Individuell rapport

Kandidatnr 732, medlem av inspera-gruppe 4321.

Generell refleksjon

Jeg synes semesteroppgaven var utfordrende. Jeg fikk erfare hvordan både nye og kjente deler fra pensum fungerer sammen i et program.

Da jeg tar dette faget på mitt 3. år, så kjenner jeg ingen som tar dette faget nå. Jeg har derfor måtte skaffe meg en gruppe, kun basert på hvordan de klarte ukesoppgavene. Dette viste seg å slå begge veier da jeg endre opp med et gruppemedlem som viser mye engasjement for oppgaven, og et som ikke har bidratt i det hele tatt.

Jeg brukte Maven for første gang, noe som jeg slet litt med å få til å fungere i starten, men når jeg først fikk konfigurert den riktig så fungerte det fint gjennom resten av prosjektet.

Mine individuelle bidrag

- Lese og skrive kontaktpersoner, billetter og arrangementer til og fra csv fil, med feilhåndtering.
- Oppsett av strategy desingmønster for filhåndtering.
- Egendefinerte exception-klasser.
- Innlesning fra filer i egen tråd, med progressbar. (Se belysning av bidrag lenger ned)
- Endring av data i liste i GUI.
- Alert-bokser.
- Informasjonsvisning av arrangementer i eget vindu.

Jeg har også møtt opp tidsnok til hvert eneste møte, og vært den siste som har dratt. Jeg har også jobbet en del hjemme.

Jeg har også «ryddet» mye for å beholde en god MVC struktur, har jobbet med å flytte ting som ikke direkte behandler GUI elementer i egne klasser, vekk fra controlleren, samt å opprette metoder for kode som brukes flere steder.

Evaluering av andres bridrag

Kandidatnummer 741 sine bidrag

- Skrive kontaktperson, billett, arrangement og lokale klassen.
- GUI elementer, lister med innebygd sorteringsfunskjon fra SceneBuilder.
- Søkefunskjon.
- Inputvalidering på felt der det er naturlig å validere.
- Datepicker, lesing fra filer til comboboxer.
- Filechooser.
- Legge til og slette data til/fra liste.
- Lese og skrive kontaktpersoner, billetter og arrangementer til og fra jobj fil.

Han har møtt opp på alle møter, men har noen ganger kommet sent, og han går mye til og fra (drar på butikken, treffer kompiser ol.) når vi sitter og jobber sammen. Han har jobbet mye hjemme utenom møtene, ihvertfall mot slutten av prosjektet. Grunnet misforståelser tok han ikke del i grupperapporten som skulle leveres underveis i prosjektet.

Kandidatnummer 759 sine bidrag

- Ingen bidrag.

Han har heller ikke dukket opp på flere av møtene vi har hatt, med svært dårlige unnskyldninger i etterkant. Dukket ikke opp som avtalt når vi skulle skrive grupperapport underveis i prosjektet, la derimot til et par setninger og endret font 1 time før grupperapporten skulle leveres.

Belysning av ett av mine bidrag, med kildekode

Nedenfor har presenterer jeg ett av mine bidrag til prosjektet. Jeg viser klassen ReaderThread som utvider Task. Call() metoden returnerer en ObservableList, som senere (se bruk i MainController) castes til det objektet som etterspørres, avhening av inputvariablen «String type» til ReaderThread sin konstruktør.

Jeg synes dette er kode av høy kvalitet, da jeg har skrevet den slik at jeg kun trenger èn klasse som håndterer lesing i egen tråd. Ved å bruke et wildcard objekt som return-verdi i call metoden, så kan jeg bruke denne på alle filer som inneholder en av de tre objektene vi har i programmet: kontaktperson, arrangement og billett, da disse klassene natuligvis arver fra Object.

Jeg sjekker filtypen ved å sammenligne filendingene. Jeg har her, som på alle andre String.equals(«String») kall, valgt å legge til en String.toLowerCase() metode, for å slippe unødvendige feil, hvor for eksempel filendingen er i sore bokstaver.

Videre utfører jeg riktig read-metode basert på klassens inputverdi «type». Denne vil da være en string: enten kontaktperson, arrangement eller billett. Resultatet fra read-metoden legges så inn i en Observable list av, på dette tidspunktet, en ukjent type. Listen returneres etter for-løkken som emulerer en stor fil har kjørt ferdig.

