6. laboratorijska vježba (11 bodova)

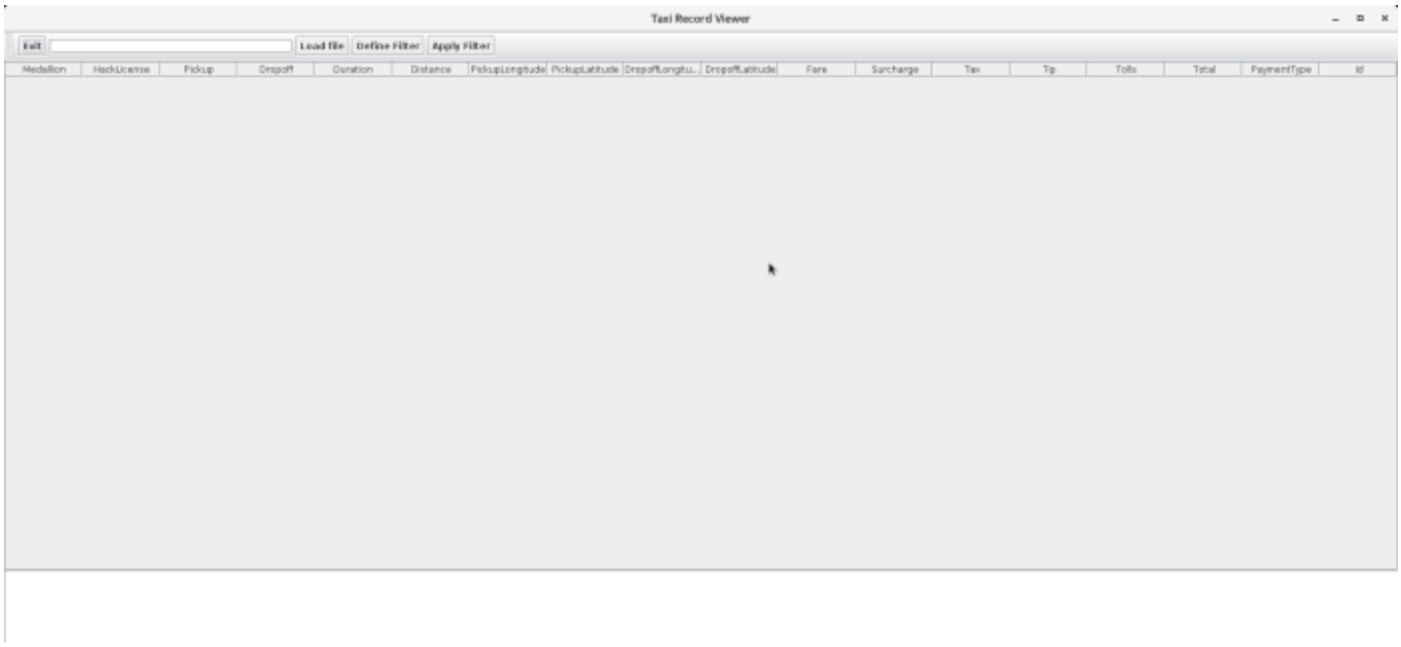
Važna napomena: *u svim zadacima potrebno je napisati Javadoc komentare za svaki razred te generirati dokumentaciju. Svi nazivi razreda, metoda i varijabli moraju biti na engleskom. Sav napisani programski kod mora biti napisan u skladu s konvencijama imenovanja varijabli, metoda i razreda (varijable i metode: malo početno slovo,* camel-case*; razredi i sučelja: veliko početno slovo,* camel-case*; konstante: uobičajeno sve veliko i razdvajanje podvlakom) te ostalim pozitivnim praksama (uključivo i korektno uvlačenje redaka; smisleno razdvajanje više različitih semantički grupiranih redaka praznim redcima, pravilnim razmještajem otvorene i zatvorene vitičaste zagrade i slično). Za više informacija pogledajte* ***http://www.oracle.com/technetwork/java/codeconventions-150003.pdf***

Zadatak 1 – Filtriranje zapisa

Potrebno je napraviti aplikaciju za prikaz i filtriranje podataka o vožnjama taksija koji su bili korišteni na natjecanju ACM DEBS 2015 Grand Challenge. Ovi podaci su veličine 18,6 MB i mogu skinuti sa sljedeće poveznice: https://github.com/MarioKusek/FER-OOP/blob/master/Lab6/data_small.csv. Podaci u ovoj datoteci su u tekstualnom (CSV) obliku gdje svaka linija predstavlja zapis o jednoj taksi vožnji. Linija se sastoji od vrijednosti (medallion, hack_license, pickup_datetime, dropoff_datetime …) odvojenih zarezom kao što je objašnjeno na web stranici natjecanja ACM DEBS 2015 Grand Challenge.

