MI 2014.

1. je bio refaktoriranje i njegova važnost u agilnom developmentu.

2. Što su: White i Black box use case. navedi primjer

3. Razlika između abstract class i interface

4. Koji su onih 5 grešaka u OO dizajnu (i opiši)

5. Demetrov zakon. Uz koji se princip usko veže? Koje se pozitivne i negativne strane zakona.

6. Dosta česti primjer s VideoProcesorom. Dodati mogućnost za različite VideoReadere [AVI, MP4, ..] i različite tansformacije videa [koje se mogu ulančavati]. (Pretpostavljam da je videoProc.transformA().transformB()...)

7. Na sustav za upravljane prometom(dohvaćanje podataka o cestama) dodati dohvaćanje meteroloških podataka. Tu je bilo i spajanje na Jezgru sustava (ali to je napravljeno i oba sustava pozivaju tu konekciju)

8. je bilo da imaš ogromnu fju tipa ComplexFourier(nije se tako točno zvala) i u njoj se koristi dosta različitih funkcijica za dijelove fourierove transformacije i nakon svake se notificira progress bar.

Treba restrukturirati tako da se mogu dodavati nove kompleksne operacije i mijenjati već postojeće.

MI 2013.

1. Objasnite i opišite svojim riječima pojam kohezije u kontekstu objektnog dizajna. S kojim je principom OOD-a kohezija usko povezana?

2. Objasnite razliku između koncepata *overriding* (prekrivanje) i *overloading* (preopterećenje) u objektnoj paradigmi. Ilustrirajte primjerom.

3. Koja je osnovna razlika između Entiteta i Vrijednosnog objekta (*Value Object*)?

4. Navedite i opišite pet karakteristika **lošeg** dizajna.

5. U klasi `WAVAnalyzer` je implementirana funkcionalnost za učitavanje, analizu i transformaciju WAV audio datoteke nekim algoritmom koji računa određene značajke amplitude audio signala te generira transformirani audio zapis.

Deklaracija razreda je sljedeća:

```
class WAVAnalyzer {
      public void loadWAVFile(string filePath);
      public void transform();
      // … plus još sve dodatne potrebne funkcije i članske varijable
}
```

Nakon nekog vremena pojavila je najprije potreba da se omogući učitavanje audio zapisa i iz datoteka drugih formata (.mp3, .wma, .pcm), a zatim i potreba da se ugrade i drugi algoritmi za transformaciju zapisa, s dodatnim zahtjevom da se navedene transformacije mogu ulančavati

(dakle, da se audio signal transformiran jednim algoritmom može staviti kao ulazni signal za transformaciju drugim algoritmom).

Kako bi modificirali opisani dizajn (u skladu s naučenim principima OO oblikovanja) da se omogući ugradnja tražene podrške? Ilustrirati odnose klasa pomoću *class diagrama* te napisati relevantne odsječke programskog koda.

6. U malom startupu su implementirali jednostavan sustav za praćenje projekata i developera koji rade na njima. Klasa `Project` sadrži listu `Developer`-a (klasa koja modelira jednog developera na projektu), od kojih je jedan voditelj projekta (i identificiran zasebnom referencom `_projectLeader` od `Project` na `Developer` klasu), dok se za svakog `Developera` zna na kojem projektu radi (dakle, ima referencu na `Project`) te u kakvom je statusu (enum s vrijednostima `Member` i `Leader`).

I sve bi bilo savršeno dok se nije pojavila potreba da se kroz sustav prati i mijenjanje sastava tima na pojedinim projektima, bilo da se radi o odlascima ljudi iz firme, o prebacivanju djelatnika iz jednog tima u drugi ili o napredovanju/nazadovanju člana tima u statusu.

Vaš zadatak je modificirati opisani dizajn kako bi se za sve članove mogli vjerno rekonstruirati povijesni podaci o članstvu u timovima i njihovom statusu.

7. Mladi developer je dobio zadatak da implementira aplikaciju unutar sustava za nadzor prometa koja će se spajati na centralnu jezgru sustava te na njoj objavljivati sve relevantne podatke prikupljene s uređaja za brojanje prometa. Aplikacija prilikom pokretanja mora učitati određen broj konfiguracijskih parametara iz datoteke (pretpostavite jednostavnu text datoteku sa *string* i *int* parametrima), odraditi spajanje na jezgru (putem relativno kompleksne procedure razmjene kriptiranih podataka s jezgrom – iz sigurnosnih razloga) te nakon toga odrađivati svoj posao prikupljanja i objavljivanja podataka.

