PROG-2

ASSIGNMENT Semester-Projekt

Das Semesterprojekt für PROG-2 bestehen aus fünf bis sechs „**Projektteilen**", auch **SCRUM-SPRINTS** oder kurz **SPRINT** genannt.

- Alle Projekte folgen dem **Prinzip „Inverted Classroom**",
- werden zu definierten Zeitpunkten im ersten Mittwochsblock persönlich abgegeben (samt Doku) und werden benotet.
- Die Teilnehmer sollen anhand der Teilprojekte den Softwareentwicklungsprozess vollständig durchlaufen und so Erfahrungen bei der Umsetzung eines komplexeren Auftrags sammeln.

Gesamtprojektziel ist der Erwerb von vertieften Kenntnissen und Fertigkeiten in den nachfolgend aufgeführten Bereichen und die Beherrschung ihrer praktischen Umsetzung:

- Methodenwissen und Schlüsselkompetenzen
- Fortgeschrittene Programmierung
- Objektorientierte Programmierung
- Programmierung paralleler Threads (concurrent programming)
- Modellierung verteilter Applikationen
- Netzwerkprogrammierung
- Programmierung grafischer Oberflächen
- Datenbankprogrammierung
- Entwicklungsprozess methodisch an SCRUM angelegt
- Dynamics und Pitfalls im Softwareentwicklungsprozess
- Dynamics und Pitfalls im Projektmanagement
- Projektdokumentation: **API-Doku mit Javadoc** Reader/Handout (statt Doku für den Kunden hier ein interner „Projektbericht") im Wordformat.
- **Abnahme** & **Qualitätssicherung** der Software: Autorenteam demonstriert mit QS-Team vor einem „Kunden" („Kunde" wird von *Prof. Illik* dargestellt oder dediziert ernanntem HiWi). Wird die QS nicht durchgeführt, so wird das Projekt als „nicht geliefert" (Note fünf) gewertet. Die Abnahme erfolgt innerhalb von 14 Tagen nach Abgabe. Die Bewertung während der QS ist vorläufig und wird während der Prüfungszeit fixiert.
- **Siehe dazu auch PROMOD2_PraktOrg.pdf**

Aufgabenstellung für **SPRINT 1**.

SCRUM-SPRINT-1: ADRELI_1_CON

Aufgabenbeschreibung

Entwerfen Sie ein Programm für eine einfache Adreßlistenverwaltung mit zeichenorientierter Benutzerschnittstelle[1]. Dem Benutzer sollen folgende Dienste angeboten werden:

1. Neue **Personendaten aufnehmen** und in einer hauptspeicherinternen Datenstruktur puffern. Diese Datenstruktur ist entweder leer oder enthält bereits Daten (aus der Operation 4.). Der **Puffer** ist **zwangsweise** eine **String-Matrix = DoD**.
2. **Datensätze** mit den Inhalten von bereits erfassten Personendaten **anzeigen** (die gepufferten Daten anzeigen).
3. Die erfassten (im Hauptspeicher gepufferten) **Datensätze in einer NoSQL-Datenbasis = Datei[2] im Filesystem speichern** (ggf. werden dabei in der Datei vorhanden Daten überschrieben).
4. Gespeicherte **Datensätze aus einer Datei laden** und in einer hauptspeicherinternen Datenstruktur puffern, sodass einige Operationen auf der Datenbasis „**in-memory**" erfolgen
5. In der **Datei** mit den erfaßten Daten die Records **nummerieren**.
6. Außerdem soll der Benutzer das Programm auf einfache Weise verlassen können.

Das Programm bietet dem Benutzer alle Möglichkeiten mittels eines **Menüs** an; siehe hierfür Bild 1.

Entscheidet sich der Benutzer für das Eingeben von Personendaten, so bietet das Programm dem Benutzer eine „Maske", wie im Bild 2 dargestellt.

[1] Gui-Elemente (AWT, Swing, SWT, ... sind **nicht** erlaubt -> kommt in einem Folgeprojekt!)
[2] Siehe „Bemerkung" zum Thema Datei weiter unten.

Nachdem der Benutzer die Personendaten eingegeben hat, gibt das Programm dem Benutzer die Möglichkeit, die eingegebenen Daten zu korrigieren, oder aber die Richtigkeit der erfaßten Personendaten zu bestätigen. Siehe hierfür Bild 3.

Nach der Korrektur oder der Bestätigung der erfaßten Daten kann der Benutzer weiter Personendaten erfassen, oder die Erfassung beenden (siehe Bild 4). Wird die Erfassung beendet, so geht es mit dem Hauptmenü weiter.

Weitere Aufgabendetails in der Live-Veranstaltung Vorlesung/Kolloquium.

Bemerkungen 1:

Die erfaßten Personendaten werden sequentiell nacheinander in einer Random-Access-Datei abgelegt. Möchte der Benutzer die Datensätze auflisten (Menüpunkt 2 im Hauptmenü), so bekommt er den Dateiinhalt in der physikalischen Reihenfolge angezeigt. Die Anzeige erfolgt satzweise bis zum Dateiende, wobei nach jedem Satz die <RETURN>-Taste zu betätigen ist.

