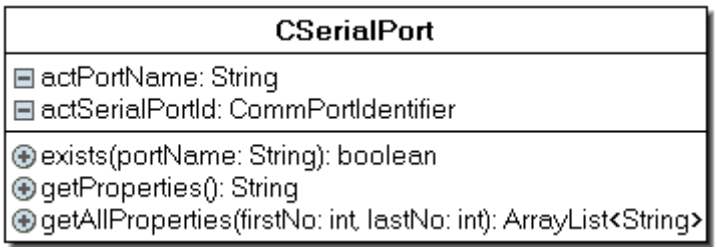
Serielle Kommunikation - Kodierung

(1.) Erstellen Sie nachfolgende Klasse:



Dabei haben die Methoden folgende Funktionen:

exists(): Überprüft, ob eine serielle Schnittstelle existiert

getProperties(): Liefert die Parameter der letzten mit exists() überprüften Schnittstelle als Zeichenkette zurück

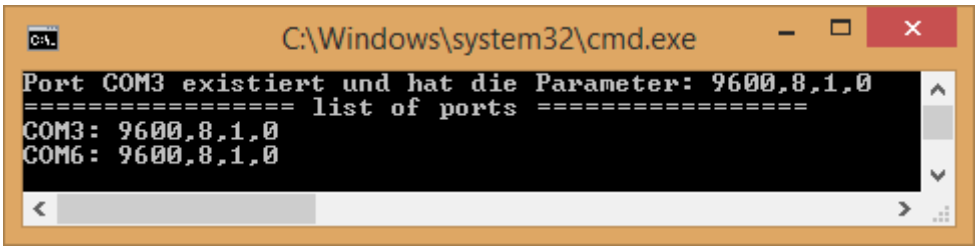
getAllProperties(): Liefert ein ArrayList-Objekt von Stringeinträgen zurück, die alle die Informationen über die jeweilige Schnittstelle enthalten.

Wenn das Hauptprogramm die Form

```
import java.util.*;
public class TestCSerialPort {
  public static void main(String[] args) {
    CSerialPort mySerialPort=new CSerialPort();
    String testComPort="COM3";
    // Test with one port
    if (mySerialPort.exists(testComPort))
       System.out.println("Port "+ testComPort+" existiert und hat die Parameter:
"+mySerialPort.getProperties());
    else
      System.out.println("Port "+ testComPort+" existiert nicht!");
    // Test with multiple ports
    System.out.println("================= list of ports ================= ");
    ArrayList<String> arrAllPorts=mySerialPort.getAllProperties(1,10);
    for (int i=0;i<=arrAllPorts.size()-1;i++) {
      System.out.println(arrAllPorts.get(i));
    } // end of for

  } // end of main
} // end of class TestCSerialPort
```

dann ist die Ausgabe:



(2.) „Konvertieren“ Sie die zur Verfügung gestellten Quelltexte für eine serielle Kommunikation zwischen einem Empfänger und Sender für eine Nullmodemverbindung, so dass nur noch die Klasse Serial(->Abitur) verwendet wird(Entfernen Sie die entsprechende Import Anweisung!)

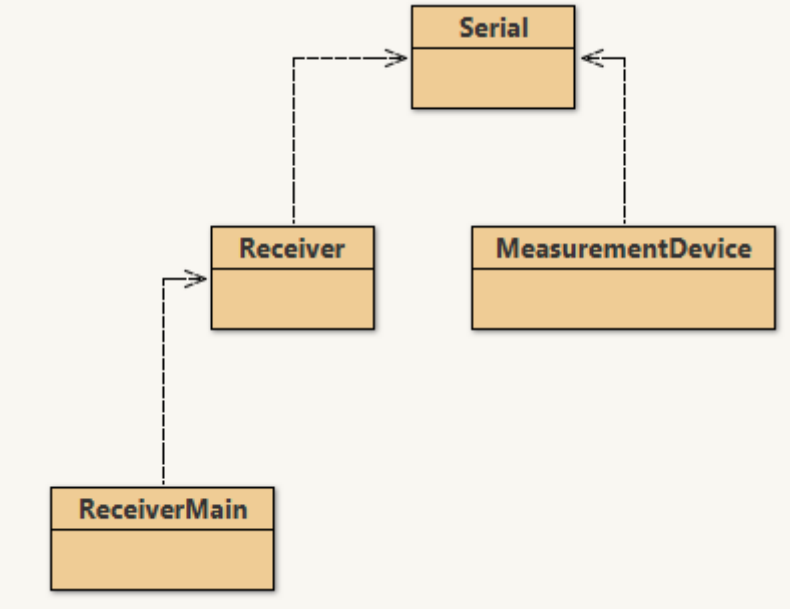
Welche Varianten funktionieren, wenn Sie nicht mit Strings, sondern mit anderen Datentypen arbeiten?

