Zusatzaufgaben Exceptions, Streams

```
public class Ex {
   public static void main(String[] args) {
      int  x=0; int y = 10;

      try{

         y = y / x;

         System.out.print("Division");

      }catch(Exception ex){

         System.out.print(" Fehler");

      }

  }
}
```

- ☐ Division
- ☐ Division Fehler
- ☐ Fehler
- ☐ Das Programm läuft, macht aber keine Aufgabe
- ☐ das Programm lässt sich nicht übersetzen
- ☐ das Programm wirft einen Laufzeitfehler

Welche Aussagen sind im Zusammenhang mit Dateien in Java wahr?

- ☐ Dateien, die mit Java erstellt werden sind immer nur Sequenzen von Bytes.
- ☐ Jedes Java-Objekt kann mit einem `java.io.ObjectOutputStream` und einem `java.io.FileOutputStream` in einer Datei abgespeichert werden.
- ☐ Wenn man ein Objekt `x` in einer Datei abspeichert und anschließend in eine Variable `y` wieder einliest, so liefert `x == y` immer den Wert `false`.
- ☐ Beim Serialisieren von Objekten werden die Attribute mit dem Modifikator `private` ausgeschlossen
- ☐ Zum Verketten von Streams werden in Java idR. die Konstruktoren der Streams verwendet

Welche Aussagen treffen auf Java Exceptions zu?

- ☐ Nur solche Exceptions die Spezialisierungen der Klasse `RuntimeException` sind müssen gefangen oder im throws-Ausdruck deklariert werden
- ☐ Alle Exceptions haben Methoden, Attribute und Konstruktoren
- ☐ Im throws-Ausdruck einer Methode kann maximal eine Exception deklariert werden.
- ☐ Statische Methoden dürfen keine Exceptions werfen
- ☐ Zu einem try-Block kann es mehrere catch-Blöcke geben
- ☐ Der finally-Block beendet das Programm nach dem Auftreten einer Exception

Welche Aussagen treffen für die Behandlung von Laufzeitfehler zu?

- ☐ Laufzeitfehler sollten immer genau dort behandelt werden, wo sie auftreten.
- ☐ Return Codes von Methoden sind ein geeignetes Mittel, um Informationen zu Laufzeitfehlern an das zuständige Abstraktionsnivieau zu kommunizieren
- ☐ Das Werfen von Exceptions ist ein geeignetes Mittel, um Informationen zu Laufzeitfehlern an das zuständige Abstraktionsnivieau zu kommunizieren
- ☐ Exceptions müssen immer vollständig oder gar nicht behandelt werden. Ein Exceptionhandler darf also selbst keine Exception werfen.
- ☐ Es wird empfohlen unchecked Exceptions zu verwenden, weil damit der Umfang des Quellcodes reduziert werden kann.

Programmieraufgabe

Implementieren Sie ein Client-Server-System, das über Sockets auf dem localhost über Port 8080 kommuniziert. Über den Umgang mit Sockets (v.a. die Klassen Socket und SocketServer) können Sie sich im Java Tutorial belesen. In dieser Aufgabe geht es um den sonnvollen Umgang mit Kommunikationsproblemen durch geeignetes Exception Handling.

Der Server soll als Broadcaster für eine Gruppe von Clients dienen. Clients können sich bei ihn anmelden und ihm dann Nachrichten (als Strings) schicken. Der Server schickt jede Nachricht an alle angemeldeten clients weiter. Durch die Nachricht „Bye“ eines Clients wird seine Verbindung beendet.

Beginnen Sie zunächst mit einem Client, der dann vom server nur seine eigenen Nachrichten, wie ein Echo, zurückbekommt. Implementieren Sie sinnvolles Exception Handling, so dass Clients angemessen reagieren, wenn der Server ausfällt und umgekehrt. Simulieren Sei solche Störungen durch Beenden der Anwendungen mittels Ihres Betriebssystems.