Drugi međuispit iz Programskih paradigmi i jezika

10. svibnja 2010.

Važne napomene:

- Ispit se piše 150 minuta
- Odgovori na pitanja moraju biti koncizni, jasni i precizni. Ukoliko koncept treba objasniti, očekuje se što kraće i jednostavnije objašnjenje.
- Zabranjeno je prepisivanje. Isto će se kazniti za to na Fakultetu predviđenim mjerama.

1. (2 boda) Objasnite kako je moguće koristiti varijable, svojstva i metode osnovne klase (klasa roditelj; ona koju nasljeđuje izvedena klasa) označene modifikatorima vidljivosti `private`, `public` i `protected` unutar izvedenih klasa (klase djeca; one koje nasljeđuju osnovne klase).

   Pristup članovima označenim s protected ili public unutar izvedene klase je moguć. Pristup članovima označenim s private nije moguć unutar izvedene klase, već samo unutar osnovne klase.

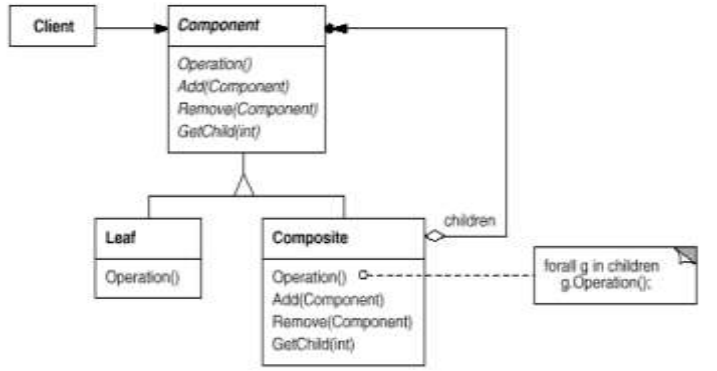
   Upute: Samo točno rješenje donosi dva boda. Bilo koja pogreška se kažnjava s 0 bodova, bez iznimki.

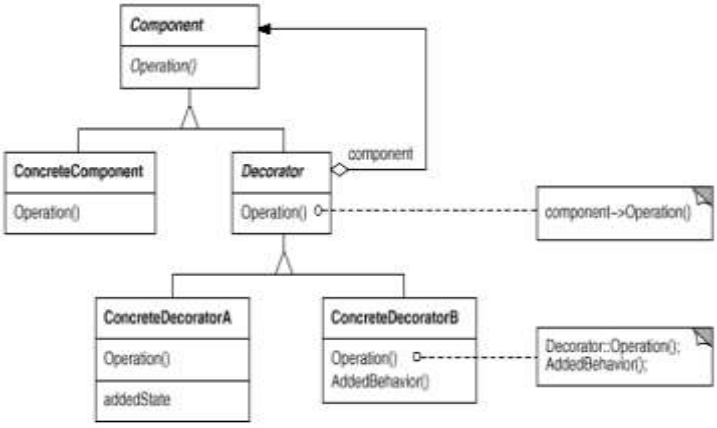
2. (3 bodova) Objasniti razliku uloge višeobličja (polimorfizma) u oblikovnim obrascima Composite i Decorator.
   Napomena: Nije potrebno detaljno ilustrirati oba oblikovna obrasca već samo dijelove koji se odnose na korištenje polimorfizma. Uz ilustraciju nužno je i detaljno opisno objašnjenje razlike uloge polimorfizma u navedenim obrascima.

   (1 bod) U slučaju oblikovnog obrasca <u>Composite</u> konkretna klasa Composite (vidjeti prvi dijagram) sadrži listu objekata tipa Component (apstraktna klasa ili sučelje iz koje je Composite izveden) te jednu ili više polimorfnih metoda (metoda Operation klase Composite na prvom dijagramu). Prilikom poziva metode Operation sučelja (ili apstraktne klase) Component, poziv metode se polimorfizmom <u>propagira</u> na sve elemente liste koju sadrži klasa Composite što se u konačnici može i propagirati na niže hijerarhijske razine (ukoliko iste postoje), ovisno o broju razina strukture (definiranom sa strane korisnika).