Nakon nekog vremena, mladi developer je dobio novi zadatak – ovaj put se radilo o izgradnji aplikacije koja će unutar istog sustava na istu centralnu jezgru objavljivati podatke prikupljene s meteorološke stanice. Mladi developer se odmah prihvatio posla, ali je vrlo brzo shvatio da je prekršio DRY princip, dupliciranjem koda za učitavanje konfiguracijskih parametara te spajanje na jezgre, a koji je već implementirao unutar prve aplikacije. Doduše, ne potpuno zato što je samo određeni broj parametara iz konfiguracije druge aplikacije bio isti kao za prvu aplikaciju (IP jezgre, podaci o statusima te još nekolicina), dok je dio bio specifičan i različit u odnosu na aplikaciju za brojač prometa (kod za spajanje na jezgru i uspostavljanje komunikacije je potpuno isti u oba slučaja).

Kakav bi dizajn Vi predložili mladom developeru kako bi se izbjeglo i minimalno kršenje DRY principa?

8. U nekom sustavu za očitavanje podataka sa senzora definirana je klasa `Senzor` kroz koju se obavlja čitanje, odnosno dohvaćanje pročitanih podataka u softverski sustav. Klasa `Senzor` ima callback funkciju `OnReadData()` koja se poziva svaki put kad se pročita novi podataka i kojoj se prosljeđuje SensorData objekt u kojem su zapisani pročitani podaci. Klasa `SensorData` sadrži jedinstvenu oznaku izvora podatka, odnosno uređaja s kojeg je podatak skupljen (može se

pretpostaviti da je skup izvora podataka, odnosno konkretnih senzora u sustavu fiksan i definiran u compile-timeu) te listu parova (ključ, vrijednost) gdje ključ predstavlja identifikaciju konkretne vrste podatka na dotičnom senzoru.

Vaš zadatak je proširiti sustav s komponentom/skupom klasa koja će obrađivati očitane podatke i slati obavijesti korisnicima u slučaju zadovoljenja postavljenih uvjeta nad očitanim vrijednostima.

Sustav mora omogućiti da se svaki od korisnika u sustavu može pretplatiti na proizvoljan skup notifikacija s time da korisnik za svaku od notifikacija može postaviti i uvjet (definiran nad praćenim podacima) koji određuje slučaj u kojem će se notifikacija poslati. Uvjet može biti jednostavan (reakcija na kritičnu vrijednost – operatori <, <=, >, >=, =) ili pak složeni koji uključuju više različitih izvora podataka i logičke operatore između njih (AND, OR, NOT). Ovisno o zadovoljenom uvjetu ili uvjetima generira se i odgovarajuća poruka koja se korisniku (ovisno o njegovim postavljenim preferencijama) šalje na e-mail ili SMS (postoje utility klase `EmailSender` i `SMSSender` koje obavljaju slanje poruke). Poruka se kontinuirano šalje u određenim intervalima dok se stanje ne normalizira ili korisnik potvrdi da je primio poruku.

Ilustrirati rješenje pomoću *class diagrama* s prikazom odnosa među svim klasama te napisati relevantne odsječke programskog koda.

MI 2012.

1. Black/white use case, razlika i navesti primjer

2. Kohezija, definicija, svojim riječima objasni i kojem je principu OOP sličan.

3. Definiraj overloading i override, te navedi primjer za oba slučaja.

4. Demeterovo pravilo koje se nalazi u required reading sa novčanikom.

5. U klasi `WAVAnalyzer` je implementirana funkcionalnost za učitavanje, analizu i transformaciju WAV audio datoteke nekim algoritmom koji računa određene značajke amplitude audio signala te generira transformirani audio zapis.

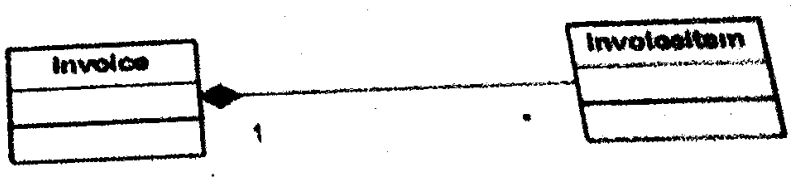
Deklaracija razreda je sljedeća:

```
class WAVAnalyzer {
      public void loadWAVFile(string filePath);
      public void transform();
      // … plus još sve dodatne potrebne funkcije i članske varijable
}
```

Nakon nekog vremena pojavila je najprije potreba da se omogući učitavanje audio zapisa i iz datoteka drugih formata (.mp3, .wma, .pcm), a zatim i potreba da se ugrade i drugi algoritmi za transformaciju zapisa, s dodatnim zahtjevom da se navedene transformacije mogu ulančavati (dakle, da se audio signal transformiran jednim algoritmom može staviti kao ulazni signal za transformaciju drugim algoritmom).