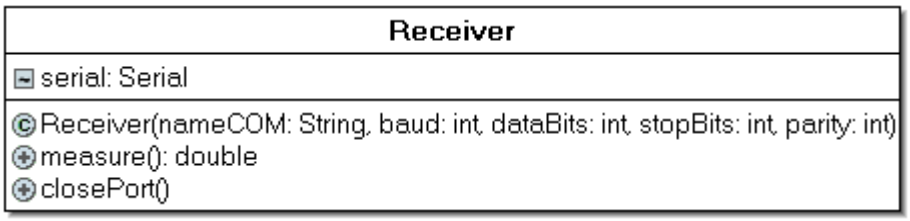
(3.) Kodieren Sie einen Sender, welcher unter Verwendung des ETX/ACK-Protokolls Daten über die RS232-Schnittstelle sendet und erstellen Sie einen passenden Empfänger. Testen Sie anschließend die Datenübertragung für den Fall, dass die Schnittstellenparameter von Sender und Empfänger nicht übereinstimmen!

(4.) Ein Messgerät mit einer RS232-Schnittstelle sendet Messwerte im Byte-Format, 1 Byte entspricht dabei immer genau einem Messwert. Diese Messwerte sollen von einem Rechner empfangen und auf der Konsole ausgegeben werden.

(a.) Erstellen Sie eine Simulation eines Messgerätes, welches auf die Bereitschaft des Empfängers wartet und anschließend Zufallszahlen im Bereich 0..255 sendet!

Kodieren Sie ferner einen Empfänger, welches die oben beschriebenen Messwerte im Polling-Betrieb empfängt und auf der Konsole ausgibt.