   (1 bod) Za razliku od gornjega, pozivi nad metodama sučelja ili apstraktne klase Component oblikovnog obrasca Decorator se polimorfno prosljeđuju klasi Decorator koja nema listu objekata već implementaciji sučelja osnovne klase <u>dodaje</u> ponašanje (npr. poziv neke privatne metode klase Decorator). U skladu s time propagacija polimorfnih poziva kod Decoratora jest linearna (jer npr. možemo imati nekoliko Decoratora jednog za drugim oko jednog početnog objekta), a kod Composite jest hijerarijska i nalikuje na stablo.

   (1 bod) Studenti ne moraju ilustrirati pune dijagrame, već samo sljedeće: Za Composite klasu Component i klasu Composite i to na način da je vidljivo je klasa Component sučelje ili apstraktna klasa, a Composite konkretna klasa i da obje imaju metodu(e) istog imena s time da ju klasa Composite implementira. Mora biti vidljiva i veza „children" kojem se pokazuje da klasa Composite sadrži listu objekata klase Component. Za Decorator moraju biti vidljive klase Component i Decorator s vezom decoratora i Component i ponašanjem Operation koje se nadjačava u Decorator.





3. (10 bodova) Potrebno je ostvariti objektni model meteorološke stanice. Model je potrebno ilustrirati dijagramom i programom u jeziku C#. Meteorološka stanica sastoji se od više senzora s kojih se u diskretnim vremenskim intervalima očitavaju vrijednosti (svakih n sekundi, pri čemu je n parametar). Senzori mogu biti ili analogni ili digitalni. Digitalni senzori su, primjerice, termometar, higrometar i barometar. Analogni su anemometar, kišomjer i vjetrokaz.
   Na jednu meteorološku stanicu moguće je spojiti više istovrsnih i/ili raznovrsnih senzora, te ih dinamički odvajati i priključivati na stanicu. Tek nakon što se isključi, senzor ne obavlja mjerenja (kako bi ga se isključilo nije dovoljno ne prikazivati njegovo mjerenje već ga je potrebno u potpunosti isključiti iz sustava meteorološke stanice).
   Meteorološka stanica klijentima mora omogućiti dohvat trenutnog stanja svih priključenih senzora (zadnje očitanje sa svakog senzora) kao i dohvat svih očitavanja svih senzora. Prilikom zahtjeva za dohvat očitanja klijent mora biti u mogućnosti parametrom specificirati mjernu jedinicu u kojoj stanica vraća očitanja (npr. temperatura u °C, °F ili K, brzina vjetra u km/h ili čvorovima, tlak zraka u barima ili hektopaskalima itd).
   Napomena: Program koji ilustrira model mora sadržavati klase koje se sastoje varijabli, svojstava i prototipa metoda. Nije potrebno pisati implementacije metoda. Kompliciranе programske odsječke objasniti komentarom.

   Ovaj je zadatak moguće riješiti na mnogo načina i zato se neće davati konkretno rješenje već samo opis.
   (3 bodova) Prikaz dijagramom (sve klase i metode navedene donjim opisom; udio po komponentama vidjeti dolje)
   (5 bodova) Kostur programa (sve klase i metode navedene donjim opisom; udio po komponentama vidjeti dolje)
   (1 bod) Adekvatno imenovane varijable (dodjeljuje se temeljem općenitog dojma)
   (1 bod) Uredan i pregledan dijagram, uredan i lako čitljiv kod (dodjeljuje se temeljem općenitog dojma)

   Rješenje mora sadržavati (sve stvari koje su viška, a imaju smisla i nisu pogrešne prema tekstu zadatka su dobrodošle):

