PROGRAMSKI JEZICI 1

Pismeni ispit (22.02.2021.)

1. **(15)** Definisati interfejs **`IPrintable`** kojim se predstavlja mogućnost ispisa objekata implementirajuće klase na izlazni tok, bez potrebe za redefinisanjem operatora za ispis. Interfejs mora biti apstraktna klasa. Definisati klasu **`Vector3D`** koja predstavlja vektor u trodimenzionom prostoru. Korištenjem preklapanja operatora, omogućiti osnovne operacije nad trodimenzionim vektorima: sabiranje, množenje skalarom i skalarni proizvod. Dva vektora se smatraju jednakim samo ako su im odgovarajuće komponente jednake. Preklopiti unarni minus operator, tako da se njime vrši operacija množenja vektora sa -1. Operacija množenja skalarom treba biti komutativna. Omogućiti ispis vektora na izlazni tok implementacijom odgovarajućeg interfejsa. Na pravilan način omogućiti pristup x, y i z komponentama vektora. Definisati statičku metodu koja konstruiše i vraća nula-vektor. Pored navedenog, ispuniti i sljedeće zahtjeve:
    a. Definisati zaglavlje klase **`Vector4D`** koja javno nasljeđuje **`Vector3D`** i pravilno proširuje funkcionalnost tako da se dozvoljava rad sa četvrtom komponentom w. Omogućiti kastovanje iz objekta klase **`Vector3D`** u objekat klase **`Vector4D`**. Pravilno implementirati naslijeđenu metodu za sabiranje vektora. Izbjeći dupliranje koda iz natklase. Izbjeći redefinisanje metoda čija se funkcionalnost ne mora mijenjati. Greške obraditi sa izuzecima.
    b. Definisati klasu **`Vector2D`** koja predstavlja vektor u dvodimenzionom prostoru, koja javno nasljeđuje **`Vector3D`** i omogućava korištenje potrebnih funkcionalnosti iz natklase. Izbjeći dupliranje koda iz natklase. Izbjeći redefinisanje metoda čija se funkcionalnost ne mora mijenjati. Korištenjem mehanizma izuzetaka onemogućiti nepotrebne funkcionalnosti. Pri tome, neophodno je koristiti sopstvenu klasu za izuzetak koja nasljeđuje odgovarajuću baznu klasu za izuzetke iz standardne biblioteke.

2. **(10)** Definisati šablonske interfejse za rad sa kolekcijama podataka:
    a. Definisati šablonski interfejs **`IEnumerableCollection<T>`** koji predstavlja prebrojivu kolekciju podataka. Interfejs mora da sadrži odgovarajuće metode za umetanje elementa na proizvoljnu poziciju, za uklanjanje elementa sa proizvoljne pozicije i za dobavljanje elementa sa proizvoljne pozicije. Omogućiti dobavljanje veličine niza. Greške se trebaju obraditi sa izuzecima.
    b. Definisati šablonski interfejs **`IExtendedEnumerableCollection<T>`** koji, pored mogućnosti iz **`IEnumerableCollection`**, omogućava dobavljanje elementa kolekcije sa proizvoljne pozicije i dodavanje elementa na kraj kolekcije korištenjem odgovarajućih operatora bez bočnih efekata. Interfejs treba da sadrži i moguće varijante tih operatora sa dodjelom koji omogućavaju ulančano pozivanje, kao i metodu za indeksiranje koja kao dodatni parametar sadrži referencu na **`bool`** tip podatka kojim se signalizira da li je došlo do greške pri indeksiranju. Pravilno specifikovati podrazumijevanu implementaciju navedenih operatora i metode za indeksiranje, korištenjem postojećih metoda iz interfejsa **`IEnumerableCollection`**.

3. **(25)** Definisati šablonsku klasu **`DynamicArray<T>`** koja implementira interfejs **`IEnumerableCollection`.** Ova klasa treba da predstavlja dinamički alociran niz elemenata proizvoljnog tipa podatka. Niz treba da se automatski proširuje za broj pozicija (podrazumijevano jedan) vraćen pozivom odgovarajuće zaštićene metode pri umetanju novog elementa u niz, ukoliko je broj elemenata dostigao kapacitet niza. Pri uklanjanju elementa vrši se pomjeranje preostalih elemenata, provjerava se uslov za realokaciju pozivom odgovarajuće zaštićene virtuelne metode (koja podrazumijevano uvijek vraća **`true`**) i zatim se vrši realokacija prostora ukoliko je taj uslov zadovoljen. Omogućiti inicijalizaciju kolekcije inicijalizatorskom listom. Greške obraditi izuzecima. Korištenjem koncepata, ograničenja i statičkih pretpostavki obezbijediti ograničenja: 1) za tip **`T`** mora da postoji

podrazumijevani konstruktor, 2) objekti tipa **T** se mogu ispisati na izlazni tok, 3) dva objekta tipa **T** se mogu pomnožiti i rezultat je tipa **T**. Implementirati metodu za množenje svih elemenata niza. Ispuniti i sljedeće zahtjeve:

