PROGRAMSKI JEZICI 1

Zadatak 3 – Nasljeđivanje i polimorfizam

Postavka

Kreirati projekat pod nazivom *Zadatak3* unutar rješenja (i.e. *Visual Studio solution*) pod nazivom *PJ1Zadaci2020*, te kreirati projekat pod nazivom *Zadatak3.Demo* unutar istog rješenja. Projekat *Zadatak3.Demo* treba da referencira biblioteku iz projekta *Zadatak3* i demonstrira njen rad.

Zadatak

Unutar projekta *Zadatak3*, kao statičku biblioteku, implementirati klase za rad sa dinamičkim kolekcijama, i to prema sljedećim uputstvima.

- → Pri implementaciji klasa, koristiti isključivo dinamičku alokaciju memorije.
- → Dozvoljeno je korištenje funkcija iz standardne biblioteke, te je dozvoljeno korištenje kolekcija iz standardne biblioteke, ako one nisu korištene pri definisanju podataka članova i parametara javnih metoda.
- → Implementirati interfejs **`IPrintable`** koji obezbjeđuje ispis objekata na standardni izlaz, tako da se koristi operator za ispis na standardni izlaz, te da se on ponaša polimorfno, bez potrebe za redefinicijom za svaku klasu koja implementira interfejs. Intefejs **`IPrintable`** predstavlja „ono što može da se ispiše“.
- → Implementirati interfejs **`IComparable`** koji obezbjeđuje poređenje dva podatka klase **`IComparable`** korištenjem operatora trosmjernog poređenja, analogno implementaciji interfejsa **`IPrintable`**. Intefejs **`IComparable`** predstavlja „ono što može da se uporedi“.
  - ⇝ Obezbijediti da sve što može da se uporedi može da se ispiše.
- → Implementirati klasu **`Complex`** koja predstavlja kompleksan broj, te implementirati bar osam osnovnih operacija nad ovim entitetom. Omogućiti kastovanje između tipa **`Complex`** i primitivnih numeričkih tipova.
  - ⇝ Implementirati interfejs **`IPrintable`**.
  - ⇝ Implementirati interfejs **`IComparable`**, tako da se omogući poređenje kompleksnih brojeva po rastojanju od koordinatnog početka u kompleksnoj ravni.
- → Implementirati klasu **`UnorderedSet`** koja predstavlja neuređen skup podataka tipa **`IPrintable`**.
  - ⇝ Elemente kolekcije čuvati kao pokazivače na originalne podatke.
  - ⇝ Obezbijediti umetanje i brisanje elemenata kolekcije uz odgovarajuću alokaciju novih praznih članova niza radi akomodacije za nove elemente.
    - ↳ Implementirati odgovarajuće zaštićene metode koje omogućavaju redefiniciju mehanizma za alokaciju novih članova niza (i.e. omogućiti da se u izvedenim klasama izmijeni način alokacije novih elemenata kada se dostigne kapacitet trenutno korištenog niza pokazivača).
  - ⇝ Obezbijediti dobavljanje elemenata kolekcije korištenjem odgovarajućeg operatora.
  - ⇝ Obezbijediti ispis elemenata kolekcije implementacijom interfejsa **`IPrintable`**.
- → Implementirati klasu **`OrderedSet`**, izvedenu iz klase **`UnorderedSet`**, koja predstavlja uređen skup podataka tipa **`IComparable`**.
  - ⇝ Podaci treba da se čuvaju u sortiranom poretku, korištenjem operatora trosmjernog poređenja.
  - ⇝ Klasa **`OrderedSet`** ne smije da dodaje nove podatke članove u odnosu na natklasu.
  - ⇝ Obezbijediti istu funkcionalnost kao u natklasi, uz omogućavanje sortiranog ispisa.
  - ⇝ Korištenjem mehanizma kovarijacije, obezbijediti specijalizaciju povratnog tipa operatora indeksiranja.

→ Implementirati klasu **`BufferedSet`**, izvedenu iz klase **`UnorderedSet`**, koja predstavlja neuređen skup podataka tipa **`IPrintable`** i omogućava baferisanu alokaciju novih elemenata niza.
  - ↝ Obezbijediti konstruktor koji prihvata broj za koji će biti uvećana dužina alociranog niza pokazivača pri realokaciji.
  - ↝ Redefinisati odgovarajuću metodu iz natklase kako bi se omogućila realokacija bafera pri umetanju novih elemenata.

→ Implementirati klasu **`OrderedBufferedSet`** koja čuva elemente u sortiranom poretku i omogućava baferisanu alokaciju novih elemenata niza.
  - ↝ Klasa **`OrderedBufferedSet`** treba da koristi funkcionalnost iz klasa **`BufferedSet`** i **`OrderedSet`**, korištenjem višestrukog nasljeđivanja.
  - ↝ Izbjeći definisanje novih podataka članova i metoda u klasi **`OrderedBufferedSet`**.

Sve implementirane klase treba da poštuju pravilo trojke i pravilo petorke. Sve implementirane klase treba da budu objedinjene u jedinstven prostor imena, te treba da imaju ispravno razdvojena zaglavlja od implementacije. Pored toga, implementirane klase treba da obezbjeđuju mogućnost korištenja u obliku konstantnih objekata. Preklopiti operatore u svim situacijama kada je to semantički ispravno. Izbjeći svako dupliranje koda.

U projektu *Zadatak3.Demo* demonstrirati rad svih implementiranih klasa i metoda individualno. Demonstrirati polimorfno ponašanje svih klasa iz hijerarhije.