Elektrotehnički fakultet
Banja Luka

Programski jezici 2
- 08.09.2021. –

1. (30) Napisati program za obradu podataka izmišljene kompanije. Osnovni dokumenti sa kojima kompanija radi dobijaju se od nepoznatih spoljašnjih sistema u formatima koji se nalaze na Moodle stranici predmeta. Svi dokumenti se smiještaju u folder *PJ2_exam_data*. Program čita sadržaj foldera sa dokumentima i uvozi podatke u sistem na dva načina: ručno (pri čemu se šalje naziv fajla) i automatski. Automatski način praćenja podrazumijeva da program prati u realnom vremenu da li je u folder dodan neki fajl, a obavlja se u pojedinačnim nitima tako da se obezbijedi konkurentan uvoz podataka (jedan folder jedna nit). Čim se detektuje dodavanje fajla počinje se sa uvozom podataka. Za uvoz podataka potrebno je kreirati odgovarajuće adapter klase, apstraktnu i konkretne za svaki format podataka i pozivati ih na optimalan način da se obezbijedi proširivost formata podataka i izbjegne ponavljanje istog koda. Parsiranje, odnosno obrada podataka obavlja se pozivom metode *import* iz odgovarajućeg konkretnog adaptera. Po potrebi koristiti generičke mehanizme za implementaciju. Adapteri su zaduženi da izvrše parsiranje sadržaja fajlova i kreiraju odgovarajuće objekte. Objekti koji se mogu napraviti su instance klasa: Proizvod (sifra, naziv, količina, cijena), Proizvođač (naziv) i Vrsta (naziv). Na početku svakog naziva fajla nalazi se broj proizvoda u fajlu. Program pravi objekte na osnovu podataka u fajlu, a ako ukupan broj instanciranih objekata nije isti kao broj proizvoda koji se nalazi u nazivu fajla ili ukoliko nedostaje bilo koji od navedenih atributa objekata, objekat se ne dodaje u sistem, ali se proces uvoza podataka ne prekida. Svi objekti se smiještaju u jednu dijeljenu kolekciju.

Tokom rada programa korisnik može bez ometanja procesa obrade podataka unijeti naredbe sa tastature:

- IMPORT fajl - ručni uvoz fajla,
- AUTO_IMPORT folder - automatski uvoz podataka,
- STATUS - vraća spisak foldera koji se prate i broj stavki u kolekciji,
- SAVE - serijalizuje kolekciju u Home direktorij korisnika računara u fajl podaci.ser.

2. (20) Studentska služba čuva sljedeće podatke o studentima: ime, prezime, broj indeksa, godina rođenja, trenutnu godina studija i prosječnu ocjenu. Pripremiti izvještaj za koji se sastoji od sljedećeg:

- Ukupan broj studenata po godini studija,
- Tri najčešća prezimena studenata,
- Najčešće ime studenta,
- Prikaz svih studenata grupisanih po godinama starosti na sljedeći način: 18-23, 24-29, 30-35...
- Prikaz svih studenata grupisanih po prezimenu,
- Prikaz najboljeg studenta za svaku godinu studija,
- Prikaz svih različitih godina rođenja u formatu godina1#godina2#... korištenjem *reduce* metode i
- Prikaz svih studenata sortiranih po prezimenu u opadajućem redoslijedu.

Prije testiranja svih implementiranih metoda, generisati automatski minimalno 80 studenata. Pri generisanju početno slovo imena i prezimena birati *random*, a ostatak može da bude “ime/prezime”. Godinu rođenja generisati tako da godine starosti studenata budu u opsegu od 18 do 35. Prosječna ocjena je slučajan broj u opsegu od 6.0 do 10.0.

3. (20) Napisati klasu *UlancaniStek* koja predstavlja implementaciju ulančanog steka koji u sebi čuva generičke elemente. Ova klasa ima tri atributa: *vrijednost, tos* i *referenceNext*, gdje *vrijednost* predstavlja generički element, *tos* predstavlja referencu na vrh steka, a *referenceNext* predstavlja referencu na naredni element

steka. Klasa posjeduje dvije metode *push* i *pop*, za smiještanje i skidanje elemenata sa steka, respektivno. U slučaju da je stek prazan, prilikom poziva metode *pop,* potrebno je baciti i obraditi korisnički-implementiran *StackEmptyException*. Neophodno je obezbijediti sinhronizovan pristup steku. *AddThread* i *RemoveThread* su klase koje su niti i zadužene su za stavljanje i skidanje elemenata sa steka, respektivno. Ove klase imaju po 2 atributa: stek (tipa *UlancaniStek*) i number (tipa int), gdje *stek* predstavlja stek sa kojim niti vrše intekrakciju, a *number* broj elemenata koje *AddThread* i *RemoveThread* niti trebaju dodati na stek, odnosno obrisati sa steka, respektivno. Napisati klasu *StackMain* u čijoj će main metodi biti instanciran stek, te niti *AddThread* i *RemoveThread* koje će vršiti interakciju sa stekom. Elementi koji se smiještaju u stek su cjelobrojne vrijednosti i generišu se proizvoljno u opsegu od 0 do 100. Simulacija punjenja/pražnjenja steka traje trideset sekundi, pri čemu se punjenje i pražnjenje izvršava u vremenskom intervalu od 0.2 do 0.3 sekunde za svaku od niti odvojeno. Broj iteracija koje *AddThread* i *RemoveThread* imaju je *200 (number).* U slučaju da po završetku simulacije stek nije prazan, potrebno je stek isprazniti u glavnom programu i ispisati sadržaj na standardni izlaz. Sve događaje potrebno je ispisati na standardnom izlazu.

Vrijeme za rad: 180 minuta