)`

Die Anfrage liefert den Wert `true`, wenn der Server mit dem durch `pClientIP` und `pClientPort` spezifizierten Client aktuell verbunden ist. Ansonsten liefert die Methode den Wert `false`.

`void send(String pClientIP, int pClientPort, String pMessage)`

Die Nachricht `pMessage` wird – um einen Zeilentrenner erweitert – an den durch `pClientIP` und `pClientPort` spezifizierten Client gesendet. Schlägt der Versand fehl, geschieht nichts.

`void sendToAll(String pMessage)`

Die Nachricht `pMessage` wird – um einen Zeilentrenner erweitert – an alle mit dem Server verbundenen Clients gesendet. Schlägt der Versand an *einen* Client fehl, wird dieser Client übersprungen.

Name: _____

`void closeConnection(String pClientIP, int pClientPort)`

Die Verbindung des Servers zu dem durch `pClientIP` und `pClientPort` spezifizierten Client wird getrennt. Zuvor wird die Methode `processClosingConnection` mit IP-Adresse und Port des jeweiligen Clients aufgerufen. Ist der Server nicht mit dem in der Parameterliste spezifizierten Client verbunden, geschieht nichts.

`void close()`

Alle bestehenden Verbindungen zu Clients werden getrennt und der Server kann nicht mehr verwendet werden. Ist der Server bereits vor Aufruf der Methode in diesem Zustand, geschieht nichts.

`void processNewConnection(String pClientIP, int pClientPort)`

Diese Ereignisbehandlungsmethode wird aufgerufen, wenn sich ein Client mit IP-Adresse `pClientIP` und Portnummer `pClientPort` mit dem Server verbunden hat. Die Methode ist abstrakt und muss in einer Unterklasse der Klasse `Server` überschrieben werden, so dass auf den Neuaufbau der Verbindung reagiert wird. Der Aufruf der Methode erfolgt nicht synchronisiert.

`void processMessage(String pClientIP, int pClientPort, String pMessage)`

Diese Ereignisbehandlungsmethode wird aufgerufen, wenn der Server die Nachricht `pMessage` von dem durch `pClientIP` und `pClientPort` spezifizierten Client empfangen hat. Der vom Client hinzugefügte Zeilentrenner wurde zuvor entfernt. Die Methode ist abstrakt und muss in einer Unterklasse der Klasse `Server` überschrieben werden, so dass auf den Empfang der Nachricht reagiert wird. Der Aufruf der Methode erfolgt nicht synchronisiert.

`void processClosingConnection(String pClientIP, int pClientPort)`

Sofern der Server die Verbindung zu dem durch `pClientIP` und `pClientPort` spezifizierten Client trennt, wird diese Ereignisbehandlungsmethode aufgerufen, unmittelbar *bevor* die Verbindungstrennung tatsächlich erfolgt. Wird die Verbindung unvermittelt unterbrochen oder hat der in der Parameterliste spezifizierte Client die Verbindung zum Server unvermittelt getrennt, erfolgt der Methodenaufruf *nach* der Unterbrechung/Trennung der Verbindung. Die Methode ist abstrakt und muss in einer Unterklasse der Klasse `Server` überschrieben werden, so dass auf das Ende der Verbindung zum angegebenen Client reagiert wird. Der Aufruf der Methode erfolgt nicht synchronisiert.

Name: _____

Die generische Klasse `List<ContentType>`

Objekte der generischen Klasse **`List`** verwalten beliebig viele, linear angeordnete Objekte vom Typ `ContentType`. Auf höchstens ein Listenobjekt, aktuelles Objekt genannt, kann jeweils zugegriffen werden. Wenn eine Liste leer ist, vollständig durchlaufen wurde oder das aktuelle Objekt am Ende der Liste gelöscht wurde, gibt es kein aktuelles Objekt. Das erste oder das letzte Objekt einer Liste können durch einen Auftrag zum aktuellen Objekt gemacht werden. Außerdem kann das dem aktuellen Objekt folgende Listenobjekt zum neuen aktuellen Objekt werden.

