

Sva teorijska pitanja ikad postavljena na
međuispitima iz PROGI (OPP) u zadnjih par
godina

sretno!!!!!!

ROUND I



Akronimi MDSD, MBASE, MBE, MDE, MDD, MDA, DDD označavaju metodologije oblikovanja programske potpore zasnovane na:

- A. Komponentama
- B. Modularnosti
- C. Domenskom znanju
- D. Modelima
- E. Inženjerstvu

Akronimi MDSD, MBASE, MBE, MDE, MDD, MDA, DDD označavajo metodologije oblikovanja programske podpore zasnovane na:

- A. Komponentama
- B. Modularnosti
- C. Domenskom znanju
- D. Modelima**
- E. Inženjerstvu

Zahtjevi koji **ne** sadržavaju konflikte ili kontradikcije u opisima zahtijevanih funkcionalnosti zovu se:

- A. Kompletni
- B. Konzistentni
- C. Razumljivi
- D. Adaptibilni
- E. Provjerljivi

Zahtjevi koji **ne** sadržavaju konflikte ili kontradikcije u opisima zahtijevanih funkcionalnosti zovu se:

A. Kompletni

B. Konzistentni

C. Razumljivi

D. Adaptibilni

E. Provjerljivi

Oblikovanje sustava zasnovano na ponovnoj ili višestrukoj uporabi elemenata sustava (engl. reuse-oriented development) glavno je obilježje kojeg modela procesa programskog inženjerstva?

- A. Unificirani proces
- B. Vodopadnog
- C. Agilnog
- D. Evolucijskog
- E. Komponentnog

Oblikovanje sustava zasnovano na ponovnoj ili višestrukoj uporabi elemenata sustava (engl. reuse-oriented development) glavno je obilježje kojeg modela procesa programskog inženjerstva?

- A. Unificirani proces
- B. Vodopadnog
- C. Agilnog
- D. Evolucijskog
- E. Komponentnog**

U spiralni pristup iteracijama razvoja programske potpore ne pripada sektor:

- A. Studije izvedivosti
- B. Postavljanja ciljeva
- C. Razvoja i validacije
- D. Procjene i smanjivanja rizika
- E. Planiranja

U spiralni pristup iteracijama razvoja programske potpore ne pripada sektor:

A. Studije izvedivosti

B. Postavljanja ciljeva

C. Razvoja i validacije

D. Procjene i smanjivanja rizika

E. Planiranja

Što je od navedenog značajka Unificiranog procesa (engl. Unified Process) modela procesa programskog inženjerstva?

- A. Korištenje malih inkremenata
- B. Malo dokumentacije
- C. Modeli se dokumentiraju dijagramima
- D. Svaka aktivnost se provodi samo u jednoj određenoj fazi razvoja
- E. Ad-hoc određivanje zahtjeva klijenata

Što je od navedenog značajka Unificiranog procesa (engl. Unified Process) modela procesa programskog inženjerstva?

A. Korištenje malih inkremenata

B. Malo dokumentacije

C. Modeli se dokumentiraju dijagramima

D. Svaka aktivnost se provodi samo u jednoj određenoj fazi razvoja

E. Ad-hoc određivanje zahtjeva klijenata

Koja od sljedećih tvrdnji **ne** vrijedi za UML-sekvencijske dijagrame?

- A. Mogu prikazivati aktore i objekte
- B. Dobri su za prikaz komunikacije među objektima
- C. Dobri su za praćenje stanja sustava
- D. Prikazuju objekte kroz životne crte
- E. Prikazuju interakciju u sustavu

Koja od sljedećih tvrdnji **ne** vrijedi za UML-sekvencijske dijagrame?

