4.Četvrta laboratorijska vježba

4.1. TEMA VJEŽBE

Svrha laboratorijske vježbe je korištenje zbirki iz skupina listi, setova i mapa te zamjena svih polja u aplikaciji sa zbirkama. U sklopu vježbe je također sve skupine konstanti potrebno zamijeniti enumeracijama. Također se u vježbi implementira i algoritam prema kojim se sortiraju objekti unutar zbirki.

4.2. ZADATAK ZA PRIPREMU

Proširiti rješenje treće laboratorijske vježbe na način da se kopira rješenje te preimenuje u naziv koji sadrži indeks „4", umjesto „3". Osim same mape s projektom, potrebno je promijeniti i naziv projekta unutar IntelliJ-a korištenjem opcije „Refactor->Rename". Program je potrebno proširiti na sljedeći način:

1. Na svim mjestima u aplikaciji gdje se koriste polja objekata klase „Item" potrebno je polje zamijeniti listom, osim u klasama „Factory" i „Store" gdje je polje objekata klase „Item" potrebno zamijeniti zbirkom tipa „Set".

2. Nadjačati „equals" i „hashCode" metode za klase, „NamedEntity", „Address", „Category", „Discount", „Factory", „Item", „Store" i svim ostalim klasama koje nasljeđuju klasu „Item" i implementiraju sučelja „Edible" i „Technical" korištenjem opcije IntelliJ-a opisane na sljedećem linku: „https://www.jetbrains.com/help/idea/generate-equals-and-hashcode-wizard.html".

3. Kreirati novi paket „hr.java.production.enum".

4. U paket „hr.java.production.enum" dodati enumeraciju koja će definirati nekoliko gradova koji se moraju moći odabrati kod definiranja adresa. Svaki grad treba imati definiran svoj naziv i poštanski broj. Enumeracija mora imati svoj konstruktor (koji prima String s kojim se definiraju naziv i poštanski broj grada) te „getter" metode za te vrijednosti.

5. Doraditi klasu „Address" na način da umjesto varijabli „city" i „postalCode" sadrže vrijednost enumeracije definirane u četvrtom koraku te prema tome prilagoditi i ostatak klase (promjenom konstruktora te „getter" i „setter" metoda).

6. Kreirati novi paket „hr.java.production.sort".

7. Unutar paketa „hr.java.production.sort" kreirati klasu „ProductionSorter" koja implementira sučelje „Comparator" te unutar metoda „compare" implementira algoritam koji sortira podatke objekata klase „Item" po cijeni. Prilagoditi klasu da se može sortirati uzlazno i silazno po cijeni (nije dozvoljeno imati dvije zasebne klase za te dvije namjene).

8. Metodu „main" u glavnoj klasi proširiti zbirkom iz tipa mapa koja će za ključ imati objekt klase „Category", a za vrijednost listu objekata klase „Item". Na kraju programa ispisati najskuplji i najjeftiniji artikl po svakoj od kategorija korištenjem te mape. Prilikom sortiranja je potrebno koristiti „sorter" klasu definiranu u prošlom zadatku.

9. Na kraju programa ispisati podatke o najskupljem i najjeftinijem artiklu iz klasa koje nasljeđuju sučelja „Edible" i „Techical".

NAPOMENE:

1. Osim implementacija vježbe prema uputama, dozvoljeno je uvoditi i promjene ako su opravdane i ne narušavaju koncepte objektno-orijentiranog programiranja.