Das aktuelle Objekt kann gelesen, verändert oder gelöscht werden. Außerdem kann vor dem aktuellen Objekt ein Listenobjekt eingefügt oder ein Listenobjekt an das Ende der Liste angefügt werden.

Dokumentation der Klasse `List`

`List<ContentType>()`

Eine leere Liste wird erzeugt.

`boolean isEmpty()`

Die Anfrage liefert den Wert `true`, wenn die Liste keine Objekte enthält, sonst liefert sie den Wert `false`.

`boolean hasAccess()`

Die Anfrage liefert den Wert `true`, wenn es ein aktuelles Objekt gibt, sonst liefert sie den Wert `false`.

`void next()`

Falls die Liste nicht leer ist, es ein aktuelles Objekt gibt und dieses nicht das letzte Objekt der Liste ist, wird das dem aktuellen Objekt in der Liste folgende Objekt zum aktuellen Objekt, andernfalls gibt es nach Ausführung des Auftrags kein aktuelles Objekt, d. h., `hasAccess()` liefert den Wert `false`.

`void toFirst()`

Falls die Liste nicht leer ist, wird das erste Objekt der Liste aktuelles Objekt. Ist die Liste leer, geschieht nichts.

`void toLast()`

Falls die Liste nicht leer ist, wird das letzte Objekt der Liste aktuelles Objekt. Ist die Liste leer, geschieht nichts.

Name: _____

`ContentType getContent()`

Falls es ein aktuelles Objekt gibt (`hasAccess() == true`), wird das aktuelle Objekt zurückgegeben, andernfalls (`hasAccess() == false`) gibt die Anfrage den Wert `null` zurück.

`void setContent(ContentType pContent)`

Falls es ein aktuelles Objekt gibt (`hasAccess() == true`) und `pContent` ungleich `null` ist, wird das aktuelle Objekt durch `pContent` ersetzt. Sonst bleibt die Liste unverändert.

`void append(ContentType pContent)`

Ein neues Objekt `pContent` wird am Ende der Liste eingefügt. Das aktuelle Objekt bleibt unverändert. Wenn die Liste leer ist, wird das Objekt `pContent` in die Liste eingefügt und es gibt weiterhin kein aktuelles Objekt (`hasAccess() == false`). Falls `pContent` gleich `null` ist, bleibt die Liste unverändert.

`void insert(ContentType pContent)`

Falls es ein aktuelles Objekt gibt (`hasAccess() == true`), wird ein neues Objekt vor dem aktuellen Objekt in die Liste eingefügt. Das aktuelle Objekt bleibt unverändert. Falls die Liste leer ist und es somit kein aktuelles Objekt gibt (`hasAccess() == false`), wird `pContent` in die Liste eingefügt und es gibt weiterhin kein aktuelles Objekt. Falls es kein aktuelles Objekt gibt (`hasAccess() == false`) und die Liste nicht leer ist oder `pContent` gleich `null` ist, bleibt die Liste unverändert.

`void concat(List pList)`

Die Liste `pList` wird an die Liste angehängt. Anschließend wird `pList` eine leere Liste. Das aktuelle Objekt bleibt unverändert. Falls `pList null` oder eine leere Liste ist, bleibt die Liste unverändert.

`void remove()`

Falls es ein aktuelles Objekt gibt (`hasAccess() == true`), wird das aktuelle Objekt gelöscht und das Objekt hinter dem gelöschten Objekt wird zum aktuellen Objekt. Wird das Objekt, das am Ende der Liste steht, gelöscht, gibt es kein aktuelles Objekt mehr (`hasAccess() == false`). Wenn die Liste leer ist oder es kein aktuelles Objekt gibt (`hasAccess() == false`), bleibt die Liste unverändert.

