Zadaci za Laboratorijsku vježbu 13

NAPOMENA: Studenti bi trebali da razmisle o zadacima koji će se raditi na laboratorijskoj vježbi prije nego što dođu na vježbu, tako da već u startu imaju osnovne ideje kako riješiti zadatke. U suprotnom, rad na vježbi neće biti produktivan. Činjenica je da će mali broj studenata stići uraditi sve zadatke za vrijeme koje je predviđeno za vježbu. Međutim, svim studentima se toplo savjetuje da dovrše samostalno one zadatke koje nisu stigli uraditi za vrijeme vježbe, s obzirom da je razumijevanje ovog tipa zadataka od vitalne važnosti za polaganje drugog parcijalnog ispita.

1. Definirajte apstraktnu baznu klasu "`Lik`", koja predstavlja temelj za hijerarhijsku izgradnju porodice klasa koje predstavljaju geometrijske likove. Klasa ne sadrži nikakve atribute, nego samo apstraktne metode "`IspisiSpecificnosti`", "`DajObim`" i "`DajPovrsinu`", kao i (konkretnu) metodu "`Ispisi`" koja ispisuje podatke o liku pozivajući metode "`IspisiSpecificnosti`", "`DajObim`" i "`DajPovrsinu`" (tačan format ispisa biće vidljiv iz primjera dijaloga između programa i korisnika). Zatim, iz klase "`Lik`" izvedite konkretne klase "`Krug`", "`Pravougaonik`" i "`Trougao`". U sve tri klase dodajte atribute koji su neophodni za specifikaciju primjeraka te klase (poluprečnik za krug, te dužine stranica za pravougaonik odnosno trougao), zatim odgovarajući konstruktor koji inicijalizira atribute i baca izuzetak tipa "`domain_error`" uz prateći tekst "Neispravni parametri" u slučaju da su parametri besmisleni (poput poluprečnika koji je negativan ili 0, ili dužina stranica od kojih se ne može formirati trougao), te konkretne metode "`IspisiSpecificnosti`", "`DajObim`" i "`DajPovrsinu`" koje ispisuju specifične podatke o konkretnom liku, odnosno vraćaju kao rezultat obim i površinu konkretnog lika. Sa tako definiranim klasama napravite program koji prvo traži da se sa tastature unese broj likova. Nakon toga, korisnik treba da unosi sa tastature podatke o svakom liku u sljedećem formatu:

   `K`r — Krug radijusa r (npr. `K5`)
   `P`a`,`b — Pravougaonik sa stranicama a i b (npr. `P3,2`)
   `T`a`,`b`,`c — Trougao sa stranicama a, b i c (npr. `T3,4,5`)

   Nakon svakog unesenog podatka, program treba dinamički kreirati odgovarajući objekat, i smjestiti pametni pokazivač na njega u vektor (čiji su elementi pametni pokazivači na likove). U slučaju neispravnog unosa, ispisuje se poruka o greški i traži se ponovni unos za isti lik. Po okončanju unosa svih podataka, program treba ispisati podatke o svim unesenim likovima, koristeći poziv metode "`Ispisi`". Dijalog između korisnika i programa trebao bi izgledati poput sljedećeg:

```
Koliko zelite likova: 5
Lik 1: P3,2
Lik 2: pppqqq
Pogresni podaci, ponovite unos!
Lik 2: K2,3
Pogresni podaci, ponovite unos!
Lik 2: K5
Lik 3: T1,2,3
Pogresni podaci, ponovite unos!
Lik 3: T3,4,5
Lik 4: P10,20
Lik 5: K10

Pravougaonik sa stranicama duzine 3 i 2
Obim: 10 Povrsina: 6
Krug poluprecnika 5
Obim: 31.4159 Povrsina: 78.5398
Trougao sa stranicama duzine 3, 4 i 5
Obim: 12 Povrsina: 6
Pravougaonik sa stranicama duzine 10 i 20
Obim: 60 Povrsina: 200
Krug poluprecnika 10
Obim: 62.8319 Povrsina: 314.159
```

   Možete pretpostaviti da će na početno pitanje (koliko želite likova) uvijek biti dat smislen odgovor. Također, ne trebate testirati šta će se desiti ukoliko nema dovoljno memorije (nek tada program prosto "crkne").