Kako bi modificirali opisani dizajn (u skladu s naučenim principima OO oblikovanja) da se omogući ugradnja tražene podrške? Ilustrirati odnose klasa pomoću *class diagrama* te napisati relevantne odsječke programskog koda.

6. U nekoj poslovnoj aplikaciji postoji uspostavljena standardna veza između Invoice (račun) i InvoiceItem (stavka računa) kako je prikazano na slici. Zatim se pojavio zahtjev da se korisnicima računi mogu slati na jedan od dva načina: faksom ili e-mailom, uz dodatni zahtjev za implementaciju prilagođene enkripcijske sheme kod takvog slanja.



Vaš zadatak je doraditi navedeni dizajn kako bi se omogućila tražena funkcionalnost. Osnovni kriterij kvalitete rješenja je očuvanje kohezije klasa (a što znači da trivijalno rješenje koje se može sažeti u „dodatne" operacije faxInvoice() i mailInvoice() donosi ravno 0 (NULA) bodova!!!), a obavezno uzmite u obzir da su implementacija funkcionalnosti za slanje dokumenata faksom (softverski!) i mailom složene/uknjižene (??????), isto kao i implementacija *custom* enkripcije pa razmislite i u svom rješenju uzmite u obzir da je dobro omogućiti ponovno iskorištavanje te funkcionalnosti i za neke druge vrste dokumenata koje se mogu pojaviti u budućnosti (i zanemarite YAGNI princip u ovom slučaju ☺!).

7. U nekom sustavu za očitavanje podataka sa senzora definirana je klasa `Senzor` kroz koju se obavlja čitanje, odnosno dohvaćanje pročitanih podataka u softverski sustav. Klasa `Senzor` ima callback funkciju `OnReadData()` koja se poziva svaki put kad se pročita novi podataka i kojoj se prosljeđuje SensorData objekt u kojem su zapisani pročitani podaci. Klasa `SensorData` sadrži jedinstvenu oznaku izvora podatka, odnosno uređaja s kojeg je podatak skupljen (može se pretpostaviti da je skup izvora podataka, odnosno konkretnih senzora u sustavu fiksan i definiran u compile-timeu) te listu parova (ključ, vrijednost) gdje ključ predstavlja identifikaciju konkretne vrste podatka na dotičnom senzoru.

Vaš zadatak je proširiti sustav s komponentom/skupom klasa koja će obrađivati očitane podatke i slati obavijesti korisnicima u slučaju zadovoljenja postavljenih uvjeta nad očitanim vrijednostima.

Sustav mora omogućiti da se svaki od korisnika u sustavu može pretplatiti na proizvoljan skup notifikacija s time da korisnik za svaku od notifikacija može postaviti i uvjet (definiran nad praćenim podacima) koji određuje slučaj u kojem će se notifikacija poslati. Uvjet može biti jednostavan (reakcija na kritičnu vrijednost – operatori <, <=, >, >=, =) ili pak složeni koji uključuju više različitih izvora podataka i logičke operatore između njih (AND, OR, NOT). Ovisno o zadovoljenom uvjetu ili uvjetima generira se i odgovarajuća poruka koja se korisniku (ovisno o njegovim postavljenim preferencijama) šalje na e-mail ili SMS (postoje utility klase `EmailSender` i `SMSSender` koje obavljaju slanje poruke). Poruka se kontinuirano šalje u određenim intervalima dok se stanje ne normalizira ili korisnik potvrdi da je primio poruku.

Ilustrirati rješenje pomoću *class diagrama* s prikazom odnosa među svim klasama te napisati relevantne odsječke programskog koda.

8. U softveru za planiranje i praćenje realizacije postavljanih zadataka, definiran je sljedeći dizajn za modeliranje timova i njihovih članova. Klasa `Team` sadrži listu `TeamMember`-a (klasa koja modelira jednog člana tima) od kojih je jedan voditelj tima (i identificiran zasebnom referencom - `_teamLeader` od `Team` na `TeamMember` klasu) dok se za svakog `TeamMembera` zna kojem timu pripada (dakle, ima referencu na Team) te u kakvom je statusu (enum s vrijednostima `Member` i `Leader`). Pored toga je auto-refleksivnom vezom iz `Team`-a na `Team` (klasa `Team` ima referencu `_parentTeam` kojom je identificiran „roditeljski“ tim) omogućeno hijerarhijsko organiziranje timova.