- A. Mogu prikazivati aktore i objekte
- B. Dobri su za prikaz komunikacije među objektima
- C. Dobri su za praćenje stanja sustava**
- D. Prikazuju objekte kroz životne crte
- E. Prikazuju interakciju u sustavu

Princip oblikovanja programske potpore kod kojeg se, između ostalog, izbjegava uporaba najnovijih tehnologija kao i rijetko upotrebljavanih dijelova knjižnica naziva se:

- A. Oblikuj po ugovoru
- B. Oblikuj za fleksibilnost
- C. Planiraj zastaru
- D. Oblikuj za prenosivost
- E. Oblikuj konzervativno

Princip oblikovanja programske potpore kod kojeg se, između ostalog, izbjegava uporaba najnovijih tehnologija kao i rijetko upotrebljavanih dijelova knjižnica naziva se:

- A. Oblikuj po ugovoru
- B. Oblikuj za fleksibilnost
- C. Planiraj zastaru**
- D. Oblikuj za prenosivost
- E. Oblikuj konzervativno

Princip oblikovanja programske potpore kod kojeg se teži minimizaciji komunikacije komponenata:
(2 točna)

- A. Smanji međuovisnost
- B. Povećaj koheziju
- C. Smanji koheziju
- D. Zadrži razinu apstrakcije
- E. Smanji komunikaciju

Princip oblikovanja programske potpore kod kojeg se teži minimizaciji komunikacije komponenata:
(2 točna)

A. Smanji međuovisnost

B. Povećaj koheziju

C. Smanji koheziju

D. Zadrži razinu apstrakcije

E. Smanji komunikaciju

Metode razreda, koje se deklariraju unutar razreda ključnom riječi (1), pozivaju se preko (2).

- A. (1) static (2) naziva razreda
- B. (1) static (2) reference na objekt
- C. (1) abstract (2) naziva razreda
- D. (1) abstract (2) naziva sučelja
- E. (1) const (2) naziva razreda

Metode razreda, koje se deklariraju unutar razreda ključnom riječi (1), pozivaju se preko (2).

A. (1) static (2) naziva razreda

B. (1) static (2) reference na objekt

C. (1) abstract (2) naziva razreda

D. (1) abstract (2) naziva sučelja

E. (1) const (2) naziva razreda

Za objekt u UML-dijagramu objekata vrijedi tvrdnja:

- A. Objekti imaju definiciju metoda.
- B. Simbol objekta je pravokutnik s tri prelinca.
- C. Objekti imaju definiciju atributa.
- D. Objekti imaju vrijednost atributa.
- E. Objekti imaju označenu vidljivost atributa.

Za objekt u UML-dijagramu objekata vrijedi tvrdnja:

- A. Objekti imaju definiciju metoda.
- B. Simbol objekta je pravokutnik s tri prelinca.
- C. Objekti imaju definiciju atributa.
- D. Objekti imaju vrijednost atributa.**
- E. Objekti imaju označenu vidljivost atributa.

Liskovin princip zamjene (engl. Liskov substitution principle) kaže da ako se u varijablu tipa _____ stavi objekt tipa _____ , program se mora korektno izvoditi.

Liskovin princip zamjene (engl. Liskov substitution principle) kaže da ako se u varijablu tipa nadrazreda stavi objekt tipa podrazreda, program se mora korektno izvoditi.

Vrsta pridruživanja na UML dijagramima razreda kod koje se, ako je uništen objekt agregata, nužno uništavaju i dijelovi tog agregata, naziva se

_____.

Vrsta pridruživanja na UML dijagramima razreda kod koje se, ako je uništen objekt agregata, nužno uništavaju i dijelovi tog agregata, naziva se kompozicija.

Na UML-dijagramima obrazaca uporabe, veza _____ može se crtati između dva aktora kao i između dva obrasca uporabe.

Na UML-dijagramima obrazaca uporabe, veza generalizacije može se crtati između dva aktora kao i između dva obrasca uporabe.

ROUND 2

(MI 2016-A)



Uvođenjem modela u razvoj programske potpore nastoji se doskočiti prekomjernoj _____ programske potpore.

- A. Analizi
- B. Modularnosti
- C. Složenosti
- D. Ispitljivosti
- E. Koheziji

Uvođenjem modela u razvoj programske potpore nastoji se doskočiti prekomjernoj _____ programske potpore.

