Objektno-orijentisano programiranje, Ispit JUN2, Grupa 1
Matematički fakultet
Školska godina 2018/2019

Napomena: Na Desktop-u napraviti direktorijum pod imenom `oop_Asistent_Grupa_Prezime_Ime_Indeks` (npr. `oop_NM_1_Peric_Pera_mi12082`). Pokrenuti *Intellij Idea* i u napravljenom direktorijumu napraviti projekat sa istim nazivom. U napravljenom projektu, napraviti paket sa istim nazivom.
Kod ne sme imati sintaksnih grešaka niti izbacivanje `NullPointerException`-a.
Vreme za rad: **2.5 sata**
Inicijalini asistenata: `Biljana - BS`, `Nemanja - NM`, `Anja - AB`, `Ivan - IR`, `Rastko - RD`

1. Napraviti apstraktnu klasu `Proces` koju karakterišu atributi: identifikator procesa (`int`) (u daljem tekstu `pid`), naziv (`String`) i memorijsko zauzeće u MB (`int`). Implementirati konstruktor koji prima vrednosti za sve atribute i potrebne `get` metode.

2. Napraviti klasu `AktivanProces` koja nasleđuje klasu `Proces` i predstavlja proces koji se aktivno izvršava u prednjem planu. Aktivan proces se dodatno karakteriše poljem `iskoriscenostCPU` (tipa `int`) koje označava procenat iskorišćenosti CPU - dakle uzima vrednosti iz intervala $[0 - 100]$ (nije potrebno validirati). Implementirati metod `toString()` koji vraća nisku kao u test primeru.

   Primer za metod `toString` (pid, naziv, memorijsko zauzeće, iskorišćenost CPU)

   ```
   [2123] IntelliJ IDEA | 300 MB / 21% CPU
   ```

3. Napraviti klasu `PozadinskiProces` koja nasleđuje klasu `Proces` i predstavlja proces koji se izvršava u pozadini. Klasa se karakteriše oznakom koja služi za raspoznavanje sistemskih procesa (`boolean`). Dodati neophodne `get` metode kao i `toString` metod.

   Primer za metod `toString` (pid, naziv, memorijsko zauzeće, indikator sistemskog procesa - ako nije sistemski proces ne pisati tekst `(System)`)

   ```
   [1003] Windows Defender | 10 MB (System)
   [3021] explorer.exe | 5 MB (System)
   [2222] background_process.exe | 5 MB
   ```

4. Napraviti klasu `MenadzerProcesa` koja omogućava pregled i kontrolu procesa. Klasa sadrži atribut `procesi` (`Map<Integer, Proces>`) koje predstavlja mapu koja preslikava pid programskog jezika u odgovarajući objekat klase `Proces` sa tim pid-om.

   Implementirati metod `toString()` tako da vraća spisak procesa kao što je prikazano na slici 1 (za svaki proces u mapi po jedna linija koja predstavlja rezultat poziva `toString` metoda za taj proces).

   Implementirati metod `boolean ucitajProcese(String putanja)` koji iz datoteke koja se nalazi na prosleđenoj putanji učitava podatke i smešta ih u mapu `procesi`. Ukoliko je učitavanje uspešno, vraća `true`, a inače vraća `false`.

   Datoteka sadrži prvo indikator da li je proces aktivan ili pozadinski (A ako je aktivan, P ako je pozadinski), zatim pid, naziv procesa, memorijsko zauzeće. Aktivni procesi imaju CPU iskorišćenost u nastavku dok pozadinski procesi imaju indikator da li je proces sistemski ili ne (tekst `sys` na kraju linije).

   Primer datoteke:

   ```
   A , 1022 , Google Chrome , 4020 , 62
   P , 1212 , Windows Defender, 10 , sys
   A , 3122 , Steam , 121 , 5
   A , 1190 , VLC , 912 , 10
   P , 912 , explorer.exe , 32 , sys
   P , 1001 , Avast Antivirus , 123
   P , 2100 , Windows update , 92 , sys
   P , 3123 , Steam update , 9
   ```

5. U klasu `MenadzerProcesa` dodati metode:

   - `Proces memorijskiNajzahtevniji(int gornjaGranica)` koji pronalazi proces koji ima najveće memorijsko zauzeće manje od argumenta `gornjaGranica`. Ukoliko je `gornjaGranica` postavljena na 0, razmatrati sve procese. Ukoliko je više rezultata, vratiti prvi pronađeni.
   - `List<Proces> sistemskiProcesi()` koji vraća listu svih sistemskih pozadinskih procesa. Ukoliko ne postoji nijedan takav proces, vratiti `null`.
   - `int ukupnaIskoriscenostCPU()` koji vraća ukupnu iskorišćenost CPU za sve aktivne procese.
   - `boolean zaustaviProces(int pid)` koji iz mape uklanja proces sa datim pid-om. Ukoliko proces sa datim pid-om ne postoji, vratiti `false` u protivnom vratiti `true`.