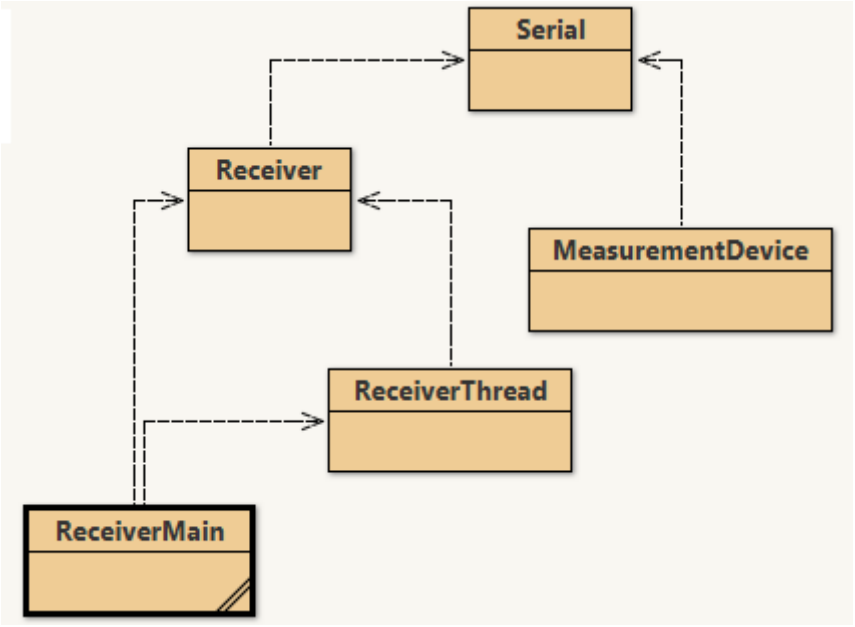


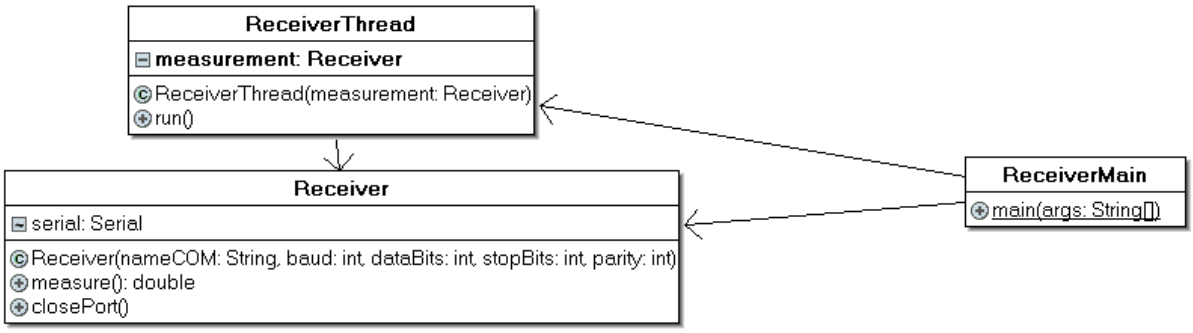


Dabei soll ein Objekt vom Type Receiver mit der Methode measure() seine Sendebereitschaft signalisieren, ein Byte lesen und wieder die Sendebereitschaft deaktivieren.

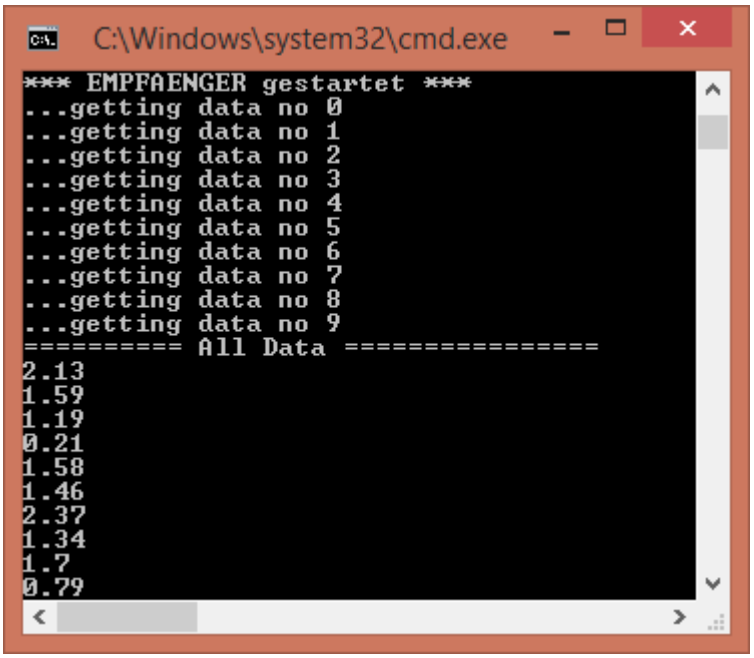
Das Hauptprogramm lässt ein Receiver-Objekt mehrere Male mit measure() Daten einlesen! Gehen Sie davon aus, dass der Sender(Messgerät) ihnen um den Faktor 100 zu große Werte geliefert hat(Da dieser nur Bytes senden kann!)!

(b1.) Das Programm aus (a.) soll verändert werden, das regelmäßige Abfragen der Schnittstelle soll jetzt ein zusätzlicher Thread übernehmen.





(b2.) Erweitern Sie das Programm aus (b1.) so, dass der Thread aus (b1.) die Messwerte in eine geeignete Liste schreibt, die im Hauptprogramm zur Verfügung steht!