Bemerkungen 2:

- Prüfen Sie alle Feldeingaben mit Hilfe **regulärer Ausdrücke** auf syntaktische Korrektheit.
- Ihre Lösung muss ein eigenes **Exception Handling** beinhalten. (Ohne System-Exception Handling geht ohnehin keine Übersetzung...!)

Bemerkungen 3: Für die Archivierung der Daten in der Datei können zahlreiche Möglichkeit genutzt werden:

- eine Random-Access-Datei nutzen (Klasse **File** und **RandomAccess**). Die Daten **lesbar** (nicht binär) oder binär (nicht lesbar) abgelegen (s.u. Objekte speichern).
- einfaches Textfile mit den **Klassen FileReader** und **FileWriter** bearbeiten.
- Oder die **Klasse Scanner** einsetzen oder sonstige Klasse, wie im Script erwähnt.
- Objekte ein-/ausgeben mit den Klassen **FileOutputStream / ObjectOutputStream** und **FileInputStream / ObjectInputstream**

Beachten Sie: aus der Vielzahl von Möglichkeiten schreibt die DoD das RandomAccessFile mit direkt lesbarem Inhalt vor.

Bringen Sie Spezifikationslücken Mittwochs beim „SCRUM-Meeting" zur Diskussion!

Bild 1: Die **Funtionen** 1 bis 7 von ADRELI_1_CON

```
      ADRELI - Adressverwaltung

Wollen Sie...

      eine neue Person aufnehmen:  > 1
              Records[3] auflisten:  > 2
   Records in eine Datei[4] sichern: > 3
   Records aus einer Datei laden:  > 4
   in-memory Records sortieren[5]  : > 5
Datei[6] löschen Zeilen nummerieren: > 6
          das Programm verlassen:  > 7
```

[3] Die in-memory Records aus dem Puffer werden auf den Bildschirm ausgegeben
[4] aus dem in-memory-Puffer in ein File auf der lokalen Speicher-Platte/Harddisk/SSD
[5] Die in-memory Records werden nach **Namen** sortiert. Angezeigt werden die sortierten Daten mit der Funktion 2 Records auflisten
[6] Datenbasis auf der lokalen Platte.

Bild 2:

```
Geben Sie bitte die Daten ein:

     Name:
  Vorname:
   Anrede:
   Straße:
      PLZ:
      Ort:
  Telefon:
      Fax:
Bemerkung:
```

Bild 3:

```
Geben Sie bitte die Daten ein:

     Name: Lusser
  Vorname: Claudia
   Anrede: Frau
   Straße: Leopoldstraße 88
      PLZ: D-80333
      Ort: München
  Telefon: 089.112
      Fax: 089.110
Bemerkung: Forschungsschwerpunkt: Lean Workflow-
           management


                        Stimmt's (J/N)
```

Bild 4:

```
Geben Sie bitte die Daten ein:

     Name: Lusser
  Vorname: Claudia
   Anrede: Frau
   Straße: Leopoldstraße 88
      PLZ: D-80333
      Ort: München
  Telefon: 089.112
      Fax: 089.110
Bemerkung: Forschungsschwerpunkt: Lean Workflow-
           management


          Noch eine Person aufnehmen? (J/N)
```

Bild 5:

Satzinhalt (1. Satz)

Name:	Lusser
Vorname:	Claudia
Anrede:	Frau
Straße:	Leopoldstraße 88
PLZ:	D-80333
Ort:	München
Telefon:	089.112
Fax:	089.110
Bemerkung:	Forschungsschwerpunkt: Lean Workflow-management

;-)

Generell:

- Ohne dies weiter auszuformulieren: denken Sie an die User-Freundlichkeit, wenn es darum geht, bestimmte Dinge dem User mitzuteilen.
- **KISS** – „keep it simple stupid" versuchen Sie's! Komplexität der Software so einfach wie möglich zu halten, und nur so komplex wie notwendig werden zu lassen. Nicht mehr und nicht weiniger.
- Spezifikation ist Gesetz! (Der Kunde wird Sie sonst nicht bezahlen.In unserem Fall ist die Lösung nicht erreicht, was sich in der Note ausdrückt.)

Verbindliche Reglen (siehe auch Checklist):

- Projekt wird in **(max.) 3er-Teams** bearbeitet
- Projektabnahmetermine: siehe Vorlesung.
- Alle **SPRINT-Ergebniss** werden im **SPRINT-Reviewmeeting** präsentiert. Zum Sprint-Rewiewmeeting sind alle Teammitgieder anwesend.

Abgegeben werden **VOR** der Präsentation in **Papierform die Projekt-Dokumentation (Projektordner)**.

