Objektno-orijentisano programiranje, ispit JAN2
Matematički fakultet, školska godina 2024/2025

Uputstvo: Na Desktop-u napraviti direktorijum pod imenom `oop_i_InicijaliAsistenta_Ime_Prezime_Alas` (npr. `oop_i_IA_Marko_Markovic_mi23101`). Pokrenuti *Intellij Idea* i u napravljenom direktorijumu napraviti projekat sa istim nazivom. Zadatke sačuvati redom u okviru paketa `zadatak1`, `zadatak2` i `zadatak3`.

Kôd **ne sme** imati sintaksnih grešaka niti izbacivanje `NullPointerException`-a.
U tekstu je dat opis klasa, njihovih atributa i metoda. **Dozvoljeno** je dodati nove atribute, klase, metode, enume, interfejse u slučaju da olakšavaju implementaciju i/ili poboljšavaju kvalitet koda i slično.

Da bi se uspešno položio ispit potrebno je osvojiti **barem 30 poena**.

1. Implementirati klase za podršku aerodromu.

   - Definisati nabrojivi tip `TipLeta` čije su vrednosti: `DOMACI` i `MEDJUNARODNI`. Dodati statički metod `String oznaka(TipLeta t)` koji vraća `"D"` ili `"M"`.
   - Definisati apstraktnu klasu `Putnik` sa poljima `ime` (String) i `int` (godine) i `String` (destinacija).
   - Definisati klasu `PutnikBezPasosa` koja nasleđuje `Putnik`. Konstruktor postavlja ime, godine i destinaciju. Metod `toString()` vraća oblik:
     `Putnik Marko, godine: 23, destinacija: Nis`
   - Definisati klasu `Pasos` sa poljima: `datumIsteka (LocalDate)` i `ime (String)`. Implementirati metod `boolean validan(LocalDate datum) koja provera da li je pasos validan na prosleđeni datum`.
   - Definisati klasu `LaznoPredstavljanjeIzuzetak` koja nasleđuje `RuntimeException`.
   - Definisati klasu `PutnikSaPasosom` koja nasleđuje `Putnik` i ima polje `Pasos pasos`. Ukoliko konstruktor klase primi `pasos` na tudje ime, ispaljuje `LaznoPredstavljanjeIzuzetak`. Metod `toString()` vraća oblik:
     `Putnik Milos, godine: 30, destinacija: Pariz, ima pasos`
   - Definisati klasu `Let` sa poljima: `destinacija` (String), `tip` (TipLeta), `putnici` (List<Putnik>). Konstruktor klase prima destinaciju i tip leta.
   - U klasi `Let` implementirati metodu `dodajPutnika(Putnik p)` koja dodaje putnika samo ako se destinacija poklapa i ako za međunarodni let putnik ima važeći pasoš u momentu poziva funkcije (Koristi metod `LocalDate.now()`). Metod `toString()` ispisuje destinaciju, tip ("D" ili "M") i sve putnike. Obratiti pažnju da spisak putnika može biti dugačak.
   - U klasi `Aerodrom` kreirati let za Nis (domaci) i let za Pariz (medjunarodni). Obavezno kreirati `putnikaSaPasosem` kome je istekao pasos, kreirati `putnikaSaPasosem` cije ime u pasosu ne odgovara njegovom imenu, nekoliko validnih putnika i dodati ih na letove. Ispisati oba leta na standardni izlaz. U slučaju putnika koji se lažno predstavlja, ispisati na standardni izlaz za greške njegovo stvarno ime i nastaviti sa programom.

2. `[20p]` U korenom direktorijumu projekta napraviti datoteku `meni.csv`, čiji je jedan red oblika:

   `naziv, cena`

   (npr. `"Pizza, 600"`).

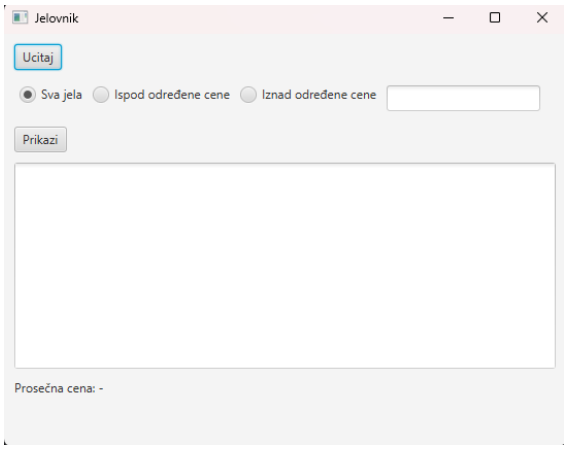
   Napraviti JavaFX aplikaciju sa sledećim komponentama:

   - Dugme `Ucitaj` – učitava jela iz fajla i pamti u listu.
   - Dugme `Prikazi` – prikazuje jela prema odabranim filterima.
   - Tekstualno polje – u koje se može uneti broj (koristi se kod filtera "ispod određene cene" ili "iznad određene cene").
   - RadioButton filteri (omogućiti da je maksimalna jedan moguće uključiti):
     - `Sva jela`
     - `Jela ispod određene cene`
     - `Jela iznad određene cene`
   - Labela - prikazuje prosečnu cenu svih jela učitanih iz fajla.

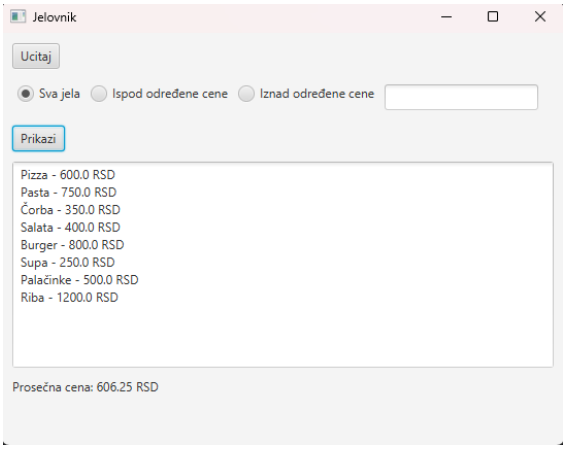
   Klikom na `Prikazi`:

- Ako je označeno "Sva jela", prikazuju se sva jela u formatu: `"Naziv - cena RSD"`.
- Ako je označeno "Ispod određene cene", prikazuju se samo jela čija je cena manja od unete vrednosti.
- Ako je označeno "Iznad određene cene", prikazuju se samo jela čija je cena veća od unete vrednosti.

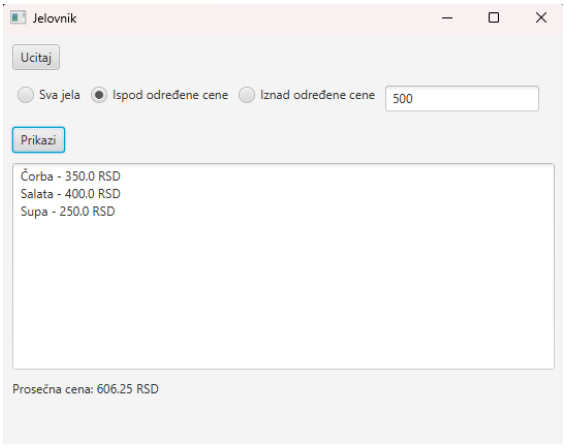
Na dnu aplikacije u labeli se uvek prikazuje **prosečna cena svih jela** učitanih iz datoteke. Obezbediti da se učitavanje iz datoteke izvrši tačno jednom.



(a) Inicijalni prozor



(b) Prikaz u slučaju svih jela



(c) Prikaz jela manje cene od zadate

Slika 1: Jelovnik

3. [25p] Implementirati generičku klasu `Hrana<T>` sa atributima `sadrzaj` tipa `T` i `kolicina` tipa `double`.
   Implementirati metod `compareTo` koji poredi objekte po `sadrzaj`. Metod `toString` vraća oblik:
   `"sadrzaj - kolicina kg"`.
   Definisati klasu `Magacin` sa atributom `Map<Hrana<String>, String>` gde se čuva par (`hrana, vrstaŽivotinje`).
   Implementirati:

   - konstruktor bez argumenata (inicijalizuje `TreeMap`),
   - konstruktor sa komparatorom (npr. poređenje po količini),
   - metoda `dodaj` koja dodaje novu hranu i životinju,
   - metoda `print` koja ispisuje sve parove.

   U klasi `Main` testirati:

   - kreirati jedan objekat tipa `Magacin` sa poređenjem po sadržaju i dodati nekoliko primeraka hrane,
   - ispisati ga metodom `print`,
   - kreirati drugi objekat tipa `Magacin` sa poređenjem po količini i dodati nekoliko različitih primeraka hrane,
   - ispisati ga metodom `print`.