- A. Analizi
- B. Modularnosti
- C. Složenosti**
- D. Ispitljivosti
- E. Koheziji

Element provjere zahtjeva koji ustanovljuje je li naveden izvor dokumenta i koji su razlozi uvrštavanja zahtjeva naziva se:

- A. Konzistencija
- B. Valjanost
- C. Adaptibilnost
- D. Sljedivost
- E. Kompletnost

Element provjere zahtjeva koji ustanovljuje je li naveden izvor dokumenta i koji su razlozi uvrštavanja zahtjeva naziva se:

- A. Konzistencija
- B. Valjanost
- C. Adaptibilnost
- D. Sljedivost**
- E. Kompletnost

Dobar obrazac uporabe u inženjerstvu
zahtjeva prvenstveno specificira:

- A. Zahtjeve okoline.
- B. Mogućnosti razvojnog tima.
- C. Detaljne opise algoritma zahtjeva.
- D. Najvažnije funkcionalne zahtjeve.
- E. Sve navedeno.

Dobar obrazac uporabe u inženjerstvu
zahtjeva prvenstveno specificira:

- A. Zahtjeve okoline.
- B. Mogućnosti razvojnog tima.
- C. Detaljne opise algoritma zahtjeva.
- D. Najvažnije funkcionalne zahtjeve.**
- E. Sve navedeno.

Aktivnosti u pojedinim fazama Unificiranog procesa (engl. Unified process) imaju pridružene _____ koji se dokumentiraju dijagramima.

- A. Modele
- B. Iteracije
- C. Ključne točke
- D. Računalne resurse
- E. Arhitekturne stilove

Aktivnosti u pojedinim fazama Unificiranog procesa (engl. Unified process) imaju pridružene _____ koji se dokumentiraju dijagramima.

A. Modele

B. Iteracije

C. Ključne točke

D. Računalne resurse

E. Arhitekturne stilove

Koji je model procesa programskog inženjerstva najpogodniji za male i srednje interaktivne sustave?

- A. Komponentno usmjeren
- B. Vodopadni
- C. Evolucijski
- D. Unificirani proces
- E. Ad hoc model

Koji je model procesa programskog inženjerstva najpogodniji za male i srednje interaktivne sustave?

- A. Komponentno usmjeren
- B. Vodopadni
- C. Evolucijski**
- D. Unificirani proces
- E. Ad hoc model

Što je zajedničko svim pogledima kod modela arhitekture programske potpore "pogled 4+1" (engl. 4+1 view):

- A. Matematička logika
- B. Vremenski odziv
- C. Scenariji
- D. Stanja sustava
- E. Komponente

Što je zajedničko svim pogledima kod modela arhitekture programske potpore "pogled 4+1" (engl. 4+1 view):

A. Matematička logika

B. Vremenski odziv

C. Scenariji

D. Stanja sustava

E. Komponente

Koje se tri aktivnosti ponavljaju tijekom razvoja u evolucijskom modelu?

- A. Analiza, oblikovanje, implementacija
- B. Specifikacija, oblikovanje, ispitivanje
- C. Specifikacija, razvoj, validacija
- D. Analiza, implementacija, ispitivanje
- E. Specifikacija, validacija i verifikacija, evolucija

Koje se tri aktivnosti ponavljaju tijekom razvoja u evolucijskom modelu?

- A. Analiza, oblikovanje, implementacija
- B. Specifikacija, oblikovanje, ispitivanje
- C. Specifikacija, razvoj, validacija**
- D. Analiza, implementacija, ispitivanje
- E. Specifikacija, validacija i verifikacija, evolucija

Koji princip dobrog oblikovanja programske potpore se krši uporabom globalnih varijabli?

- A. Povećanje kohezije
- B. Smanjenje međuovisnosti
- C. Podijeli pa vladaj
- D. Oblikovanje za ispitivanje
- E. Planiranje zastare

Koji princip dobrog oblikovanja programske potpore se krši uporabom globalnih varijabli?