Gliederung der Projekt-Dokumentation: = **DoD**

- *Deckblatt*
- *Autorendatenblatt*
- *Inhaltsverzeichnis*
- *Screen-Shots der Benutzerschnittstelle*
- *Print out der Datenbasis (10 Sätze)*
- *UML-Diagramme*
- *Kommentierter Quell-Code (javadoc)*

Anmerkungen zur Projektdokumentation (gilt für alle PROG2-Projekte):

- der **kommentierte Quell-Code (Achtung:** Quellcode-Zeilen im Printout dürfen nicht länger als 80 Zeichen sein; trennen Sie Methoden durch zwei Leerzeilen, trennen Sie Klassen durch 4 Leerzeilen**) Printout der SourceCodeFiles aus ECLIPSE mit Zeillennummerierung und POINTSIZE 9.**
- **Gernerell** für alle Projekte: orientieren Sie sich weitestgehend an den **Sun Java Code Conventions** in der Fassung vom 12. September 1997 (= aktuelltes Fassung).
- Ein **Printout** vom Dateiinhalt (**Datenbasis**) (mindestens 10 sinnvolle Einträge)
- **UML-Dokumentation**. Notwendige Diagrammarten:
  Class-Diagramm
  Kompositionsstrukturdiagramm (optional)
- Die Diagramme müssen **lesbar** sein (**keine Mini-Bilder (jpgs, etc.**) und werden in einem Word-File (Microsoft Office oder Open Office) entsprechend Kommentiert.
- Digital auf dem USB-Stick: Im Projektverzeichnis befinden sich neben den oben erwähnten Artefakten die mit **javadoc** erzeugte **Dokumentation**. In javadoc arbeiten Sie sich selbständig ein (Sun-/Oracle-Site; Wikipedia; Helge Maus V2B) Die Abgabe erfolgt jeweils im dritten Block Mittwochs.
- Bitte geben Sie die Datenbasis ebenfalls mit ab, damit die QS leichter mit dem Test starten kann. (Funktion 4: Records laden...).

- Zusätzlich zur Dokumentation liefern Sie einen **Screen-Shot** von der Bedienschnittstelle ab.
- Eclipse Workspaces **DÜRFEN NICHT** abgegeben werden. Abgegeben werden, neben der Doku, bis auf weiteres ausschliesslich java-Files und keine class-Files.
- Die Anwendungsdemo muss auf auf einem **PC** in der **PC-Hall C104** laufen**.** Damit wird gewährleistet, dass der oben erwähnte Punkt erfüllt ist. (**Präsentationen** auf dem eigenen Laptop werden nicht akzeptiert. Alle **Präsentationen erfolgen mit Eclipse** – gilt auch für Folgeprojekte)
- In jedem **Quellcode-File** sind die **Autoren** zu vermerken.
- Siehe: **ACCEPTATION CRITERIA im SCRUM_BOARD (gilt für alle ADRELI-SPRINTS)**
- Im **Autorendatenblatt** ist beschrieben**, wer für welche Quellcodeteile zuständig** ist (**Deckblatt** für Ihren Papierstapel).
- Die Verzeichnisstruktur Ihrer Lösung besteht aus den Verzeichnissen **src**" (für den Source-Code), „**Javadoc**"für die mit javadoc generierte Dokumentation im HTML-Format (in Eclipse ->„Project" -> „Generate Javadoc") und „**Doku**" für Dokumentation

Geben Sie der Projektdoku das Aussehen einer Thesis. Auch der formale Aufbau sollte sich an dem Aufbau einer Thesis orientieren.

Abgenommen werden **nur** Projekte mit „kundentauglicher" Projektdoku (Papier und digital).
Erst wenn die Projektdoku „kundentaulich" ist, werden Noten besser als „3" vergeben und ein QS-Review durchgeführt auf Ihren eigenen Laptops und den „Kundenmaschinen" (= PC-Hall-PCs)

WEITERE DETAILS ZUM PROJEKT WERDEN IN DER **VORLESUNG** UND DEM **ÜBUNGSBLOCK** ZU GEGEBENER ZEIT BESPROCHEN.

Noch ein Hinweis an die Spezialisten:

A) Bei den einzelnen Aufgabenstellungen zu den 5 Projekten ist der Lösungsweg teilweise vorgeschrieben und sicherlich gibt es im ein oder anderen Fall sicherlich andere Lösungsmöglichkeiten, im Vergleich zu den geforderten. Auch für Java gilt: „Viele Wege führen nach Rom!". Bitte halten Sie sich an die in der Aufgabenstellung geforderten Lösungsweg – selbst wenn Ihnen eine

(vermeintlich oder echt) bessere Lösung einfällt.
Verzichten Sie auf funktionale Erweiterungen!
B) Es gilt das Prinzip **KISS** („Keep it simple stupid"):
Klassen nur soviele wie unbedingt notwendig und so wenig wie möglich!

Siehe auch: PROG2_PraktOrg_WS1718.pdf

Inverted Classroom – im Plenum: Methode **AKTIVES PLENUM:**

Sprint-Planning-Meeting 11.10.17 *„Aus dem **PROG1-**Semester**projekt** wird **ADRELI_1_CON**"* **– wie sieht dieser Prozess aus?**