Nakon nekog vremena se pokazala potreba da se sustav nadogradi kako bi se mogle opisati i situacije u kojima dolazi do promjene sastava timova, bilo da se radi o odlascima ljudi firme, o prebacivanju djelatnika iz jednog tima u drugi ili o napredovanju/nazadovanju člana tima u statusu.

Vaš zadatak je modificirati opisani dizajn kako bi se za sve članove mogli vjerno rekonstruirati povijesni podaci o članstvu u timovima u njihovom statusu.

MI 2011.

1. što je refactoring

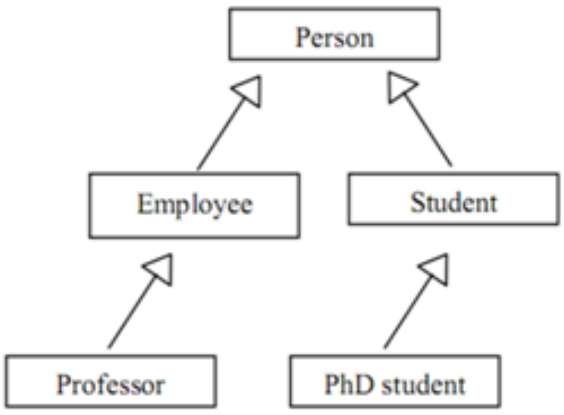
2. koje pravilo krši dani kod i kako ga popraviti? Kod je bio zadan u stilu s slajdova s osobom koja posjeduje psa

5. Zadana je hijerarhija dvodimenzionalnih likova (Paralelogram, Krug, Trokut) koji su svi izvedeni iz razreda Lik2D. Naknadno se pojavila potreba proširenja funkcionalnosti s mogućnošću iscrtavanje likova (metoda Print) na različitim tipovima printera: matrični, laserski, inkjet.

Naravno, algoritam za iscrtavanje npr. pravokutnika razlikuje se od algoritma za iscrtavanje kruga ali i algoritmi za iscrtavanje istog lika na različitim printerima su posve različiti. Koje bi sve klase, metode trebalo dodati da se omogući iscrtavanje zadanog lika na zadanom printeru. Nacrtajte dijagram razreda za vaše rješenje.

Da li se vaše rješenje mijenja (i kako) ukoliko se zna da će se u navedenu hijerarhiju dodavati novi oblici i novi tipovi printera i da je lakoća budućeg održavanja važan kriterij kvalitete rješenja?

6. 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 postavke 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 o i logStop(Person) kojima operacije logStart(Person) se obavlja konkretno zapisivanje u log.

7. razraditi use case za aplikaciju narudžbi u kafeteriji

1. MI 2010.

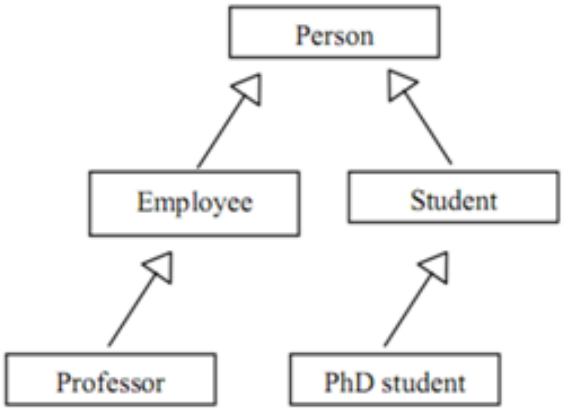
1. U par rečenica navedite i argumentirajte 3 razloga zašto je iterativni razvoj bolji od klasičnog *waterfall* procesa razvoja. Navedite i 1 „kontraindikaciju" iterativnog razvoja (dakle, što potencijalno može biti problem).

2. Opišite značenje pojma refactoringa i objasnite njegovu važnost za agilan razvoj softvera.

3. Pojasnite pojam i uloga agregata u *domain-driven design*.

4. Objasnite razliku između *interface class* i *abstract class*.

5. 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 postavke 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 o i logStop(Person) kojima operacije logStart(Person) se obavlja konkretno zapisivanje u log.

6. 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.

