Simulacija prioritetnog zauzimanja radne memorije

Napisati program koji simulira konkurentno zauzimanje radne memorije od strane procesa različitih prioriteta. Radna memorija je predstavljena klasom **radna_memorija** i podeljena je na konstantan broj okvira određen konstantom **UKUPNO_OKVIRA** datom u postavci zadatka.

Procesi

Procesi su predstavljeni nitima. Da bi se izvršio, svaki proces mora da zauzme određeni broj okvira u radnoj memoriji u koji smešta svoje instrukcije i druge podatke potrebne za izvršavanje. Postoje dva tipa procesa: **USER odnosno korisnički** i **SYSTEM odnosno sistemski proces**i. Prilikom zauzimanja okvira u radnoj memoriji, prednost imaju sistemski procesi. Proces ne može sam da zauzme potrebnu memoriju, nego može samo da inicira zauzimanje memorije **slanjem zahteva** opslužiocu (više o opslužiocu u nastavku). Nakon što pošalje zahtev, proces mora da sačeka dok opslužilac ne zauzme zahtevani broj okvira i može da nastavi svoju aktivnost tek kada opslužilac to signalizira. U postavci zadatka je data metoda **napravi_zahtev** koja kao parametre prima broj okvira koji su potrebni procesi, identifikator i tip procesa. **Implementacija metode napravi_zahtev predstavlja deo projektnog zadatka!**

Nakon što nit opslužilac zauzme traženi broj okvira za proces, proces prelazi u stanje spreman i može da nastavi svoje izvršavanje kada mu procesor bude ponovo dodeljen. Izvršavanje procesa se simulira uspavljivanjem procesa na onoliko sekundi koliko ima strana. **Obratite pažnju na to da je simulacija izvršavanje procesa već implementirana!** Nakon završetka izvršavanja, proces mora da oslobodi okvire koje mu je dodelio opslužilac. Nakon oslobađanja svojih okvira, proces signalizira da postoje slobodni okviri u radnoj memoriji i to tako što prvo signalizira sistemskim procesima, ukoliko ih ima, odnosno korisničkim procesima ukoliko nema sistemskih. U postavci zadatka je data metoda **oslobodi** koja kao parametar prima identifikator procesa koji želi da oslobodi svoje okvire. **Implementacija metode oslobodi predstavlja deo projektnog zadatka!**

Radi lakšeg praćenja izvršavanja procesa, svaki proces nakon što dobije okvire za izvršavanje ispisuje okvire koje je dobio. Zbog toga je potrebno pozvati metodu **ispisi_okvire** iz Dijagnostika klase kada proces zauzme okvire.

Opslužilac

Proces opslužilac je predstavljen pomoću demon niti. Opslužilac je **zadužen za zauzimanje okvira radne memorije** za procese koji se izvršavaju. Procesi komuniciraju sa opslužiocem šaljući mu zahteve. Ukoliko nema zahteva, opslužilac čeka dok se oni ne pojave, u suprotnom opslužuje zahteve. **Zahtevi se opslužuju prema prioritetu** i to tako što se prvo opslužuju zahtevi sistemskih procesa, ako ih ima, pa tek onda korisničkih procesa ukoliko nema zahteva koje su uputili sistemski procesi. Ukoliko postoji više zahteva sistemskih procesa, **opsluživanje se vrši po FIFO principu**, a ukoliko postoji više zahteva korisničkih procesa, **opsluživanje se vrši po principu najmanjeg broja stranica, odnosno proces kome treba najmanje stranica, prvi će biti učitan.** Opslužilac može da zauzme onoliko okvira koliko je proces zahtevao samo ukoliko u radnoj memoriji postoji

najmanje toliko slobodnih okvira, u suprotnom mora da sačeka da neki od procesa koji je prethodno dobio okvire oslobodi svoje okvire. Ukoliko postoji dovoljan broj slobodnih okvira da bi zahtev bio ispunjen, opslužilac zauzima okvire, pri čemu **zauzeti okviri ne moraju biti uzastopni**. Nakon što zauzme potreban broj okvira, opslužilac signalizira procesu koji je uputio zahtev da može da nastavi svoju aktivnost. U postavci zadatka je data metoda **opsluzi** koju izvršava nit opslužilac i **njena implementacija predstavlja deo projektnog zadatka.**

Napomene:

1. **Ne pokušavati rešavati ceo zadatak odjednom! Prvo krenuti od skice problema na papiru. Papir ćete dobiti od asistenata. Dobro i sa razumevanjem pročitati tekst zadatka. Realizovati deo po deo zadatka.**
2. **Funkcija main sadrži test scenario koji treba da bude od pomoći u proveri validnosti**
   algoritma. Test scenario se može menjati po želji ali je napravljen tako da se lako može istestirati da li vam program radi ispravno. Pre predaje zadatka trebalo bi vratiti test scenario u originalni oblik (tj. funkciju main), ako su u međuvremenu rađene neke promene kako bi zadaci bili efikasno pregledani.
3. **Komentari su obavezni.**