- ***(10%) Klasu za meteorološku stanicu (s generičkom listom senzora; i konstruktorom koji ima n za broj očitanja kao parametar)***
- ***(30%) Apstraktnu klasu ili sučelje za senzor (metoda za očitanje vrijednosti), apstraktnu klasu ili sučelje za analogni i digitalni senzor nasljeđene iz senzora te konkretne klase termometar, higrometar, barometar (nasljeđene iz digitalnog senzora), anemometar, kišomjer i vjetrokaz (nasljeđene iz analognih senzora). U svim klasama nasljeđenim iz digitalnog i analognog senzora mora biti kostur implementacije metode za očitanje vrijednosti.***
- ***(20%) Metoda za očitanje vrijednosti klase/sučelja mora imati vlastiti povratni tip (npr. klasa RezultatOcitanja). Sve klase nasljeđene iz digitalnog i analognog senzora moraju za povratni tip metode očitavanja vrijednosti vlastiti tip nasljeđen iz klase RezultatOcitanja. RezultatOcitanja mora imati i svojstvo kojim se specificira mjerna jedinica.***
- ***(30%) MeteoroloskaStanica ima metodu za dodavanje senzora (parametar metode jest osnovna klasa senzora; metodu za isključivanje senzora (parametar metode jest osnovna klasa senzora); metodu za dohvat trenutnog stanja svih senzora (vraća Listu objekata tipa RezultatOcitanja) i metodu za dohvat svih očitanja svih senzora (vraća rječnik tipa (Senzor, List<RezultatOcitanja>). Zadnje dvije metode kao parametar moraju primati klasu/strukturu kojom se specificira izlazna mjerna jedinica).***
- ***(10%) Potrebno je implementirati i posebnu statičku klasu za pretvaranje mjernih jedinica. Ona mora imati statičke metode za pretvaranje vrijednosti jedne jedinice u vrijednosti druge jedinice (mora se ograničiti samo na smislene parove – npr. C u K i sl; a ne kmh u C).***

4. (3 bodova) Objasnite princip objektnog dizajna *Open/Closed principle* i primjerom u programskom jeziku po izboru ilustrirajte kršenje navedenog principa. Predložite rješenja u skladu s principom *Open/Closed*.
   Napomena: Programski odsječak treba biti što je moguće jednostavniji. Strelicama naznačiti problematične dijelove te dati kratki tekstualni opis problema.

   (1.5 bodova) Klase bi trebale biti otvorene za proširivanje, a zatvorene za modifikaciju. Trebalo bi težiti dizajnu klasa koji se neće morati mijenjati kada dođu zahtjevi za novom funkcionalnošću. Najbolji način za proširenje sustava je dodavanje nove funkcionalnosti, a ne mijenjanje postojeće funkcionalnosti. Postiže se uz pomoć apstrakcije, polimorfizma, nasljeđivanja i sučelja.

   (1.5 bodova) Studenti mogu biti maštoviti prilikom predlaganja primjera. Bilo koji primjer koji ilustrira kako se izmjenama u objektnom modelu može lakše dodavati funkcionalnost. Primjer može biti jednostavniji – od proglašavanjem metode virtualnom i njenim nadjačavanjem; do oblikovnih obrazaca tipa Decorator (dinamičko dodavanje funkcionalnosti). Potrebno je procijeniti i da li je student dobro objasnio uzroke i način izmjene te benefite.

5. (2 boda) Objasnite na koji je način moguće parametrizirati primjenu savjeta u prekidnim točkama u aspektno orijentiranom programiranju. Uz objašnjenje navedite i opisni primjer (nije potrebno pisati programske linije).

   (1 bod) Primjerice, u C#, prekidne točke se definiraju oznakama koje se sastoje od uglatih zagrada i imena klasa koje su nasljeđene iz predefiniranih klasa iz Postsharpa. Te oznake u C# nazovamo atributima i stavljamo iz iznad programskih struktura (npr iznad klasa).
   Primjena savjeta u prekidnim točkama se može parametrizirati parametrima koji se dodaju gore navedenim atributima. Njima se primjerice može definirati da se neki savjet primjenjuje samo na neke elemente programske strukture.

   (1 bod) Primjer:
   [AOPMethodBoundary(AttributeTargetMembers="Method*")]
   AOPMethodBoundary jest naziv savjeta
   AttributeTargetMembers je parametar kojim se definira na koje metode se primjenjuje savjet AOPMethodBoundary

Method* jest vrijednost parametra AttributeTargetMembers kojom se definira da se savjet primjenjuje samo na metode koje počinju s Method.

Uputa: Primjer može biti i apstraktan, tj. studenti mogu izmisliti i nazive struktura, sve dok je ispravljaču jasno da student razumije tematiku. Npr:
[MojSavjet(MojAtribut="Method*")]