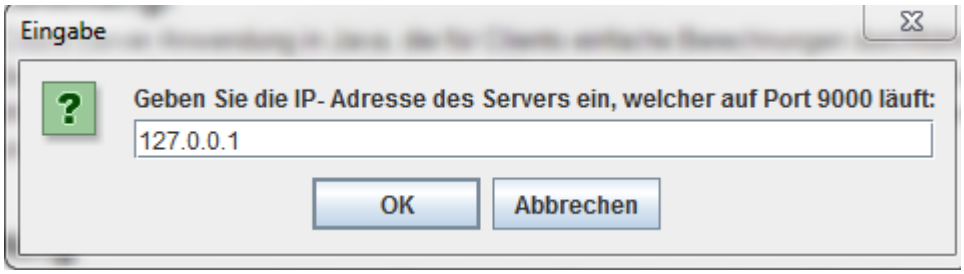
Protokoll Aufgabe 1:

Aufgabenstellung:

Erstelle eine Client/Server Anwendung in Java, die für Clients einfache Berechnungen durchführt. Dabei sollte die Addition, Subtraktion, Multiplikation und Division mit Integer-Zahlen ermöglicht werden. Der Client wählt eine Operation aus und schickt dem Server eine Anfrage.Der Server führt diese Anfrage aus und schickt dem Client das Ergebnis zurück.

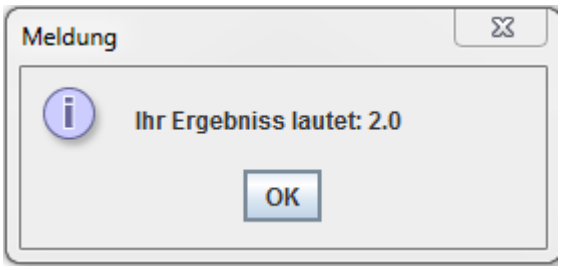
Handhabung

Das Programm besteht aus zwei ausführbaren Dateien, eine Serverdatei und eine Client Datei. Wichtig ist, dass der Server vor dem Client ausgeführt wird, sonst kommt es zu Problemen. Sollte man es geschafft haben, die Dateien in richtiger Reihenfolge auszuführen, muss man im Client als ersten Schritt die Rechnung angeben, im zweiten Schritt die IP-Adresse. Zum testen, wird die IP-Adresse 127.0.0.1 genutzt, da dies der Localhost ist, und beide Anwendungen auf den selben Rechner laufen.



Eingabe der IP-Adresse

Jetzt muss man nur noch auf OK klicken und die Eingaben werden zum Server geschickt, dort berechnet und das Ergebnis wird zurück zum Client geschickt und dort ausgegeben.



Ausgabe vom Ergebnis

Der Ablauf

Nachdem der Server erstellt wurde, bietet er dem Client ein ServerSocket an, mithilfe dessen sich der Client zum Server verbinden kann. Der Server wird über Port 9000 erreichbar sein.

```
ServerSocket listener = new ServerSocket(9000);
Socket socket = listener.accept();
```

Jetzt wartet der Server solange, bis sich ein Client verbindet. sollte sich ein Client verbinden wird dies duchr listener.accept(); akzeptiert und zugelassen.

Im Client wird auch ein Socket erstellt, auch dies benutzt den Port 9000.

```
Socket s = new Socket(serverAddress, 9000);
```

Um etwas zu senden, benötigt man folgende Codezeilen:

```
ObjectOutputStream out = new ObjectOutputStream(s.getOutputStream()); //Ein Object outputstream
out.writeObject(this.getZahl1()+";"+this.getZahl2()+";"+this.getRechenzeichen());//Zahlen werden
out.flush();
```

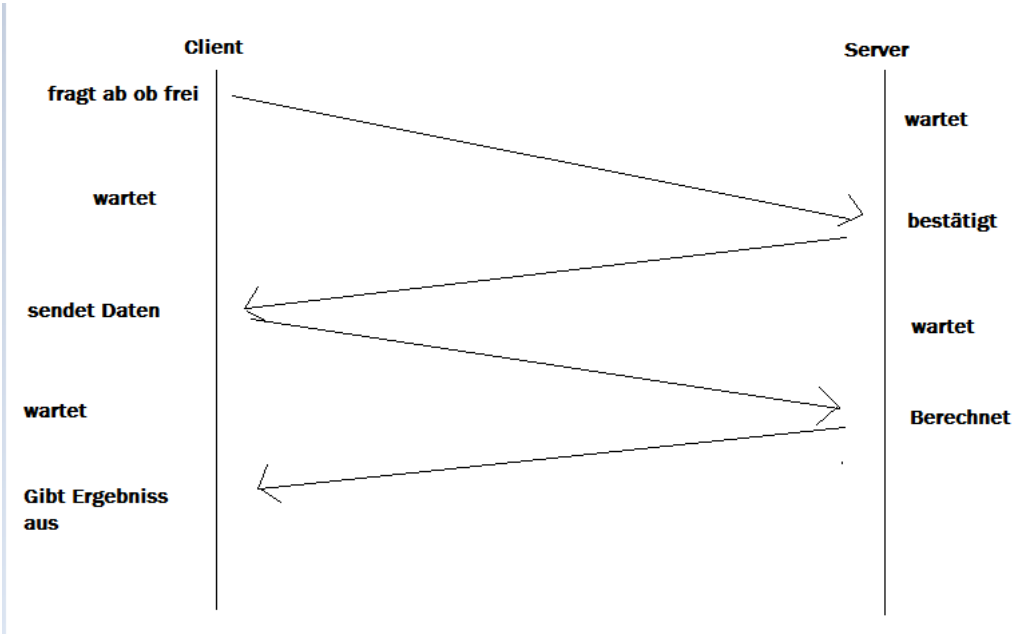
Zum Senden wird am Socket s ein ObjectOutputStream erstellt. Durch writeObject werden die Daten zum Senden vorbereitet, mit flush werden sie anschließend gesendet.

Um Daten zu lesen wird folgender Code benötigt:

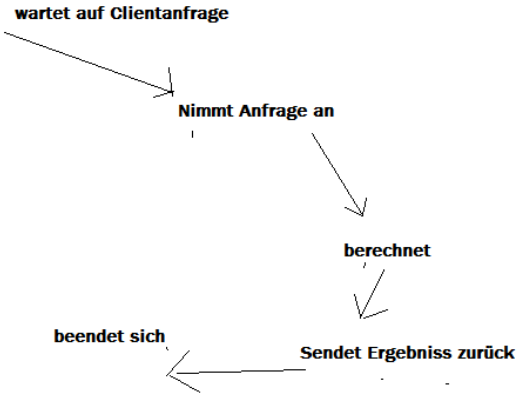
```
BufferedReader input = new BufferedReader(new InputStreamReader(s.getInputStream()));
String answer = input.readLine();
```

Mit BufferReader wird ein Buffer an Socket s erstellt, welcher solange wartet, bis Daten eintreffen. Mit readLine() kann man auf diese Daten zugreifen.
Ich habe bei dieser Übung noch ohne die abstrakte Klasse Message gearbeitet, da ich denke, es ist vorteilhaft eine Übung ohne die abstrakte Klasse als Beispiel zu haben.

Grafiken:



Server:



Client: