Međuispit iz Objektnog oblikovanja

18. studenoga 2010.

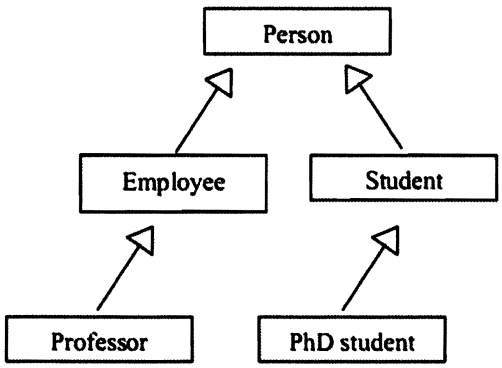
1. (*10 bodova*) U par rečenica navedite i argumentirajte tri razloga zašto je iterativni razvoj bolji od klasičnog *waterfall* procesa razvoja. Navedite i jednu „kontraindikaciju" iterativnog razvoja (dakle, što potencijalno može biti problem).

2. (*10 bodova*) Opišite značenje pojma refactoring-a i objasnite njegovu važnost za agilan razvoj softvera.

3. (*10 bodova*) Pojasnite pojam i ulogu agregata u *domain-driven design.*

4. (*5 bodova*) Objasnite razliku između *interface class* i *abstract class*.

5. (*10* bodova) Dana je sljedeća hijerarhija klasa:



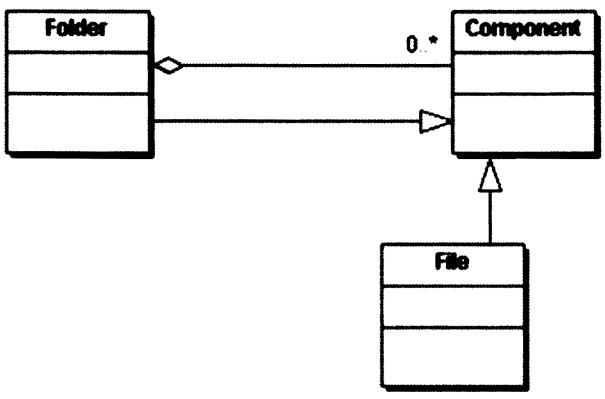
Svaka osoba (Person) ima metodu *startWorking()* i *stopWorking()* koje se pozivaju kad osoba počinje i završava s radom (odnosno, definirane su kao virtualne u klasi Person i prekrivene u izvedenim klasama).

Potrebno je modificirati navedenu hijerarhiju kako bi se dodalo *runtime* praćenje (logiranje) rada osobe. Zahtjev je (hard constraint) da se praćenje može uključivati/isključivati tijekom izvršavanja (odnosno, ne podešava se samo na početku izvođenja) za svaku pojedinačnu klasu i to na način da se može uključiti/isključiti i za cijelu klasu i za pojedinačno instancirane objekte (ukoliko se za neki objekt desi konfliktna situacija, prednost imaju poštavke za klasu!). Kada je praćenje uključeno, potrebno je u globalno dostupni log zapisivati kada je osoba počela i kada je završila s radom.

Koje bi sve klase trebalo dodati (ako ikoje!?) i kako bi ih trebalo integrirati u gore navedeni dizajn? Što bi trebalo (ako išta!?) promijeniti u implementacijama postojećih funkcija? Da li se vaše rješenje mijenja (i kako) ukoliko se zna da će se u navedenu hijerarhiju dodavati i nove klase i da je lakoća budućeg održavanja važan kriterij kvalitete rješenja?

Napomena: Pretpostavite da postoji globalno dostupna instanca (singleton) klase Logger, koja ima operacije logStart(Person) i logStop(Person) kojima se obavlja konkretno zapisivanje u log.

6. (*10 bodova*) Dan je sljedeći dizajn kojim su modelirani fajlovi i folderi (direktoriji) u nekom operativnom sustavu.



Željeli bi u klasu Folder dodati funkcionalnost za izračunavanje ukupne veličine svih fajlova i foldera unutar tog foldera.

Pitanje za vas je što bi sve trebalo dodati u navedene klase kako bi se ta funkcionalnost ugradila (pretpostavite da postoji funkcija u dostupnom API-ju getFileSize(FILEHANDLE fh) i da klasa File u sebi sadrži člansku varijablu FILEHANDLE koja predstavlja handle na fajl).

Napišite implementaciju svih funkcija koje ste dodali u dizajn (C#/Java/C++, sasvim svejedno, ali naznačite što ste odabrali).

7. (*45 bodova*) – *Sustav za planiranje i praćenje realizacije projekata*

OKRENITE STRANICU→→→

Poslovna potreba

Poduzeće Light Speed Technologies Inc. se bavi razvojem softverskih projekata i zbog brzog razvoja i rasta, planiranje i praćenje razvojnih aktivnosti preko Excel tablica više nije dovoljno. Stoga je uočena potreba za razvojem informacijskog sustava kroz koji bi se obavljale aktivnosti planiranja i praćenja realizacije postavljenih ciljeva, kao i praćenje dnevnih aktivnosti developera, što bi uz kvalitetan sustav izvještavanja omogućilo daljni kvalitetan razvoj i rast firme.

Opis domene

Svi projekti u firmi su inicirani inicirani Nalogom za izvršenje (NZI), odnosno za sav posao koji se obavlja u firmi mora postojati izdan odgovarajući radni nalog koji je definiran nazivom, imenom klijenta za kojeg se radi, te datumom početka i predviđenim datumom završetka. Firma je organizirana kroz više odjela s time da su Odjel za razvoj i Odjel za implementaciju ključni odjeli koje treba pokriti novim sustavom (Odjel računovodstva i Odjel nabave svoje aktivnosti obavljaju kroz ERP sustav). U svakom od tih odjela se nalazi više timova, od kojih svaki ima određeni broj članova (developera, odnosno općenitije članova tima – koji mogu biti „stariji" (senior) ili „mlađi" (junior) inženjeri) i jednog voditelja (Team leadera).

Planiranje se radi na tjednoj, mjesečnoj i kvartalnoj bazi definiranjem odgovarajućih ciljeva. Za svaki cilj se u sustav unosi kratki naziv, opis te jedan ili više članova tima odgovornih za realizaciju a također se za ciljeve definiraju i procjene – i to procjene utroška vremena, te procjene materijalnih troškova. Svaka vrsta cilja ima definiran i prioritet (low, medium ili high). U sustavu mora postojati mogućnost za hijerarhijsku organizacija ciljeva što podrazumijeva da se mjesečni ciljevi povezuju (odnosno „pripadaju pod") kvartalni cilj, a tjedni ciljevi pripadaju pod mjesečni cilj. Kvartalni ciljevi su na vrhu hijerarhije i oni se vezuju direktno na NZI. No, mora postojati i mogućnost vezivanja tjednih i mjesečnih ciljeva direktno na NZI (odnosno, da se NE vežu na viši cilj u hijerarhiji).

Pored planiranja, koje se obavlja kroz definiranje ciljeva, prati se i vrijeme potrošeno na njihovu realizacija na način da se zapisuju dnevne aktivnosti developera i povezuju s ciljevima za koje su odrađene. Svaki developer može tijekom dana unijeti proizvoljan broj odrađenih aktivnosti. Za svaku aktivnost se unosi njen kratki naziv, detaljan opis, potrošeno vrijeme te cilj za koji je aktivnost realizirana. Aktivnost se može povezati bilo s tjednim, mjesečnim ili kvartalnim ciljem, bilo direktno na NZI (npr. ukoliko je „uletjelo" nešto iz prošlosti a što nije bilo planirano).

Konkretna realizacija svih ciljeva se prati na način da se svaki cilj može označiti bilo kao 25%, 50%, 75% ili 100% finished. Markiranje mjere završenosti ciljeva odrađuju oni koji su za cilj zaduženi a bitno je pri tome zapisati i timestamp kada je cilj markiran, kako bi se kasnije moglo napraviti graf napretka realizacije ciljeva. Cilj se može označiti i kao „Not finished" u kojem slučaju se mora moći unijeti i bilješka s opisom razloga zašto cilj nije završen.

Napomene:

- Da li će aplikacija biti realizirana kao klijent-server ili web aplikacija za vas i vaše zadatke nije (odnosno ne bi trebalo biti) bitno
- Autorizacijsko-security aspekte informacijskog sustava možete zanemariti
- Možete zanemariti i sustav izvještavanja jer se on bazira na samoj domeni, odnosno ne doprinosi složenosti domene već samo iskorištava definirane podatke

Vaš zadatak je:

a) (*10 bodova*) Identificirati 5-6 **glavnih** use case-ova (znači, navesti ih imenom - uz kratko objašnjenje, ako iz samog imena nije sve jasno)

b) (*10 bodova*) Detaljno razraditi dva osnovna use case-a (obavezno s alternativnim scenarijima!)

c) (*25 bodova*) Izraditi model domene bazirano na danom design story-u koji će biti osnova programskog rješenja