`ReaderThread.java`

```
public class ReaderThread extends Task<ObservableList<?>> {
    private Runnable runMeWhenDone;
    private final String type;
    private final File file;

    public ReaderThread(Runnable doneFunc, String type, File file) {
        this.runMeWhenDone = doneFunc;
        this.type = type;
        this.file = file;
    }

    @Override
    protected ObservableList<?> call() throws Exception {
        ObservableList<?> list = null;
        String fileName = file.getName();
        String fileExtension = fileName.substring(fileName.lastIndexOf("."));
        try {
            if(fileExtension.toLowerCase().equals(".csv")){
                CsvReader csvReader = new CsvReader();
                if(type.toLowerCase().equals("kontaktperson")){
                    list = csvReader.kontaktpersonRead(file.getPath());
                }
                if(type.toLowerCase().equals("arrangement")){
                    list = csvReader.arrangementRead(file.getPath());
                }
                if(type.toLowerCase().equals("billett")){
```

```
                    list = csvReader.billettRead(file.getPath());
                }
            }
            if(fileExtension.toLowerCase().equals(".jobj")){
                JobjReader jobjReader = new JobjReader();
                if(type.toLowerCase().equals("kontaktperson")){
                    list = jobjReader.kontaktpersonRead(file.getPath());
                }
                if(type.toLowerCase().equals("arrangement")){
                    list = jobjReader.arrangementRead(file.getPath());
                }
                if(type.toLowerCase().equals("billett")){
                    list = jobjReader.billettRead(file.getPath());
                }
            }
            for (int p = 0; p < 100; p++) {
                Thread.sleep(10);
                updateProgress(p, 100);
            }
        } catch (InterruptedException e) {
            System.out.println("Tråd avbrutt");
            e.printStackTrace();
        }
        return list;
    }

    @Override
    protected void succeeded() {
        if( runMeWhenDone != null ) {
            runMeWhenDone.run();
        }
    }
}
```

I controlleren binder jeg de tre ulike progressBarene til readTask, som er av typen ReaderThread. Metoden disableButtons tar inn enn Boolean, og deaktiver knappene dersom den tar inn «true». På denne måten sikrer jeg at brukeren ikke får manipulert dataene i listene, eller lagre til fil mens tråden kjører.

Etter tråden har lest fra fil (readTask.setOnSucceeded), og returnert en liste av typen <?> caster jeg wildcardet til det objektet som hører til typen som sendes inn i readFromFileThread metoden. Jeg kaller tableView.setItems på listen som jeg caster til det tilhørende objektet. Videre gir jeg en melding til brukeren om hvlken fil som ble lest fra.

Jeg aktiverer her knappene igjen, da tråden har kjørt ferdig.

Dersom det fanges noen Exceptions, eller tråden av en eller annen grunn feiler, aktiveres knappene igjen (for tråden har feilet, og kjører derfor ikke lenger). Jeg oppretter et objekt av typen AlertBoxes som har en metode som oppretter en AlertBox. Boksen sin melding settes til meldingen til avviket som ble fanget i Reader-klassen, for eksempel «F ant ingen billetter i filen, prøv en annen fil.».

MainController.java

```
private ExecutorService service = Executors.newSingleThreadExecutor();

public void readFromFileThread(String type, File file) {
    disableButtons(true);

    ReaderThread readTask = new ReaderThread(null, type, file);
    arrangementProgress.progressProperty().bind(readTask.progressProperty());
    kontaktpersonProgress.progressProperty().bind(readTask.progressProperty());
    billettProgress.progressProperty().bind(readTask.progressProperty());

    readTask.setOnSucceeded(event -> {
        ObservableList<?> list = readTask.getValue();
        if(type.toLowerCase().equals("kontaktperson")){
            kontaktpersonTableView.setItems((ObservableList<Kontaktperson>) list);
            kontaktpersonMelding.setText("Lesing av " + file.getName() + " fullført!");
        }
        if(type.toLowerCase().equals("arrangement")){
            arrangementTableView.setItems((ObservableList<Arrangement>) list);
            arrangementMelding.setText("Lesing av " + file.getName() + " fullført!");
        }
        if(type.toLowerCase().equals("billett")){
            billettTableView.setItems((ObservableList<Billett>) list);
            billettMelding.setText("Lesing av " + file.getName() + " fullført!");
        }
        disableButtons(false);
    });

    readTask.setOnFailed(event -> {
        disableButtons(false);
        AlertBoxes alert = new AlertBoxes();
        alert.info("Feil", "Kunne ikke lese fra fil", readTask.getException().getMessage());
    });

    service.execute(readTask);
}
```

```
//Bruk
readFromFileThread(«Kontaktperson», file);
//Kalles for eksempel i TableView.setItems(*metodekall*).
```

Ellers har jeg vært konsekvent med hvordan jeg setter opp metodene (plassering av krøllparanteser ol.) og jeg bruker tab der det er natulig.