Unterlagen für die Lehrkraft

Beispielaufgabe

Informatik, Leistungskurs

1. Aufgabenart

Analyse, Modellierung und Implementation von kontextbezogenen Problemstellungen mit Schwerpunkt auf den Inhaltsfeldern Daten und ihre Strukturierung, Algorithmen und Informatiksysteme

2. Aufgabenstellung

siehe Prüfungsaufgabe

3. Materialgrundlage

entfällt

4. Bezüge zu dem Kernlehrplan und zu den Vorgaben

Die Aufgaben weisen vielfältige Bezüge zu den Kompetenzerwartungen und Inhaltsfeldern des Kernlehrplans bzw. zu den in den Vorgaben ausgewiesenen Fokussierungen auf. Im Folgenden wird auf Bezüge von zentraler Bedeutung hingewiesen:

1. Inhaltsfelder und inhaltliche Schwerpunkte
   Daten und ihre Strukturierung:
   - Objekte und Klassen (lineare Strukturen)

   Algorithmen:
   - Analyse, Entwurf und Implementierung von Algorithmen
   - Algorithmen in ausgewählten informatischen Kontexten (Algorithmen zur Kommunikation in Netzwerken (Klassen `Connection`, `Client`, `Server`))

   Formale Sprachen und Automaten:
   - Syntax und Semantik einer Programmiersprache (Java)

   Informatiksysteme:
   - Einzelrechner und Rechnernetzwerke
   - Sicherheit

2. Medien / Materialien
   – keine –

5. Zugelassene Hilfsmittel

- Wörterbuch zur deutschen Rechtschreibung
- Taschenrechner (grafikfähiger Taschenrechner / CAS-Rechner)

6. Modelllösungen

Die Modelllösung stellt eine mögliche Lösung bzw. Lösungsskizze dar. Der gewählte Lösungsansatz und –weg der Schülerinnen und Schüler muss nicht identisch mit dem der Modelllösung sein. Sachlich richtige Alternativen werden mit entsprechender Punktzahl bewertet (Bewertungsbogen: Zeile „Sachlich richtige Lösungsalternative zur Modelllösung“).

Teilaufgabe a)

Beispielhafte Kommunikation zwischen Client und Server:

Client sendet an Server	Server sendet an Client
ANMELDEN MaxMustermann sOn1Ce#	-ERR Fehler bei Anmeldung
ANMELDEN MaxMustermann gOn1Ce#	+OK ANGEMELDET MaxMustermann
FACHANWAHL D	+OK GEWAEHLT
ALLE_FAECHER	+OK ALLE_FAECHER_DER_SCHULE D Deutsch; M Mathematik; If Informatik; Sp Sport; E Englisch; F Französisch; L Latein
FACHANWAHL If	+OK GEWAEHLT
FACHANWAHL Sp	+OK GEWAEHLT
FACHABWAHL D	+OK ABGEWAEHLT
ABMELDEN	+OK ABGEMELDET

Erweitertes Kommunikationsprotokoll:

FACHABWAHL <Fachkürzel>	+OK ABGEWAEHLT -ERR FACH <Fachkürzel> existiert nicht -ERR FACH <Fachkürzel> war nicht gewaehlt -ERR nicht angemeldet

Teilaufgabe b)

Erläuterung im Sachzusammenhang:
In Zeile 6 wird überprüft, ob es mit der IP-Adresse und dem Port einen nicht angemeldeten Schüler gibt. Wenn der Schüler nicht angemeldet ist, muss die Nachricht mit dem Schlüsselwort `ANMELDEN` beginnen (vgl. Zeile 7). Falls dies nicht der Fall ist, dann wird die Fehlermeldung `–ERR nicht angemeldet` in Zeile 15 zurückgegeben. Falls das Schlüsselwort `ANMELDEN` lautete, wird anhand der zwei folgenden Worte der Nachricht (der Benutzername und das Passwort) überprüft, ob eine Anmeldung möglich ist (vgl. Zeile 8). Falls ja, dann wird der Benutzer mit seiner IP-Adresse und dem Port angemeldet (vgl. Zeile 9). Zusätzlich wird dem Benutzer die Nachricht `ANGEMELDET + <Benutzername>` gesendet.