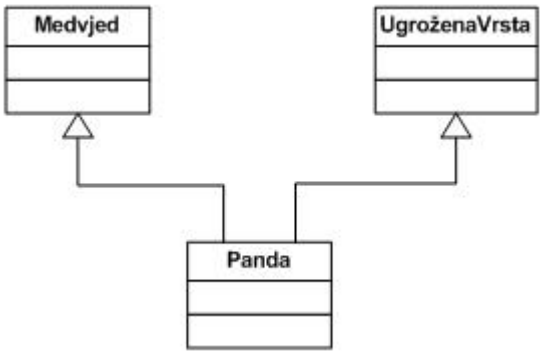
Pitanje za vas je što bi sve trebalo dodati u navedene klase kako bi se ta funkcionalnost ugradila (pretpostavite da postoji funkcija u dostupnom API-ju getFileSize(FILEHANDLE fh) i da klasa File u sebi sadrži člansku varijablu FILEHANDLE koja predstavlja handle na fajl).

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

7. je s use caseovima, ali zadnjih godina to nije bilo na MI pa nisam stavio sad …

OSTALO (pronađeno na FER2/materijalima):

1. Pri razvoju neke aplikacije za praćenje životinja, u objektnom modelu su definirane klase Medvjed i Panda. Budući da je panda vrsta medvjeda, između klasa je ustanovljen odnos nasljeđivanja tako što je Panda izvedena iz Medvjeda. Nakon nekog vremena se pojavio zahtjev da se u aplikaciji prati i, odnosno pravi razlika između ugroženih vrsta životinja i neugroženih (ie. svih ostalih). Developer je to modelirao na sljedeći način:



a) Koji je osnovni problem ove hijerarhije, odnosno njome ustanovljenih klasa i njihovih veza?

b) Kako bi poboljšali ovaj dizajn (radi općenitosti pretpostavite da postoji klasa Životinja, iz koje je izvedena klasa Medvjed, a i ostale životinje relevantne za aplikaciju)? (hint: uvedite baznu klasu ŽivotinjskaVrsta, iz nje izvedite UgroženaVrsta i NeugroženaVrsta i onda … sad vi)

2. U poduzeću „Pero i sinovi d.o.o“ su odlučiti implementirati i uvesti informacijski sustav za praćenje „imovine“ (eng. Asset – i u ovom kontekstu se misli na širi pojam imovine poduzeća u smislu svega onoga u što firma ulaže svoje resurse). S obzirom da je poduzeće orijentirano na visokotehnološku SW i HW proizvodnju, glavni Asseti firme su SW i HW proizvodi, a esencijalno važan aspekt cijele priče je i Release Management, odnosno praćenje različitih izdanih verzija SW i HW proizvoda.

Svaki Asset ima svoje ime, id, i mora se znati popis svih izdanih verzija Asseta (verzije su opisane kasnije!). Pored toga, za svaki Asset je definiran i odgovarajući tip (Asset Type) kojim je opisana vrsta Asseta. Asset Type ima svoj naziv i definiran odgovarajući skup atributa kojima su definirane vrijednosti relevantne za takav tip Asseta. Atributi (Asset Attribute) opisuju određena svojstva vrste Asseta na način da svaki atribut ima definiran svoj naziv i vrstu pripadne vrijednosti (Asset Attribute Type). Predviđeno je da u sustavu postoji ograničen broj tipova atributa koji se mogu

koristiti za opis Asseta i oni obuhvaćaju: value (broj), string, link (http) i document (PDF ili DOC format). ENUM?

Npr. za Asset Type „SW komponenta“, nekoliko tipičnih atributa bi bilo „Release directory“ (tip atributa string), „SVN source code path“ (tip atributa link), dok bi za Asset Type „SPZ znak“ tipični atributi bili „luminoznost“ (value), „potrošnja struje“(value) i sl.

Za svaki Asset se prati popis izdanih verzija (Asset Version), i za svaku verziju Asseta je definirano: oznaka verzije, datum izdavanja verzije, kratki opis promjena te popis (konkretnih) vrijednosti atributa definiranih za takvu vrstu Asseta. Kako se ne bi prilikom svakog izdavanja nove verzije morale nanovo unositi sve potrebne vrijednosti atributa, potrebno je omogućiti da se za svaki pojedinačni Asset mogu definirati default vrijednosti tih atributa, a koje se prilikom izdavanja nove verzije mogu editirati. Svako izdavanje verzije Asseta se mora planirati unaprijed na način da se odredi planirani datum izdavanja Asseta, te definira pripadni broj verzije izdanog Asseta.

ZADATAK: Nacrtajte detaljni dijagram klasa, s potrebnim pojašnjenjima, kojim bi realizirali definirani model domene.