- A. Povećanje kohezije
- B. Smanjenje međuovisnosti**
- C. Podijeli pa vladaj
- D. Oblikovanje za ispitivanje
- E. Planiranje zastare

Veza poopćenja (engl. generalization) na dijagramu obrazaca uporabe može biti definirana između:

- A. Aktivnih aktora i obrazaca uporabe
- B. Obrazaca uporabe međusobno
- C. Pasivnih aktora i obrazaca uporabe
- D. Sekundarnih aktora i obrazaca uporabe
- E. Objekata i obrazaca uporabe

Veza poopćenja (engl. generalization) na diagramu obrazaca uporabe može biti definirana između:

- A. Aktivnih aktora i obrazaca uporabe
- B. Obrazaca uporabe međusobno**
- C. Pasivnih aktora i obrazaca uporabe
- D. Sekundarnih aktora i obrazaca uporabe
- E. Objekata i obrazaca uporabe

Naredbom `git add` inačica neke datoteke pohranjuje se u:

- A. korisničku mapu u kojoj se nalaze datoteke iz repozitorija (engl. workspace)
- B. lokalni korisnički repozitorij (engl. local repository)
- C. prostor pripreme za buduću pohranu u repozitorij (engl. staging area)
- D. središnji repozitorij (engl. remote repository)
- E. Ništa od navedenoga

Naredbom git add inačica neke datoteke pohranjuje se u:

- A. korisničku mapu u kojoj se nalaze datoteke iz repozitorija (engl. workspace)
- B. lokalni korisnički repozitorij (engl. local repository)
- C. prostor pripreme za buduću pohranu u repozitorij (engl. staging area)**
- D. središnji repozitorij (engl. remote repository)
- E. Ništa od navedenoga

U načine dokumentiranja zahtjeva sustava koji se koriste u praksi NE spada:

- A. UML dijagram obrazaca uporabe
- B. Intervjuiranje
- C. Izrada scenarija
- D. UML dijagram razreda
- E. UML sekvencijski dijagram

U načine dokumentiranja zahtjeva sustava koji se koriste u praksi NE spada:

- A. UML dijagram obrazaca uporabe
- B. Intervjuiranje
- C. Izrada scenarija
- D. UML dijagram razreda**
- E. UML sekvencijski dijagram

Prilikom izlučivanja i analize zahtjeva, koriste se UML dijagrami obrazaca uporabe, uz koje se u projektnoj dokumentaciji uobičajeno navodi i detaljan opis obrazaca. Navedite najmanje 5 elemenata koje se pišu u okviru detaljnog opisa.

Moguća rješenja:

- opis početne situacije (ili preduvjeti).
- cilj obrasca uporabe (scenarija)
- opis stanja gdje scenarij završava (ili rezultat).
- opis normalnog/standardnog tijeka događaja.
- opis što se eventualno može dogoditi krivo.
- informacija o paralelnim aktivnostima.
- popis sudionika...

U objektno usmjerenim sustavima operacije su neovisne o kodu i ostvaruju _____ pojedinih razreda

U objektno usmjerenim sustavima operacije su neovisne o kodu i ostvaruju odgovornosti pojedinih razreda

ROUND 3

(MI 2016-B)



Element provjere zahtjeva koji ustanovljuje da li sustav osigurava funkcije koje podupiru potrebe korisnika naziva se:

- A. Konzistencija
- B. Valjanost
- C. Sljedivost
- D. Adaptabilnost
- E. Kompletnost

Element provjere zahtjeva koji ustanovljuje da li sustav osigurava funkcije koje podupiru potrebe korisnika naziva se:

A. Konzistencija

B. Valjanost

C. Sljedivost

D. Adaptabilnost

E. Kompletnost

Uvođenjem modela u razvoj programske potpore nastoji se doskočiti prekomjernoj _____ programske potpore.

- A. Analizi
- B. Modularnosti
- C. Složenosti
- D. Ispitljivosti
- E. Koheziji

Uvođenjem modela u razvoj programske potpore nastoji se doskočiti prekomjernoj _____ programske potpore.