Falls die Anmeldung fehlschlug, wird dem Benutzer die Fehlermeldung `-ERR Fehler bei Anmeldung` gesendet (vgl. Zeile 12). In Zeile 17 beginnt der Sonst-Teil, für den Fall, dass es einen angemeldeten Benutzer gibt.

Implementierung des fehlenden Bereichs (vgl. Zeile 18):

```
if (befehl.equals("ANMELDEN")) {
  send(pClientIP, pClientPort, "-ERR bereits angemeldet");
} else if (befehl.equals("ABMELDEN")) {
  closeConnection(pClientIP, pClientPort);
} else if (befehl.equals("FACHANWAHL")) {
  waehleFachAn(pClientIP, pClientPort, wort2);
  send(pClientIP, pClientPort, "+OK GEWAEHLT");
} else if (befehl.equals("FACHABWAHL")) {
  waehleFachAb(pClientIP, pClientPort, wort2);
  send(pClientIP, pClientPort, "+OK ABGEWAEHLT");
} else if (befehl.equals("ALLE_FAECHER")) {
  send(pClientIP, pClientPort,
               "+OK ALLE_FAECHER_DER_SCHULE " + gibAlleFaecher());
} else {
  send(pClientIP, pClientPort, "-ERR");
}
```

Teilaufgabe c)

Die Methode `sendeEtwas` überprüft zunächst, ob es einen angemeldeten Schüler mit der IP-Adresse und dem Port gibt. Falls dies der Fall ist, wird diesem Client die Nachricht `+OK Antwort <Ergebnis der Methode gibAntwort>` gesendet. Sonst geschieht nichts.

Die Methode `gibAntwort` führt (in einer kleinen Form) eine Überprüfung der Belegpflicht für die Kurswahlen eines Schülers durch. Wenn die Belegpflicht erfüllt ist, dann wird `"erfuellt"` und sonst `"nicht erfuellt"` zurückgeliefert. Damit `"erfuellt"` zurückgeliefert wird, muss der Schüler die Fächer Deutsch, Mathematik, Sport und mindestens eine Fremdsprache (Englisch, Französisch oder Latein) gewählt haben.

In Zeile 2 wird ein Array mit vier Fächern deklariert und initialisiert. In den Zeilen 3- 5 werden die Arrayfächer mit `false` vorbelegt. Die Liste der Anwahlen des Schülers wird durchlaufen (vgl. Zeilen 7-18). Das Arrayfach mit dem Index `0` wird auf `true` gesetzt, wenn das Fach Deutsch in der Liste der Anwahlen vorkommt. Für das Fach Mathematik ist das Arrayfach mit dem Index `1` und für das Fach Sport das Arrayfach mit dem Index `2` vorgesehen. Das Arrayfach mit dem Index `3` wird auf `true` gesetzt, wenn mindestens eine der Fremdsprachen Englisch, Französisch oder Latein vorhanden ist.

In Zeile 19 wird die lokale Variable `rueckgabe` auf `"erfuellt"` gesetzt. Die Schleife (vgl. Zeile 20-24) durchläuft das Array `markierungen`. Falls in dem Feld eine Markierung `false` ist, wird die Variable `rueckgabe` auf `"nicht erfuellt"` gesetzt.
Der Wert der Variablen `rueckgabe` wird zurückgeliefert.