a. Implementirati metodu bez bočnih efekata za filtriranje elemenata niza koja kao rezultat vraća **DynamicArray** sa odgovarajućim tipskim argumentom, a koja kao parametar prihvata funkcijski objekat.
b. Implementirati metodu bez bočnih efekata za transformaciju elemenata niza koja kao rezultat vraća **DynamicArray** sa odgovarajućim tipskim argumentom, a koja kao parametar prihvata odgovarajući funkcijski objekat za transformaciju kojim se svi elementi na proizvoljan način transformišu u isti proizvoljan tip podatka.
c. Definisati šablonsku klasu **InsertBufferedArray<T>** koja je specijalizacija klase **DynamicArray** i koja, kao podatak-član, sadrži broj pozicija za koji će se proširiti niz pri dostizanju kapaciteta usljed dodavanja novih elemenata. Osim navedenog, klasa **InsertBufferedArray<T>** ne smije da uvodi više novih podataka članova ni metoda u odnosu na natklasu, s tim da je dozvoljeno da se redefiniše tačno jedna metoda iz natklase.
d. Definisati šablonsku klasu **RemovalBufferedArray<T>** koja je potklasa klase **DynamicArray** i koja, kao podatak-član, sadrži brojač obrisanih elemenata i funkcijski objekat koji kao jedini parametar prima referencu na tip enkapsulirajućeg niza. Taj funkcijski objekat se inicijalizuje kroz konstruktor i ne može se mijenjati nakon što je objekat kreiran. Ova klasa treba da redefiniše odgovarajuću metodu tako da se, pri uklanjanju elementa iz niza, prvo provjeri uslov za realokaciju pozivom navedenog funkcijskog objekta kome je proslijeđen trenutni niz kao argument, te realokacija izvrši samo ukoliko je taj uslov zadovoljen. Klasa **RemovalBufferedArray<T>** ne smije da uvodi više novih podataka članova ni metoda u odnosu na natklasu, s tim da je dozvoljeno da se redefiniše tačno jedna metoda iz natklase.
e. Definisati klasu **BufferedArray<T>** koja pravilno, višestrukim nasljeđivanjem, obezbjeđuje funkcionalnosti iz **InsertBufferedArray<T>** i iz **RemovalBufferedArray<T>**, poštujući sva ograničenja navedena u zadatku.

4. **(10)** U glavnoj funkciji programa demonstrirati sljedeće stavke:
   a. Konstrukcija **DynamicArray** niza sa proizvoljnim tipom elemenata, uključujući kopirajuću konstrukciju i pomjerajuću konstrukciju, kao i korištenje operatora dodjele, kopirajućeg operatora dodjele i pomjerajućeg operatora dodjele. Demonstrirati dodavanje i uklanjanje elemenata.
   b. Demonstrirati filtriranje i transformaciju kolekcije tipa **DynamicArray** korištenjem lambda operatora.
   c. Korištenjem algoritama i struktura podataka iz standardne biblioteke, sortirati kolekciju objekata tipa **Vector3D** po proizvoljnom kriterijumu i ispisati sve permutacije kolekcije.

Sve klase moraju da poštuju pravilo trojke i pravilo petorke. Sve klase koje nisu interfejsi treba da po potrebi, na standardan način, omoguće duboko kopiranje i optimizacije performansi pri pomjeranju. Omogućiti polimorfno ponašanje svih klasa. Izbjeći svako curenje memorije. Obavezno se pridržavati principa objektno-orijentisanog programiranja i pravilno izvesti enkapsulaciju i skrivanje informacija (uključujući i ograničavanje pristupa članovima klasa). Obavezno razdvojiti interfejs od implementacije, osim pri definisanju šablonskih klasa. Svi podaci-članovi klase moraju biti inicijalizovani. Izbjeći dupliranje koda. Minimizirati količnu koda, oslanjajući se na implicitno ponašanje u jeziku, pri čemu je potrebno pridržavati se principa pisanja čistog koda. U svrhu razdvajanja odgovornosti i izbjegavanja dupliranja koda, po potrebi pisati pomoćne klase i metode. Pravilno omogućiti upotrebu svih klasa u konstantnom kontekstu. Svi interfejsi moraju biti implementirani kao apstraktne klase. Eksplicitno koristiti ključnu riječ **override**. Koristiti izuzetke za obradu svih grešaka, uključujući i greške u logici, osim gdje je to drugačije naglašeno. Za svaku metodu eksplicitno naznačiti da li ta metoda baca izuzetak, korištenjem odgovarajućeg specifikatora. Za dinamičku alokaciju memorije, obavezno koristiti operatore **new** i **delete** i izbjeći korištenje funkcija za alokaciju memorije iz C biblioteka (**malloc**, **calloc**, **realloc**, **free**). Korištenje STL struktura podataka na mjestima članova klasa ili parametara metoda nije dozvoljeno, osim u stavkama zadatka gdje je to naglašeno.