2. Definirajte i implementirajte apstraktnu baznu klasu "`ApstraktniStudent`", kao i dvije konkretne klase "`StudentBachelor`" i "`StudentMaster`" izvedene iz bazne klase "`ApstraktniStudent`". Bazna klasa "`ApstraktniStudent`" posjeduje privatne atribute koji čuvaju podatke o imenu i prezimenu studenta (oba su tipa "`string`"), broju indeksa (cijeli broj), broju položenih ispita (cijeli broj) i prosječnoj ocjeni (realan broj), i nikave druge atribute. Konstruktor klase inicijalizira ime i prezime studenta, te broj indeksa na vrijednosti zadane parametrima, dok se broj položenih ispita i prosječna ocjena inicijaliziraju respektivno na 0 i 5. Klasa dalje posjeduje trivijalne pristupne metode "`DajIme`", "`DajPrezime`", "`DajBrojIndeksa`", "`DajBrojPolozenih`" i "`DajProsjek`" koje vraćaju vrijednosti svih odgovarajućih atributa. Metoda "`RegistrirajIspit`" sa jednim parametrom, služi za registraciju novog ispita. Ona povećava broj položenih ispita za jedinicu i ažurira prosječnu ocjenu u skladu sa ocjenom iz novopoloženog ispita, koja se zadaje kao parametar (u slučaju neispravne ocjene, baca se izuzetak tipa "`domain_error`" uz prateći tekst "Neispravna ocjena"). Ocjena 5 se prihvata kao legalna, ali se ignorira (tj. niti se ažurira prosjek, niti se povežava broj položenih ispita). Metoda "`PonistiOcjene`" bez parametara poništava sve registrirane ocjene (tj. dovodi objekat u stanje kakvo je bilo odmah nakon kreiranja objekta). Konačno, klasa sadrži i apstraktne metode "`IspisiPodatke`" i "`DajKopiju`" bez parametara. Metoda "`DajKopiju`" će biti potrebna za polimorfno kopiranje u kopirajućem konstruktoru klase "`Fakultet`" u narednom zadatku, a njene implementacije u izvedenim klasama treba da obezbijede kreiranje identične kopije objekta nad kojim su pozvane, uz vraćanje adrese novokreirane kopije kao rezultata.

   Izvedena klasa "`StudentBachelor`" razlikuje se od bazne klase "`ApstraktniStudent`" samo po tome što sadrži konkretnu implementaciju apstraktnih metoda "`IspisiPodatke`" i "`DajKopiju`". Metoda "`DajKopiju`" treba da kreira identičnu kopiju objekta tipa "`StudentBachelor`" nad kojim je pozvana, dok metoda "`IspisiPodatke`" u ovoj klasi treba ispisati podatke o studentu u obliku

   `Student bachelor studija <`*`ime`*`> <`*`prezime`*`>, sa brojem indeksa <`*`indeks`*`>,`
   `ima prosjek <`*`prosjek`*`>.`

   Izvedena klasa "`StudentMaster`" razlikuje se od bazne klase "`ApstraktniStudent`" prvo po tome što sadrži dodatni atribut koji čuva informaciju kada je student završio prvi stepen studija (bachelor studij), kao i dodatni parametar u konstruktoru koji omogućava postavljanje ovog atributa. Naravno, metodu "`DajKopiju`" u ovoj klasi treba implementirati tako da kreira identičnu kopiju objekta tipa "`StudentMaster`" nad kojim je pozvana, dok implementacija metode "`IspisiPodatke`" u ovoj klasi treba da ispisuje podatke o studentu u sljedećem obliku:

   `Student master studija <`*`ime`*`> <`*`prezime`*`>, sa brojem indeksa <`*`indeks`*`>,`
   `zavrsio bachelor studij godine <`*`godina`*`>, ima prosjek <`*`prosjek`*`>.`

   Sve metode koje treba da budu inspektori obavezno deklarirajte kao takve. Napišite i mali testni program u kojem ćete testirati klase "`StudentBachelor`" i "`StudentMaster`" (naravno, primjerke klase "`ApstraktniStudent`" ne možete ni kreirati, jer je apstraktna).

