Matchplanner

1 Ziel:

Es soll ein GUI entstehen, dass oben eine Menüleiste hat, ein zentrales View in der Mitte und einen tabellarischen auf der rechten Seite.

Menüleiste:

- File
  - New => neuen Spielplan anlegen
  - Open => vorhandenen Spielplan öffnen
  - Save => Änderungen am geöffneten Spielplan speichern
  - SaveAs => geöffneten Spielplan als neue Datei speichern
  - Print/Export to Pdf => Matchplan in PDF exportieren
  - Close => geöffneten Spielplan schließen
  - Exit => Programm beenden
- Extras
  - EditTeams => Mannschaften verändern
  - Edit Days => Spieltage festlegen/verändern

Tabellenview an der rechten Seite:

Auflistung der Spieltage

zentrales View in der Mitte:

Spiele an Spieltag der im rechten Tabellenview ausgewählt wird anzeigen

2 Gruppenarbeit

Diese Aufgabe wird in den zu Semesterbeginn zugeteilten Gruppen bearbeitet. Für jede Gruppe gibt es einen Ordner auf dem Abgabe Server, in dem alle Mitglieder der jeweiligen Gruppe zugriff haben. Standardmäßig ist dort ein Git-Repository angelegt.
Pfad: 193.196.141.209:/var/svn/swp1/gruppe<n>/matchplanner.git

Das Repository enthält nach dem Klonen eine git-Ignore File, die alles ausschließt, außer den Ordnern **‚src‘** und ‚**test‘**. Das ermöglicht jedem die IDE seiner Wahl zu nutzen. Es gilt jedoch acht darauf zu geben, dass die IDE auf **Leerzeichen** Einrückung mit **4 Leerzeichen** eingestellt ist.

Nach dem Klon des Repositories hat man daraus folgenden auch noch kein für die IDE annerkanntes Projekt. Die erreicht man wie folgt:

NetBeans:

- Neues Projekt
- Java Projekt mit existierenden Quellen
- Den Ordner wo das Repo hingeklont wurde auswählen & Name seiner Wahl vergeben
- den Ordner src zu den Quellcodeordnern hinzufügen
- Projekt erstellen

Eclipse:

- Neues Projekt
- Java Projekt
- Default Location abschalten
- Ordner des geklonten Repositories auswählen
- Projekt erstellen

3 Teil 1 – Grundaufbau GUI

Für den ersten Teil, soll das GUI an sich mit Swing Elementen erstellt werden.
Hinter die Menüeinträge, für alles außer dem Beenden, soll eine Platzhalter Aktion ausgelöst werden (z.B. eine MessageBox mit einem Text was dieser Menüeintrag später tun soll).

Die rechte Tabellenansicht der Spieltage kann mit fest einprogrammierten Dummy-Daten erfolgen, die beim Klick auf einen Spieltag einen Platzhalter Text welcher Spieltag ausgewählt wurde im zentralen View in der Mitte anzeigt.

Abgabe Teil 1: ~~Dienstag, 27.11.2018 23:59 Uhr~~ gemeinsam mit Teil 2

4 Teil 2 – GUI mit ersten Funktionen erweitern

1. Im Menü sollen nur die Einträge, die gerade Nutzbar sind, kllckbar sein.

Bei geschlossenem Matchplan müssen folgende Menüeinträge enabled sein:

File:

- New
- Open
- Exit

Edit:

- Keiner

Bei geöffnetem Matchplan müssen folgende Menüeinträge enabled sein:

File:

- Close
- Save
- Save As
- Print
- Exit

Edit:

- Teams
- Dates

2. File -> New

- Neuen Matchplan anlegen durch eine dieser Optionen:
  - 1. Eingaben vom Nutzer fordern:
    - Nutzer nach Anzahl der Teams fragen
      - Mindestwert 4 Teams
      - Team Zahl muss gerade sein
      - Fehlercheck auf Zahlen, wenn nicht parsbar Fehlermeldung und erneute Eingabe Aufforderung
    - Nutzer für jedes Team nach dem Namen fragen (Team ID soll im Frage-Dialog mit angezeigt werden)
    - Spieltage werden automatisch, beginnend mit dem aktuellen Tag, aufwärts gezählt. Die tatsächlichen Spieltage können später im Edit Dialog eingetragen werden.
      => Bei n Teams entstehen n*(n-1) / (n/2) Spieltage
  - 2. Automatisch standardwerte festlegen mit Platzhalter Namen(z.B: „<bitte aendern>“)
- Spiele für jeden Spieltag berechnen und in ein TableView eintragen, das durch den Eintrag in der Navigation an der rechten Seite aufrufbar sind.
- Bei jeder Änderung an Teams oder Dates, soll ein Flag gesetzt werden, dass es nicht gespeicherte Änderungen gibt.