Teilaufgabe d)

Implementation:

```
void sendeAnwahlenFuerAlleFaecher(String pClientIP,
                                  int pClientPort) {
  String ausgabe = "+OK Anwahlen: ";
  // Durchlaufe das Array mit allen Fächern.
  for (int i = 0; i < alleFaecher.length; i++) {
    // Hänge an die Ausgabe ein die Fachbezeichnung an.
    ausgabe = ausgabe + alleFaecher[i].gibBezeichnung();
    alleSchueler.toFirst();
    // Durchlaufe die Liste mit allen Schülern.
    while (alleSchueler.hasAccess()) {
      Schueler aktuellerSchueler = alleSchueler.getContent();
      /* Falls der aktuelle Schüler das Fach gewählt hat,
       * dann hänge an die Ausgabe den Schülernamen an. */
      if (aktuellerSchueler.istFachGewaehlt(alleFaecher[i])) {
        ausgabe = ausgabe +  " " + aktuellerSchueler.gibName();
      }
      alleSchueler.next();
    }
    ausgabe = ausgabe + "; ";
  }
  // sende die Ausgabe an den entsprechenden Client
  send(pClientIP, pClientPort, ausgabe);
}
```

Teilaufgabe e)

Wenn ein Anmeldeversuch scheitert, kann der Server nicht wissen, mit welchem öffentlichen Schlüssel diese negative Rückmeldung verschlüsselt werden soll. Dies ist insbesondere der Fall, wenn sich eine Person bei der Eingabe des eigenen Benutzernamens vertippt. Nur bei richtiger Eingabe von Benutzernamen und Passwort kann der Server zweifelsfrei den Benutzer authentifizieren und ihm anschließend eine Nachricht senden, die mit dem öffentlichen Schlüssel des Benutzers verschlüsselt ist.
Das angeführte Problem in der Kommunikation zwischen Client und Server unter Anwendung eines asymmetrischen Verschlüsselungsverfahrens muss als Sonderfall berücksichtigt werden. Es ist als mögliche Lösung in diesem Sonderfall tolerierbar, dass die negative Rückmeldung auch unverschlüsselt an den Client gesendet wird.

7. Teilleistungen – Kriterien / Bewertungsbogen zur Prüfungsarbeit

Name des Prüflings: _____ Kursbezeichnung: _____

Schule: _____

Teilaufgabe a)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl (AFB)**	**EK**[1]	**ZK**	**DK**
1	stellt die beispielhafte Kommunikation zwischen Client und Server tabellarisch dar.	7 (I)			
2	erweitert das Kommunikationsprotokoll um mindestens zwei geeignete Fehlermeldungen.	3 (II)			
Sachlich richtige Lösungsalternative zur Modelllösung: (10) …….. ……..					
	Summe Teilaufgabe a)	10			

Teilaufgabe b)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl (AFB)**	**EK**	**ZK**	**DK**
1	erläutert die ausgewählten Zeilen der Methode im Sachzusammenhang.	6 (II)			
2	vervollständigt die Implementierung.	6 (II)			
Sachlich richtige Lösungsalternative zur Modelllösung: (12) …….. ……..					
	Summe Teilaufgabe b)	12			

Teilaufgabe c)

	Anforderungen	**Lösungsqualität**			
1	erläutert die Funktionalität im Sachkontext.	10 (II)			
Sachlich richtige Lösungsalternative zur Modelllösung: (10) …….. ……..					
	Summe Teilaufgabe c)	10			

[1] EK = Erstkorrektur; ZK = Zweitkorrektur; DK = Drittkorrektur

Teilaufgabe d)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl (AFB)**	**EK**	**ZK**	**DK**
1	implementiert die Methode.	7 (II)			
2	dokumentiert die Implementation durch Kommentare.	3 (I)			
Sachlich richtige Lösungsalternative zur Modelllösung: (10) …….. ……..					
	Summe Teilaufgabe d)	10			

Teilaufgabe e)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl (AFB)**	**EK**	**ZK**	**DK**
1	beurteilt die Aussage.	8 (III)			
Sachlich richtige Lösungsalternative zur Modelllösung: (8) …….. ……..					
	Summe Teilaufgabe e)	8			