- A. Analizi
- B. Modularnosti
- C. Složenosti**
- D. Ispitljivosti
- E. Koheziji

U procesu inženjerstva zahtjeva faza analize
zahtjeva rezultira:

- A. Algoritmima i podatkovnim strukturama problema
- B. Procjenom korisnosti i pouzdanosti
- C. Funkcionalnim i ponašajnim modelom
- D. Procjenom troškova
- E. Arhitekturom i strukturom programa

U procesu inženjerstva zahtjeva faza analize zahtjeva rezultira:

- A. Algoritmima i podatkovnim strukturama problema
- B. Procjenom korisnosti i pouzdanosti
- C. Funkcionalnim i ponašajnim modelom**
- D. Procjenom troškova
- E. Arhitekturom i strukturom programa

U kojem od navedenih modela je najizraženije sudjelovanje korisnika u razvojn timeru:

- A. Vodopadnom modelu (engl. Waterfall model)
- B. Prototipnom modelu
- C. Ubrzani razvoj
- D. Unificiranom procesu
- E. Ni jednom od navedenih

U kojem od navedenih modela je najizraženije sudjelovanje korisnika u razvojnom timu:

- A. Vodopadnom modelu (engl. Waterfall model)
- B. Prototipnom modelu
- C. Ubrzani razvoj**
- D. Unificiranom procesu
- E. Ni jednom od navedenih

Aktivnosti u pojedinim fazama Unificiranog procesa (engl. Unified process) imaju pridružene _____ koji se dokumentiraju dijagramima.

- A. Modele
- B. Iteracije
- C. Ključne točke
- D. Računalne resurse
- E. Arhitekturne stilove

Aktivnosti u pojedinim fazama Unificiranog procesa (engl. Unified process) imaju pridružene _____ koji se dokumentiraju dijagramima.

A. Modele

B. Iteracije

C. Ključne točke

D. Računalne resurse

E. Arhitekturne stilove

Kod inkrementalnog pristupa iteracijama vrijedi tvrdnja da se s početkom razvoja pojedinog inkrementa njegovi zahtjevi fiksiraju.

A. Točno

B. Netočno

Kod inkrementalnog pristupa iteracijama vrijedi tvrdnja da se s početkom razvoja pojedinog inkrementa njegovi zahtjevi fiksiraju.

A. Točno

B. Netočno

UML dijagrami razreda (engl. class diagram)
čine temelj opisa:

- A. Funkcionalnih zahtjeva
- B. Nefunkcionalnih zahtjeva
- C. Zahtjeva domene primjene
- D. Arhitekture sustava
- E. Održavanja

UML dijagrami razreda (engl. class diagram)
čine temelj opisa:

- A. Funkcionalnih zahtjeva
- B. Nefunkcionalnih zahtjeva
- C. Zahtjeva domene primjene
- D. Arhitekture sustava**
- E. Održavanja

Princip oblikovanja programske potpore kod kojeg se, između ostalog, izbjegava uporaba najnovijih tehnologija kao i rijetko upotrebljavanih dijelova knjižnica naziva se:

- A. Oblikuj po ugovoru
- B. Oblikuj za fleksibilnost
- C. Planiraj zastaru
- D. Oblikuj za prenosivost
- E. Oblikuj konzervativno

Princip oblikovanja programske potpore kod kojeg se, između ostalog, izbjegava uporaba najnovijih tehnologija kao i rijetko upotrebljavanih dijelova knjižnica naziva se:

- A. Oblikuj po ugovoru
- B. Oblikuj za fleksibilnost
- C. Planiraj zastaru**
- D. Oblikuj za prenosivost
- E. Oblikuj konzervativno

Koja od navedenih tvrdnji vrijedi za vezu ovisnosti (engl. dependency) na dijagramu razreda:

- A. Veza je uvijek jednosmjerna.
- B. Veza je uvijek dvosmjerna.
- C. Veza predstavlja odnos dio-cjelina između dva razreda.
- D. Moguće je refleksivno pridruživanje.
- E. Veza ovisnosti jaka je inačica veze agregacije.

Koja od navedenih tvrdnji vrijedi za vezu ovisnosti (engl. dependency) na dijagramu razreda:

- A. Veza je uvijek jednosmjerna.**
- B. Veza je uvijek dvosmjerna.
- C. Veza predstavlja odnos dio-cjelina između dva razreda.
- D. Moguće je refleksivno pridruživanje.
- E. Veza ovisnosti jaka je inačica veze agregacije.

Koja od tvrdnji (A-D) **ne vrijedi** (ili je istinita E) za izvorni kod generiran automatski na temelju UML dijagrama razreda u alatu Astah Professional?