(c.) Ein hochmotivierter Azubi hat beiliegenden Quellkode geschrieben, der Aufgabe (b2.) in einer bekannten Software-Handshake-Variante umsetzt!
Beurteilen Sie seine Leitung in funktioneller und syntaktischer Form, also z.B.:

- *Was leistet das Programm?*
- *Was hätte es leiten sollen und warum klappt es nicht?*
- *Welche „Kardinalfehler" der Kodierung sind dem Azubi unterlaufen?*
- *?????*

(5.) Über eine RS232-Schnittstelle ist ein Spannungsmessgerät(Sender) mittels Nullmodemkabel an einem PC (Empfänger) angeschlossen und mit den Parametern: 2400(Baudrate), 1 Stoppbits, Parität 0

Die analogen Messwerte werden im Messgerät unter Verwendung eines 12-Bit-Analog-Digital-Wandlers digitalisiert. Dessen Auflösung beträgt 1 mV pro Bit. Weil über die Schnittstelle nur Datenpakete mit maximal 8 Datenbits (Nutzdaten) übertragen werden können, werden 2 Datenpakete mit je 6 Datenbits gesendet. Das erste Datenpaket enthält die niederwertigeren Bits (20-25).

Beispiel:

Gemessen wurde die Spannung U = 200 mV. Der Analog-Digital-Wandler liefert dann aus 200_{10} die umgerechnete Dualzahl 11001000_2. Zuerst wird die Bitfolge 001000(dezimal 8_{10}) und danach 001100_2(12_{10}) gesendet.

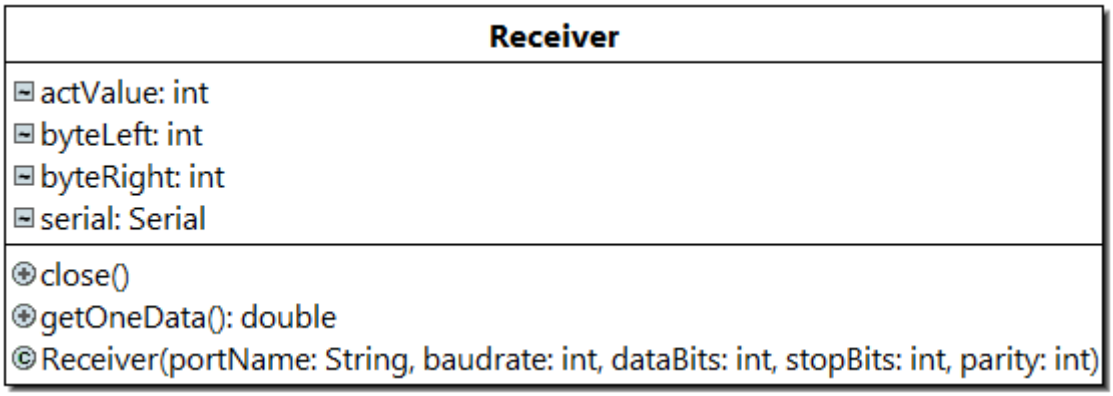
(I.) Schreiben Sie ein einfaches Programm MeasurementDevice, welches unter Verwendung der Klasse Serial das Spannungsmessgerät (Sender) simuliert. Nach dem Start soll dieses Programm endlos laufen und Daten senden.

Erstellen Sie unter Verwendung der Klasse Serial eine Klasse Receiver. In dieser Klasse sollen keine Exceptions abgefangen bzw. behandelt werden, dies soll in der main-Methode einer separaten Startklasse erfolgen. (Die Erstellung der Startklasse erfolgt hier noch nicht.)

