CAMPUS 02 GRAZ
FACHHOCHSCHULE DER WIRTSCHAFT

Fachhochschul-Studiengang „Wirtschaftsinformatik“

Klausur aus der Lehrveranstaltung

PR3 – PROGRAMMIEREN 3

Klausur am 22. Juli
1.Termin

1. SEMESTER

SS

Jahrgang ZAM

Name: _____

Beurteilung: _____

Klausurdauer: 3,5 Stunden
Erlaubte Unterlagen: Alle Unterlagen sind erlaubt.

BEISPIELE MÜSSEN KOMPILIEREN (Keine Syntax Fehler!) – SONST KÖNNEN KEINE PUNKTE DAFÜR GEGEBEN WERDEN.

Im Falle, dass der Import von Java Dateien nicht funktioniert – einfach ein neues Projekt erstellen und dann die einzelnen Klassen kopieren bzw. Struktur nachstellen.

Beispiel 1 – Netzwerk-IO (60 %)

Im Package example1 befindet sich die Klasse:

- ListenToClient

Ziel#1 – Ein Client (40%):

Implementieren Sie einen Server, der Verbindungen nacheinander abarbeitet. (Der Server soll auf Port 1111 hören.)

Details:

Der Client (Telnet/Putty) sendet zeilenweise den String „ping“ an den Server und wartet auf eine Antwort „pong“ vom Server. Der Client kann ebenso den Befehl „stop“ absenden, danach soll der Server die Anwendung beenden und eine entsprechende Nachricht ausgeben. Falls ein Befehl nicht erkannt wird, wird vom Server zum Client eine entsprechende Information ausgegeben.

Bei Annahme einer Verbindung soll eine bestimmte Anzahl an „ping“ / „pong“ Durchläufen festgelegt werden, nach denen der Server dann statt „pong“ z.B. „ende“ an den Client (Telnet/Putty) sendet und der Server die Verbindung beendet.

Tipp: Verwenden Sie den folgenden Code, um die Anzahl der Durchläufe zufällig zu ermitteln:

```
int rounds = new Random().nextInt(10);
```

Beispiel aus Sicht des Clients bei 3 Durchläufen:

- „>“ sind „ping“ die an den Server gesendet werden
- „<“ sind „pong“, die der Client empfängt

```
> ping
< pong
> ping
< pong
> ping
< stop
```

Ziel Nr. 2 – Mehrere Clients (20 %):

• Kopieren Sie die aktuelle Klasse und fügen Sie diese in ein neues Package Namens example3 ein (Dies ist nur relevant, damit Sie Ihren Code nicht zerstören, falls Sie bei Ziel Nr. 2 einen Fehler machen.)

• Passen Sie die kopierte Klasse so an bzw. fügen Sie weitere Klassen hinzu, dass diese Anwendung mehrere Clients unterstützt. (**Multithreading**)

Testen Sie ihre Implementierung mittels der Command Line und dem Befehl „telnet localhost 1111“ oder der Putty Applikation.

Beispiel 2 – File-IO - Schreiben von JAVA-Objekten in eine .dat Binär-Datei (40 %)

Im Package example2 befinden sich folgende Klassen:

- Student
- StudentManager
- Demo

Implementieren Sie in der Klasse StudentManager die Methode **exportStudentsToBinaryFile** und passen Sie die Demo und Student Klasse entsprechend an, sodass alle untenstehenden Ziele erfüllt werden.

Ziel: (20 %)

Die Methode **exportStudentsToBinaryFile** soll in eine .dat Datei Studenten Objekte aus einer selbst erstellten Studenten Liste schreiben. Weiters können Sie auch in anderen Klassen Änderungen vornehmen, wenn diese notwendig sind.

Achtung: Beachten Sie bei der Wahl Ihres „Writers“, dass es sich hier um Binär-Daten handelt und in diese Dateien Java-Objekte geschrieben werden sollen.

Dabei sind weiters noch folgende Punkte zu beachten:

- Um die Funktionalität zu der Methode **exportStudentsToBinaryFile** zu testen, müssen Sie in der Demo Klasse mehrere Studenten Objekte erstellen (mindestens 5) und diese einer Studenten Liste hinzufügen. (5%)
- Passen Sie die Methodensignatur so an, dass der „Pfad der Ziel-Datei“ entgegengenommen werden kann und somit in der Methode verwendet werden kann. (5%)
- Wenn die angegebene Datei bereits existiert, soll eine eigene Exception mit dem Namen “StudentExportException“ geworfen werden. Diese selbsterstellte Exception benötigt eine Informative Aussage was falsch gelaufen ist. (5%)
- Geben Sie wichtige relevante Informationssmeldungen über den Status Ihres Java-Programms auf der Konsole aus. (5%)

Testen Sie Ihre Implementierung mit der Demo Klasse.

Example 1 – Network-IO (60 %)

In the package example1 is one class:

- ListenToClient

Goal#1 – One Client (40%):

Implement a server that processes connections one after the other. (The server should listen on port 1111.)

Details:

The client (Telnet/Putty) sends the string "ping" to the server line by line and waits for a "pong" response from the server. The client can also send the “stop” command, after which the server should terminate the application and output a corresponding message. If a command is not recognized, corresponding information is output from the server to the client.

If a connection is accepted, a certain number of "ping" / "pong" runs should be specified, after which the server then sends, for example, "end" to the client (Telnet/Putty) instead of "pong" and the server terminates the connection.

Tip: Use the following code to randomly determine the number of passes:

```
int rounds = new Random().nextInt(10);
```

Example from the client's perspective with 3 runs

- ">" are "ping" that are sent to the server
- "<" are "pong" that the client receives

```
> ping
< pong
> ping
< pong
> ping
< stop
```

Goal#2 – Multiple Clients (20%):

- Copy and paste the current class into a new package called example3 (This is just relevant, that you don’t destroy your code, in case you make a mistake during Goal#2.)
- Adapt the copied class so, that this application support multiple clients. (**Multithreading**)

Test your implementation using the command line and the command "telnet localhost 1111" or the putty application.

Example 2 – File-IO - writing JAVA objects to a .dat binary file (40%)

The package example2 contains the following classes:

- Student
- StudentManager
- Demo

In the StudentManager class, implement the exportStudentsToBinaryFile method and modify the Demo and Student classes accordingly so that all the objectives below are met.

Goal: (20%)

The method **exportStudentsToBinaryFile** should write student objects from a self-created student list into a .dat file. You can also make changes in other classes if they are necessary.

Caution: When choosing your "writer", remember that this is binary data and that Java objects are to be written into the file.

The following points must also be observed:

- To test the functionality of the **exportStudentsToBinaryFile** method, you must create several student objects (at least 5) in the demo class and add them to a student list. (5%)
- Adjust the method signature so that the "path to the target file" can be accepted and used in the method. (5%)
- If the specified file already exists, a separate exception named "StudentExportException" should be thrown. This self-created exception needs an informational statement of what went wrong. (5%)
- Print important relevant information messages about the status of your Java program on the console. (5%)

Test your implementation with the demo class.