- A. Za svaki razred se generira zasebna datoteka.
- B. Izvorni kod svakog razreda sadrži deklaracije atributa tog razreda.
- C. Izvorni kod razreda sadrži prazno tijelo svih metoda tog razreda.
- D. Izvorni kod svakog razreda obavezno sadrži konstruktor razreda.
- E. Sve tvrdnje vrijede.

Koja od tvrdnji (A-D) **ne vrijedi** (ili je istinita E) za izvorni kod generiran automatski na temelju UML dijagrama razreda u alatu Astah Professional?

- A. Za svaki razred se generira zasebna datoteka.
- B. Izvorni kod svakog razreda sadrži deklaracije atributa tog razreda.
- C. Izvorni kod razreda sadrži prazno tijelo svih metoda tog razreda.
- D. Izvorni kod svakog razreda obavezno sadrži konstruktor razreda.**
- E. Sve tvrdnje vrijede.

Za moderne sustave za upravljanje inačicama datoteka razvoja programske potpore ne vrijedi tvrdnja:

- A. Poželjna dobra integracija s razvojnom okolinom
- B. Poželjna mogućnost stvaranja grana razvoja.
- C. Poželjna mogućnost spajanje grana razvoja
- D. Poželjan raspodijeljeni rad
- E. Poželjan rad na isključivo jednoj kopiji datoteke radi održavanja jednoznačnosti.

Za moderne sustave za upravljanje inačicama datoteka razvoja programske potpore ne vrijedi tvrdnja:

- A. Poželjna dobra integracija s razvojnom okolinom
- B. Poželjna mogućnost stvaranja grana razvoja.
- C. Poželjna mogućnost spajanje grana razvoja
- D. Poželjan raspodijeljeni rad
- E. Poželjan rad na isključivo jednoj kopiji datoteke radi održavanja jednoznačnosti.**

Navedite generičke aktivnosti procesa programskog inženjerstva:

Navedite generičke aktivnosti procesa programskog inženjerstva:

specifikacija, razvoj i oblikovanje, validacija, evolucija

U objektno usmjerenim sustavima operacije se implementiraju _____.

U objektno usmjerenim sustavima operacije se implementiraju metodama.

2017

GRUPA A

Kupac, korisnik,
programer i rukovoditelj
razvoja primjeri su

programskog proizvoda.

Kupac, korisnik,
programer i rukovoditelj
razvoja primjeri su
dionika
programskog proizvoda.

S obzirom na sadržaj,
zahtjeve klasificiramo na:

_____,
_____,
_____.

S obzirom na sadržaj,
zahtjeve klasificiramo na:
funkcionalne,
nefunkcionalne i
zahtjeve domene primjene.

Navedite barem tri načina
izražavanja zahtjeva
sustava u inženjerstvu
zahtjeva:

Navedite barem tri načina izražavanja
zahtjeva sustava u inženjerstvu zahtjeva:

strukturiranim prirodnim jezikom,
jezikom za opis oblikovanja,
grafičkim jezikom (npr. UML),
matematičkom (formalnom) notacijom

Sadržaj scenarija uobičajeno obuhvaća:

- opis početne situacije,
- opis normalnog/standardnog tijeka događaja,
- ---
- informaciju o paralelnim aktivnostima,
- opis stanja gdje scenarij završava.

Sadržaj scenarija uobičajeno obuhvaća:

- opis početne situacije,
- opis normalnog/standardnog tijeka događaja,
- opis što se eventualno može dogoditi krivo (ili opis grešaka ili opis ostalih mogućnosti rada ili opis odstupanja)
- informaciju o paralelnim aktivnostima,
- opis stanja gdje scenarij završava.