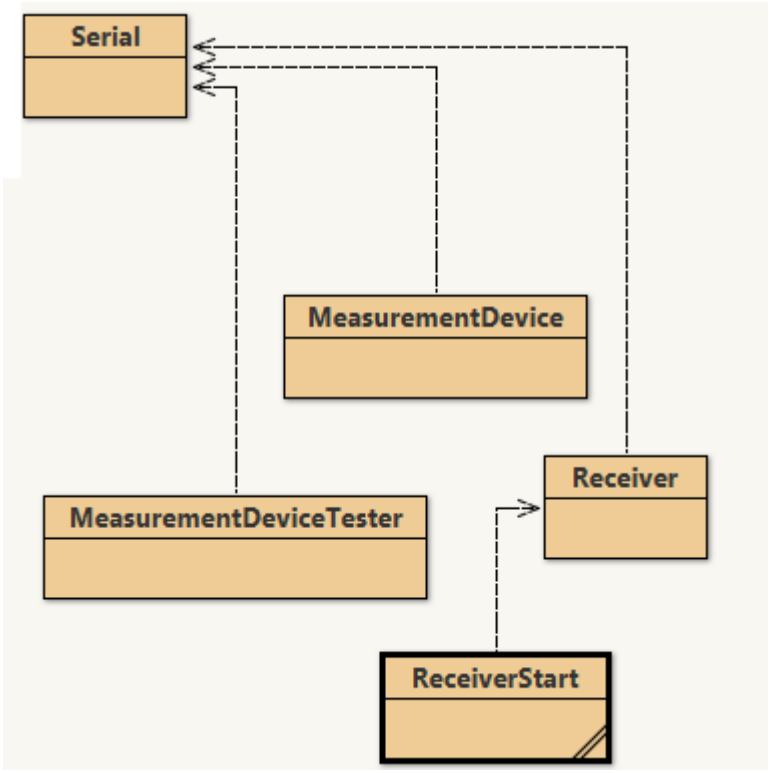
(II.) Die Klasse Receiver muss einen Konstruktor enthalten, welcher die folgenden Parameter der Schnittstelle setzt und danach die Schnittstelle öffnet: Name der Schnittstelle, Baudrate, Anzahl der Datenbits, Anzahl der Stoppbits, Parität

Die Klasse Receiver muss eine Methode getOneData enthalten. Diese Methode startet den Messvorgang, erfasst den auf zwei Datenpakete verteilten Messwert und beendet dann den Messvorgang. Die Methode berechnet aus dem 12-Bit-Messwert den Spannungswert in Volt und gibt diesen als Dezimalzahl zurück.

Weiterhin muss die Klasse Receiver eine Methode close() enthalten, welche die Schnittstelle schließt.

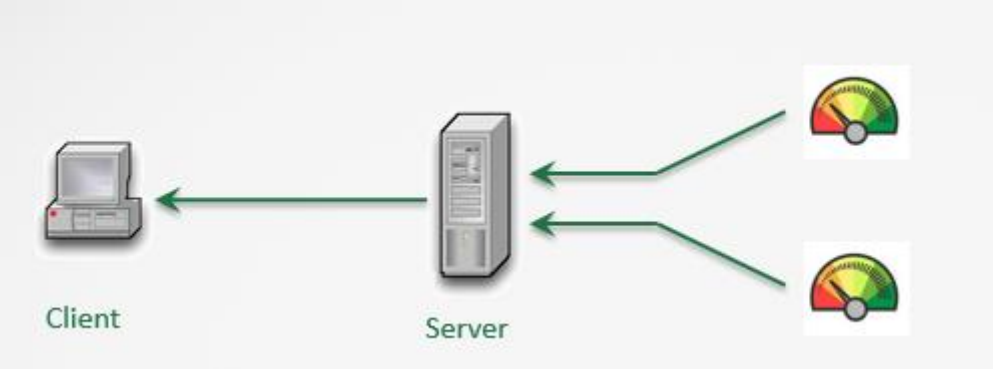


(III.) Erstellen Sie eine Startklasse ReceiverStart, welche in der main-Methode wird zunächst ein Objekt der Klasse Receiver erzeugt. Unter Verwendung der Methode getOneData der Klasse Receiver wird im 5-Sekundentakt ein Messwert erfasst und auf der Konsole ausgegeben. Nach 5 Messungen wird die Schnittstelle geschlossen und das Programm beendet.