3. Napravite klasu "`Fakultet`" koja čuva polimorfnu kolekciju podataka izvedenih iz bazne klase "`ApstraktniStudent`" (to mogu biti klase "`StudentBachelor`" i "`StudentMaster`", ali može biti i bilo koja druga novonapisana klasa izvedena iz klase "`ApstraktniStudent`". Podaci se čuvaju u dinamički alociranim objektima čije se adrese čuvaju u vektoru (običnih) pokazivača na objekte nekog od tipova izvedenih iz tipa "`ApstraktniStudent`", i taj vektor je jedini privatni atribut ove klase. Primjerci klase "`Fakultet`" moraju se moći kreirati ne navodeći nikakve dopunske informacije. Dalje, klasa treba imati destruktor koji oslobađa sve resurse koje su objekti tipa "`Fakultet`" zauzli tokom svog života, zatim kopirajući konstruktor i kopirajući operator dodjele koji omogućavaju da se objekti tipa "`Fakultet`" mogu sigurno kopirati i međusobno dodjeljivati korištenjem strategije dubokog kopiranja (za potrebe kreiranja kopija individualnih objekata pohranjenih u kolekciji koristite funkciju "`DajKopiju`"), te pomjerajući konstruktor i pomjerajući operator dodjele koji optimiziraju postupak kopiranja za slučaj kada su u igri privremeni objekti. Predviđene su dvije metode istog imena "`UpisiStudenta`" koje služe za upis novog studenta, od kojih jedna ima tri, a druga četiri parametra. Metoda sa tri parametra upisuje studenta bachelor studija, pri čemu su parametri broj indeksa, ime i prezime, dok metoda sa četiri parametra upisuje studenta master studija, pri čemu četvrti dodatni parametar predstavlja godinu kada je student završio bachelor studij. Obje metode bacaju izuzetak tipa "`logic_error`" uz prateći tekst "Student sa zadanim brojem indeksa vec postoji" ukoliko student sa zadanim brojem indeksa već postoji. Njoj je suprotna metoda "`ObrisiStudenta`" koja briše studenta sa brojem indeksa koji se zadaje kao parametar. U slučaju da

student sa zadanim brojem indeksa ne postoji, treba baciti izuzetak tipa "`logic_error`" uz prateći tekst "Student sa zadanim brojem indeksa ne postoji". Predviđen je i preklopljeni operator "`[]`" koji vraća referencu na studenta sa zadanim brojem indeksa (ukoliko takav ne postoji, baca se isti izuzetak kao u prethodnom slučaju). Pri tome, ako se ovaj operator primijeni na konstantni objekat tipa "`Fakultet`", treba vratiti referencu na konstantni objekat, da se onemogući njegova izmjena (povratni tip ne smije prosto biti ne-referenca tipa "`ApstraktniStudent`", jer bi se kopiranjem nekog objekta u privremeni objekat tog tipa koji se vraća iz funkcije izgubile specifičnosti objekta koji se kopira). Zahvaljujući ovom operatoru, moguće je vršiti manipulacije nad studentima pohranjenim u kolekciji, bez potrebe da za tu svrhu dodajemo nove metode. Recimo, ako je "`etf`" neki objekat tipa "`Fakultet`", trebaju biti moguće konstrukcije poput

```
etf[14922].RegistrirajIspit(8);
```

Konačno, treba podržati i Metodu "`IspisiSveStudente`" koja ispisuje spisak svih studenata sa pripadnim podacima (pozivom metode "`IspisiPodatke`" za svakog studenta), sortiran u opadajućem poretku po prosječnoj ocjeni (tj. student sa najvećom prosječnom ocjenom se ispisuje prvi). Ukoliko dva studenta imaju isti prosjek, treba prije da se ispiše onaj sa manjim brojem indeksa. Ova metoda treba obavezno da bude inspektor, a kao inspektore treba deklarirati i sve ostale metode koje po svojoj prirodi ne mijenjaju stanje objekta. Obavezno napišite i testni program u kojem ćete testirati sve elemente navedene klase.

4. Napravite surogatsku klasu "`Student`" koja predstavlja polimorfni omotač za proizvoljnu vrstu studenata. Preciznije, promjenljive tipa "`Student`" moraju biti takve da se u njih može smjestiti bilo student bachelor studija bilo student master studija (tj. sadržaj promjenjive čiji je tip bilo tip "`StudentBachelor`" bilo tip "`StudentMaster`"), pa čak i promjenljiva bilo kojeg tipa koji je izveden iz apstraktnog tipa "`ApstraktniStudent`" (što uključuje i tipove koji će eventualno biti kreirani u budućnosti, poput "`StudentDoktorant`" ili "`StudentSpecijalizant`"). Naravno, sa promjenljivim tipa "`Student`" mogu se raditi sve operacije koje su predviđene da rade sa svim vrstama studenata neovisno od njihovog podtipa (tj. sve operacije koje su definirane u interfejsu apstraktne bazne klase "`ApstraktniStudent`"), mogu se bezbjedno kopirati, međusobno dodjeljivati, itd. Na primjer:

```
Student s1, s2;
s1 = StudentBachelor("Dusko", "Dugousko", 1234);              // "s1" je bachelor
s2 = StudentMaster("Paja", "Patak", 4312, 2015);              // a "s2" master
s1.RegistrirajIspit(6); s1.RegistrirajIspit(9);
s2.RegistrirajIspit(8);
s1.IspisiPodatke(); s2.IspisiPodatke();
s1 = s2;                                                      // sad je i "s1" master
s1.IspisiPodatke();
```

Obavezno napišite i testni program u kojem ćete testirati funkcionalnost razvijene surogatske klase.