Podaci o vožnjama taksija trebaju biti prikazani u tablici. Kako Swingova tablica (`JTable`) nije obrađena na predavanjima, za prikaz podataka ćemo koristiti unaprijed pripremljenu klasu `GenericTablePanel` koja omogućava tablični prikaz liste objekata nekog tipa (pod uvjetom da objekti tog tipa imaju barem jedan javni *getter*) u tabličnom obliku. Ova klasa i primjer njenog korištenja se nalaze na sljedećoj poveznici: https://github.com/MarioKusek/FER-OOP/tree/master/Lab6/GenericTable.

Aplikacija treba izgledati kako je prikazano na sljedećoj slici (**pri čemu redoslijed stupaca nije bitan**).

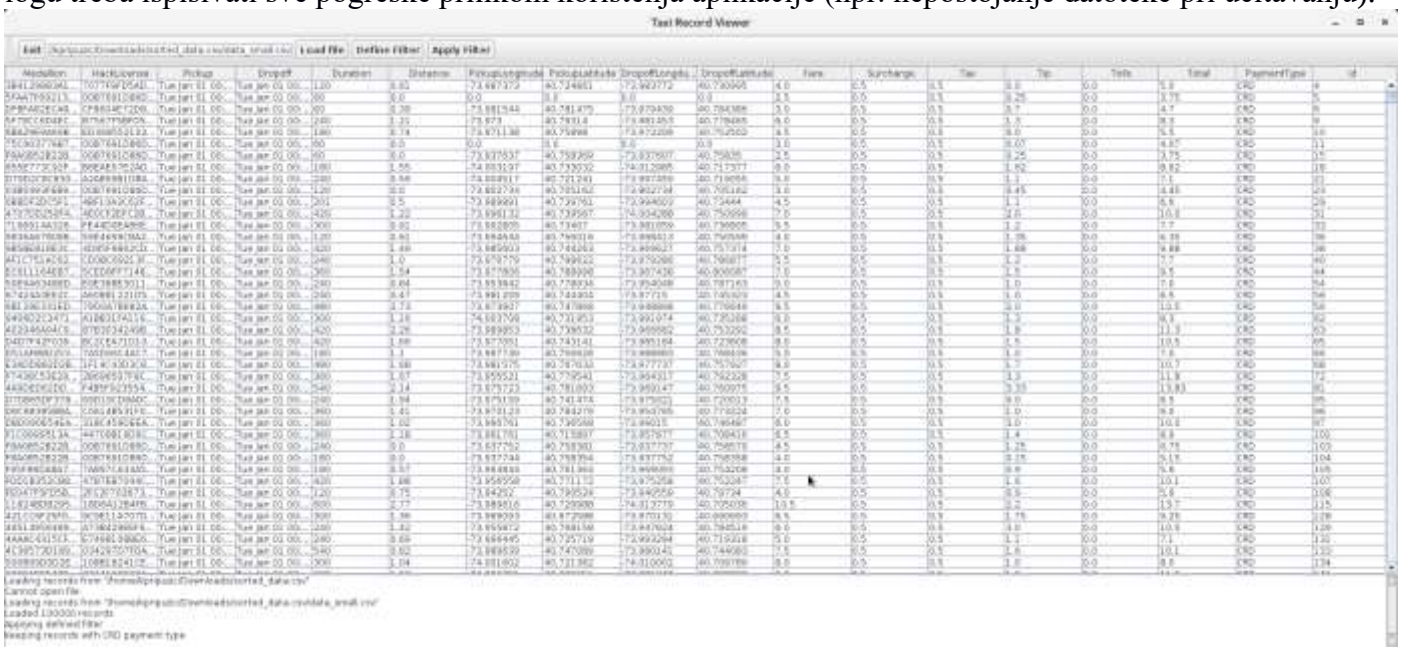


Na vrhu se nalazi traka s alatima (`JToolBar`) koja ima 4 gumba i jedno tekstualno polje. Gumb `Exit` je namijenjen izlasku iz aplikacije. Gumb `Load` je namijenjen učitavanju datoteke s podacima o taksi vožnjama. Pri tome put do datoteke treba unijeti u tekstualno polje s lijeve strane ovog gumba. Učitavanje datoteke sa zapisima je potrebno ostvariti korištenjem statičke metode `Stream<String> lines(Path path)`