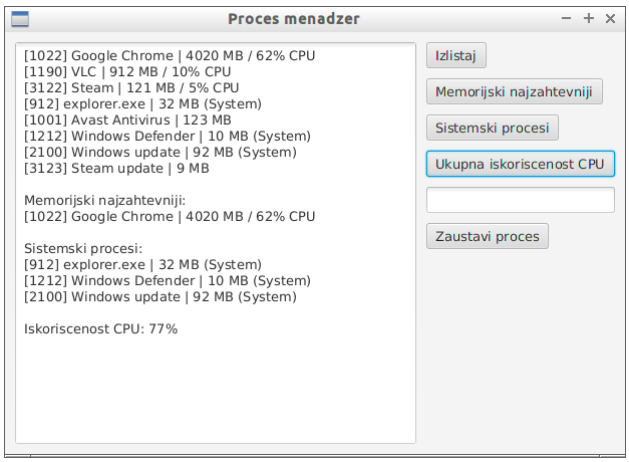
   Pretpostaviti da će se u mapi naći **barem jedan** proces. Ne proveravati da li se procenti iskorišcenosti CPU sumiraju na broj manji od 100.

6. Napraviti klasu `MenadzerGUI` koja sadrži `main` metod i u njoj kreirati grafički korisnički interfejs kao što je prikazano na slikama u daljem tekstu. Pri pokretanju programa, instancirati objekat klase `MenadzerProcesa`, pozvati metod `ucitajProcese()` i u `TextArea` element ispisati poruku da li je učitavanje bilo uspešno ili ne.

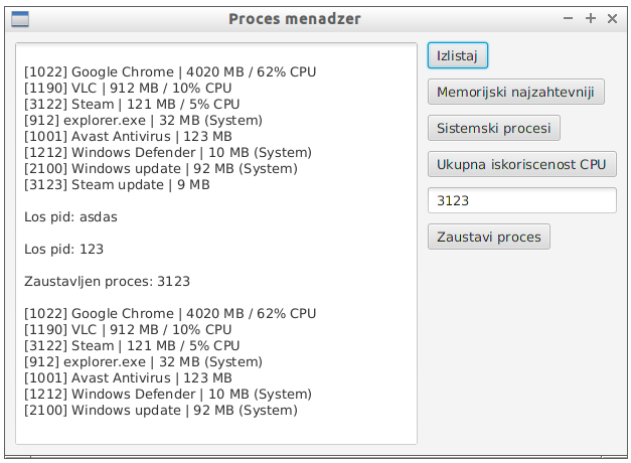
   Za implementaciju događaja je dozvoljeno koristiti prethodno implementirane metode u klasi `MenadzerProcesa`.

   Na klik dugmeta:

   - **Izlistaj** u `TextArea` element se upisuju podaci o svim procesima. Prvo se listaju aktivni procesi, zatim pozadinski. Unutar svake grupe procesa sortirati ih opadajuće po memorijskom zauzeću.
   - **Memoriski najzahtevniji** u `TextArea` element ispisati memorijski najzahtevniji proces. Vrednost za gornju granicu se unosi se učitava iz `TextField` elementa. Ako je `TextField` prazan ili sadrži nevalidnu vrednost, smatrati da je gornja granica 0.
   - **Sistemski procesi** u `TextArea` element ispisati sve sistemske pozadinske procese.
   - **Ukupna iskoriscenost CPU** u `TextArea` element ispisati ukupan procenat iskorišćenosti CPU za sve procese.
   - **Zaustavi proces** koji zaustavlja proces sa pid-om unesenim u `TextField` element. Zaustavljanje procesa znači uklanjanje istog iz spiska svih procesa u okviru trenutnog menadžera procesa. U `TextArea` element ispisati indikator uspeha zaustavljanja. Ukoliko je sadržaj `TextField` elementa nevalidan, ispisati odgovarajuću poruku.



Slika 1: Prikazivanje procesa



Slika 2: Stopiranje procesa