3. File -> Close

- Wenn es nicht gespeicherte Änderungen gibt, Bestätigungsdialog ob die Änderungen gespeichert werden sollen zeigen. (Ja, Nein, Abbrechen)

4. File -> Save

- Wenn noch keine Datei festgelegt ist, muss SaveAs ausgelöst werden, um nach einem Speicherort zu fragen
- Änderungsflag zurücksetzen

5. File -> SaveAs

- Nutzer nach Speicherort für Datei fragen.
  - FileFilter der nur Ordner, .csv, .xls und .xslx Dateien anzeigt
    Wird keine Dateiendung an den Namen geschrieben, wird implizit .csv angenommen
- Es muss noch nicht in eine Datei geschrieben werden, vorerst bleibt das speichern eine NOP

6. File -> Exit

- Close aufrufen. Wenn Close Abgebrochen wurde, darf Programm nicht beendet werden.

Abgabe: ~~Freitag, 30.11.2018 23:59 Uhr~~ Dienstag, 04.12.2018 14 Uhr

Teil 3 – Matchplan aus Datei lesen und schreiben

1 User defined Bibliotheken anlegen

Die Apache POI Bibliothek besteht aus vielen einzelnen jar-Dateien. Man kann diese alle einzeln zum Projekt hinzufügen (bzw. wenn man weiß welche man für den jeweiligen Anwendungsfall benötigt, reicht das entsprechenden Subset).
Auf Dauer könnte man die Excel und PDF Bibliotheken eventuell ja noch einmal brauchen, deshalb würde es sich vermutlich anbieten diese auf dem eigenen PC an einem zentralen Ort zu speichern und in der IDE eine Nutzer definierte Bibliothek anzulegen.

Eclipse
Window -> Preferences -> Java -> Build Path -> User Libraries

- Neue Bibliothek anlegen
- zum Ordner indem die Bibliothek entpackt wurde gehen
- alle Jars die kein javadoc oder src im Namen haben

NetBeans
Tools -> Libraries
Analog zu Eclipse

1. CSV (alle Gruppen)

- Vorgabe CSV lesen können (Entweder Herr Heusch seine, oder meine)
- Eigene Form zum ablegen der Daten in einer Textdatei entwerfen.
- CSV Erzeugung programmieren
- Beispiel Plan anlegen und in CSV schreiben
- geschriebene CSV wieder einlesen und Korrektheit prüfen

2. Excel (Gruppen mit Gruppengröße > 2 Personen)

Analog zu CSV, mit Apache POI eine Excel Arbeitsmappe erzeugen und wieder einlesen.
Dafür gibt es von Apache eine Bibliothek, die man hier findet:
https://poi.apache.org/download.html

Beim speichern muss für die Dateiendung ‚xls' eine HSSFWorkbench und für ‚xlsx' eine XSSFWorkbench erzeugt werden..
Beim lesen, kann man mit der WorkbenchFactory automatisch die richtige auswählen lassen.

Abgabe: mit Teil 4

Teil 4 – Matchplan in PDF exportieren

Der geöffnete Matchplan soll mit dem Menüpunkt Print (darf auch umbenannt werden) in ein PDF exportiert werden.
Wie die Darstellung des Matchplan im PDF stattfindet, ist jeder Gruppe selbst überlassen.
Dafür gibt es die Bibliothek iText:
https://github.com/itext/itext7/releases
https://www.slf4j.org/download.html

Achtung: Die iText Bibliothek benötigt die Bibliothek aus dem zweiten Link. Bei dieser gibt es ein paar Überlagerungen wenn ihr alle Jars hinzufügt. Daher müsst ihr dort nach dem hinzufügen den Fehlerlog in der Konsole im Blick behalten und die doppelten für andere Zwecke wieder aus der angelegten Bibliothek entfernen.

Abgabe: Dienstag, 11.12.2018 13:00 Uhr

Teil 5 – Refactoring und verstecktes Spiel

1 Alle Gruppen

Erstellt für euer Programm ein UML Klassen-Diagramm.
Damit sollt ihr einen Überblick bekommen wie die aktuelle Struktur eures Programms ist und eventuelle Strukturschwächen finden und beheben.
Dazu zählt auch, wenn eine Klassen / Methoden zu übermächtig geworden sind, diese sinnvoll aufzuspalten.

2 Gruppen > 2 Personen

Parallel zum Refactoring des bestehenden Codes der Gruppe soll das „Game of Life“ implementiert werden. Das ganze kann ganz unabhängig vom Bestandsprogramm programmiert werden. Die einzige Verbindung zum Bestandsprogramm besteht darin, dass egal in welcher Situation eine Tastenkombination das Spiel sichtbar/unsichtbar werden lassen soll.
So kann man es immer wenn jemand der es nicht sehen soll, dass nicht gearbeitet wird, zügig verstecken und wieder hervor holen.

Die Spielregeln findet man im Wiki Artikel:
https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life

- Tastendruck für Pausierung
- Tastenkombination für schnelleren/langsameren Intervall
- Mit einem Mausklick in eine Zelle wird diese manuell zum leben erweckt. So kann man neue Muster generieren und sie dann beobachten.

Abgabe: Freitag, 21.12.2018 24:00 Uhr