Serial
MeasurementDevice
Receiver
MeasurementDeviceTester
ReceiverStart

(6.)Erweiterung von Aufgabe 5:

An zwei seriellen Schnittstellen ist jeweils ein Messgerät angeschlossen. Die Art der Datenübertragung (Protokoll, Schnittstellenparameter) ist bei beiden identisch, lediglich die Schnittstelle hat eine andere Nummer. Weiterhin soll der PC jetzt als Single-Client-Server dienen, welcher die Messwerte über ein Netzwerk an einen Client schickt:



Schreiben Sie eine Klasse MessServer mit folgendem Aufbau bzw. folgender Funktionalität:

Die Klasse besitzt einen Konstruktor, welcher zwei Objekte der Klasse Receiver sowie ein Objekt der Klasse ServerSocket erzeugt. Diesem Konstruktor werden alle für benötigten Parameter übergeben:

Portnummer (für die Netzwerkverbindung)

Nummer der 1. RS232-Schnittstelle

Nummer der 2. RS232-Schnittstelle

alle Schnittstellenparameter (welche für beide Schnittstellen immer identisch sind)

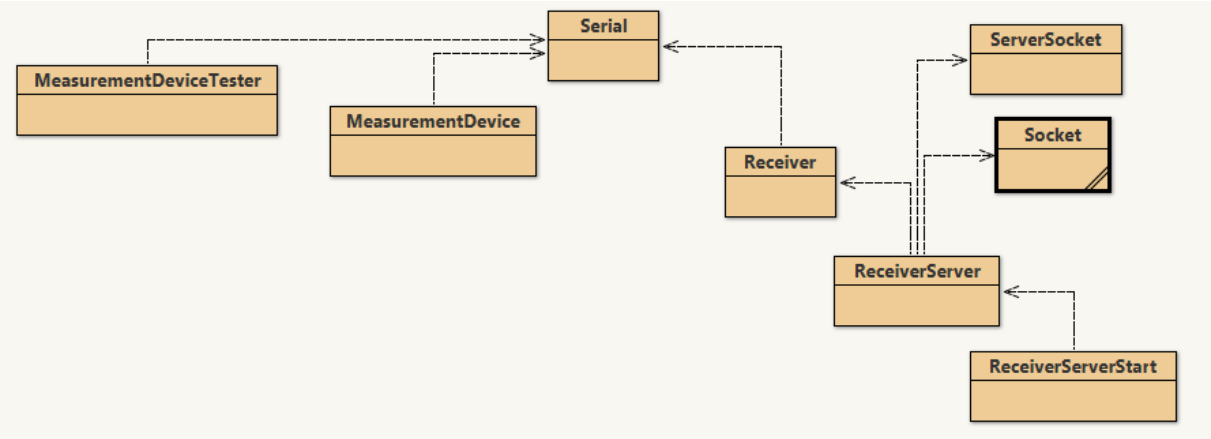
Weiterhin soll die Klasse ReceiveServer eine Methoden start() besitzen. Wird diese aufgerufen, so wartet der Server in einer Endlosschleife auf einen Client. Verbindet sich ein Klient mit dem Server, so erhält er die Nachricht: "Command:" Der Klient antwortet mit einem Befehl. Die folgende Tabelle listet die möglichen Befehle und deren Wirkungen auf:

Befehl	*Wirkung*
m1	*Für das 1. Messgerät wird eine einmalige Messung gestartet. Der ermittelte Messwert wird als String an den Client geschickt. Anschließend wartet der Server auf den nächsten Befehl.*
m2	*siehe m1, allerdings wird jetzt das 2. Messgerät verwendet*
ende	*Der Server beendet die Verbindung und wartet auf den nächsten Client.*

Sendet der Klient einen nicht existierenden Befehl, so passiert nichts und der Server erwartet den nächsten Befehl. Die Klasse hat folgenden Aufbau:

ReceiverServer
clientCmd: String receiver1: Receiver receiver2: Receiver serversocket: ServerSocket socket: Socket
ReceiverServer(socketPortName: int, serialPortName1: String, serialPortName2: String, baudrate: int, dataBits: int, stopBits: int, parity: int) start()

Schreiben Sie eine weitere Klasse ReceiveServerStart, welche die Main-Methode enthält. In der Main-Methode wird ein Objekt der Klasse MessServer erzeugt und anschließend deren Methode start()aufgerufen. Insgesamt ergibt sich folgender Aufbau:



Serielle Kommunikation - Kodierung - Lösungen

Alle in Dateiform