klase `Files`, tj. korištenjem kolekcijskih tokova na način da je iz toka `String`-ova potrebno stvoriti tok `TaxiRecord`-a kojeg treba terminirati u listu pa prikazati u tablici. Klikom na gumb `Define Filter` treba se otvoriti novi prozor unutar kojega se mogu definirati uvjeti na osnovu kojih će se filtrirati zapisi koji su trenutno prikazani u tablici. Ovaj prozor se ne može zatvoriti, već se samo može sakriti i to ili klikom na gumb `x` u njegovom gornjem desnom uglu ili ponovnim klikom na gumb `Define Filter`. Klikom na gumb `Apply Filter` je potrebno filtrirati zapise koji su trenutno prikazani u tablici primjenom definiranog filtra, nakon čega je potrebno u tablici prikazati samo one zapise koji su zadovoljili definirane uvjete. Filtriranje je potrebno ostvariti korištenjem kolekcijskih tokova na način da je iz liste podataka u tablici (oni se dohvaćaju pozivom metode `getRecords` kako je pokazano u primjeru korištenja klase `GenericTablePanel`) potrebno stvoriti tok te ga filtrirati na osnovu definiranih uvjeta. Prozor u kojem se definiraju uvjeti za filtriranje treba izgledati kako je prikazano na sljedećoj slici.

Mogu se definirati 4 uvjeta filtriranja: preskakanje prvih n zapisa, izostavljanje svih osim prvih n zapisa, izostavljanje zapisa čiji tip plaćanja ne odgovara odabranom i izostavljanje zapisa čija je vrijednost atributa `distance` manja ili „veća od ili jednaka“ definiranoj. Za svaki uvjet postoji kvačica kojem se on može odabrati ili ne. Prozor s uvjetima za filtriranje je potrebno ostvariti u zasebnoj klasi koja nasljeđuje klasu `JPanel` ili `JFrame` i ima jednu metodu kojom se mogu dohvatiti odabrani uvjeti filtriranja. Uvjete filtriranja je potrebno ostvariti u zasebnoj klasi koja se sastoji od niza *gettera* i *settera*, poput klase `UserData` koja je pokazana na predavanjima.

U sredini aplikacije se nalazi tablica u kojoj se prikazuju zapisi, a na dnu se nalazi tekstualno područje (JTextArea) unutar kojeg se ispisuje log (sa porukama za korisnika) kao što je prikazano na sljedećoj slici. U logu treba ispisivati sve pogreške prilikom korištenja aplikacije (npr. nepostojanje datoteke pri učitavanju).



Zbog složenosti posla je učitavanje i filtriranje zapisa potrebno ostvariti iz pozadinskih dretvi, a ne korištenjem grafičke dretve. Pri tome je pri učitavanju potrebno onemogućiti gumb za primjenu filtra i obratno. Osim toga je svakom zapisu prilikom učitavanja potrebno pridodati redni broj (krajnji lijevi stupac na slici iznad). Ovaj broj se ne nalazi u tekstualnoj datoteci iz koje se učitavaju zapisi, već je pridodan prilikom učitavanja zapisa.

Primijetite da klikom na ime stupca klasa `GenericTablePanel` omogućava sortiranje zapisa po tom stupcu.

Zadatak 2 – Igra života

U ovom zadatku treba napraviti igru života (Conway's Game of Life - https://en.wikipedia.org/wiki/Conway's_Game_of_Life). Igra se sastoji od ploče koja je preko rubova spojena lijevo-desno i gore-dolje. Svako polje može imati dva stanja živo i mrtvo. Jedna iteracija se izvršava tako da se za svako polje ispita da li će u sljedećoj iteraciji biti živo ili mrtvo. Za izračunavanje se koriste sljedeća pravila:

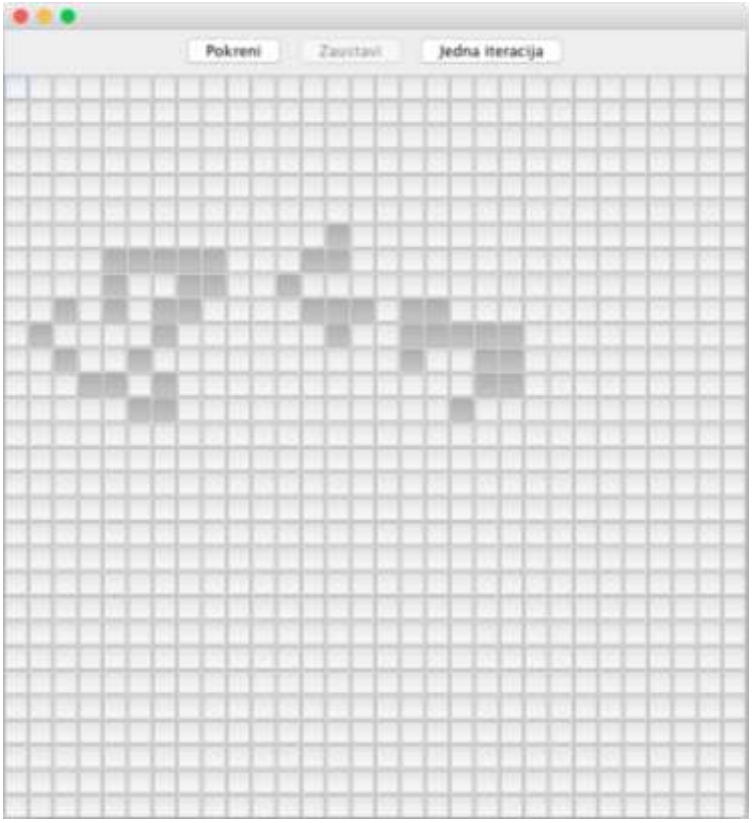
- Ako živo polje ima manje od dva živa susjedna polja umire (u sljedećoj je iteraciji mrtvo).
- Ako živo polje ima dva ili tri susjedna polja u polje ostaje živo.
- Ako živo polje ima više od tri živa susjeda umire.
- Mrtvo polje koje ima točno tri živa susjeda postaje živo (rađa se).

Svako polje ima 8 susjeda, a to su polja oko tog polja horizontalno, vertikalno i dijagonalno.

Igru života potrebno napraviti u klasi `Board`. Klasa ima konstruktor koji prima veličinu polja (npr. 30, 30). Metoda `boolean isCellAlive(int x, int y)` vraća istinu ako je polje s koordinatama živo `true`, a inače `false`. Metode `getWidth()` i `getHeight()` vraćaju veličinu polja. Metoda `setCell(int x, int y, boolean alive)` postavlja stanje jednog polja. Metoda `int countAliveNeighbors(int x, int y)` računa žive susjede. Metoda `playOneIteration()` prolazi kroz sva polja, izračunava nove vrijednosti i nakon njenog izvršavanja ploča ima vrijednosti nakon jedne iteracije. Metoda `addListener(BoardListener listener)` dodaje slušača u listu, a `removeListener(BoardListener listener)` ga miče. Nakon što se izvrši jedna iteracija sve slušače se obavještava o promjeni na ploči tako se pozove metoda iz sučalja `BoardListener`. Sučelje izgleda ovako:

```
public interface BoardListener {
    void boardChanged(Board board);
}
```

Nakon što je napravljena logika potrebno je napraviti grafičko sučelje kao na slici.



Grafičko sučelje je napravljeno u klasi `BoardFrame`. `BoardFrame` inicijalizira ploču od 30x30 polja. Svako polje je prikazano pomoću `JToggleButton`-a. Kada se klikne na pojedini gumb onda se u ploči treba podesiti vrijednost polja ovisno o tome je li gumb ostao pritisnut ili je otpušten.

Na vrhu su tri gumba:

- «Pokreni» - pokreće dretvu koja izračunava novu iteraciju svakih 500ms i osvježava polja u grafičkom sučelju. Izračunavanje se radi u novoj dretvi, a osvježavanje se mora izvršavati u grafičkoj (*event dispatch*) dretvi. Kada se klikne na pokreni onda se taj gumb onemogućava, ali omogući klik na gumb zaustavi.
- «Zaustavi» - zaustavlja izvršavanje dretve koja se pokreće pritiskom na pokreni. Nakon toga se onemogućava gumb zaustavi, ali se omogući gumb pokreni.
- «Jedna iteracija» - ovaj gumb je omogućen samo kada nije pokrenuta dretva koja izračunava. Ako je omogućen i kliknuto je na njega onda se pokreće izvršavanje jedne iteracije. Izvršavanje se može izvršiti u grafičkoj dretvi.