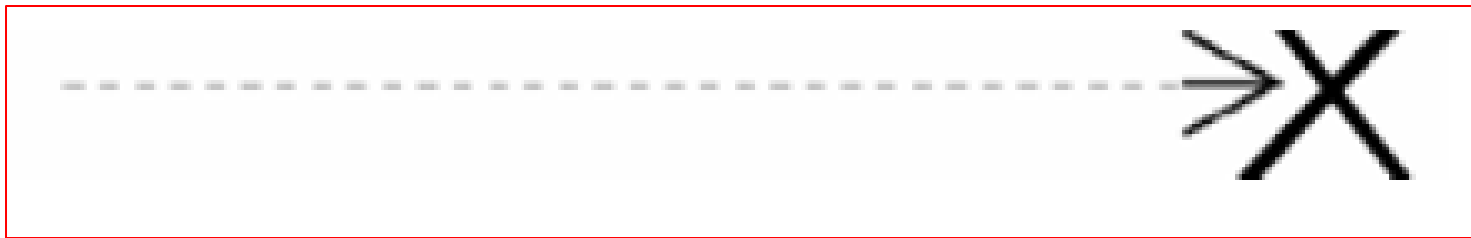
Kojim tipom veza su
aktori povezani s
obrascima uporabe?

Kojim tipom veza su akteri
povezani s obrascima uporabe?

pridruživanja (asocijacije)

Nacrtajte grafičku oznaku uništavanja (engl. *destroy*) objekta sekvencijskog dijagrama

Nacrtajte grafičku oznaku uništavanja (engl. *destroy*) objekta sekvencijskog dijagrama



Kod unificiranog procesa razvoja programske potpore, dinamička perspektiva ostvaruje se putem četiri _____, koje se sastoje od jedne ili više _____, a koje rezultiraju _____.

Kod unificiranog procesa razvoja programske potpore, dinamička perspektiva ostvaruje se putem četiri faze, koje se sastoje od jedne ili više iteracija, a koje rezultiraju artefaktima (može se reći i dokumentima ili izdanjima koda).

Vremenski ograničeni period od 30 dana ili manje, tijekom kojeg se proizvede gotov i potencijalno isporučiv inkrement proizvoda u agilnom pristupu Scrum naziva se _____.

Vremenski ograničeni period od 30 dana ili manje, tijekom kojeg se proizvede gotov i potencijalno isporučiv inkrement proizvoda u agilnom pristupu Scrum naziva se sprint.

Princip oblikovanja objektno
usmjerene arhitekture
programske potpore *podijeli pa
vlada* (engl. *divide and
conquer*) osigurava



Princip oblikovanja objektno
usmjerene arhitekture
programske potpore *podijeli pa
vlada* (engl. *divide and
conquer*) osigurava

jednostavniji rad s više manjih
djelova.

U objektno usmjerenom oblikovanju pri razradi dijagrama razreda dio vidljivog ponašanja skupa objekata opisuje se

U objektno usmjerenom oblikovanju pri razradi dijagrama razreda dio vidljivog ponašanja skupa objekata opisuje se

sučeljima, engl. interfaces

Liskovin princip zamjene
(engl. *Liskov substitution principle*)
kaže da ako se objekt tipa
_____ postavi u
varijablu tipa _____
_____, program se
mora korektno izvoditi.

Liskovin princip zamjene
(engl. *Liskov substitution principle*)
kaže da ako se objekt tipa
 nadrazreda postavi u
varijablu tipa
 podrazreda , program
se mora korektno izvoditi.

U procesu oblikovanja
arhitekture programske
potpore strukturiranje
sustava obuhvaća definiciju
podsustava i mehanizama

U procesu oblikovanja
arhitekture programske
potpore strukturiranje
sustava obuhvaća definiciju
podsustava i mehanizama
komunikacije .

U suradničkom alatu GIT
novi sadržaji iz udaljenog
repozitorija povlače se u
_____ repozitorij
naredbom _____.

U suradničkom alatu GIT novi
sadržaji iz udaljenog
repozitorija povlače se u
lokalni repozitorij
naredbom
fetch (priznaje se i pull).

2017

GRUPA B

Programsko inženjerstvo
bavi se

_____ i

pristupom procesu izrade
programske potpore.

Programsko inženjerstvo
bavi se

systematskim i

organiziranim

pristupom procesu izrade
programske potpore.

Navedite barem tri načina
izražavanja zahtjeva
sustava u inženjerstvu
zahtjeva:

Navedite barem tri načina izražavanja
zahtjeva sustava u inženjerstvu zahtjeva:

strukturiranim prirodnim jezikom,
jezikom za opis oblikovanja,
grafičkim jezikom (npr. UML),
matematičkom (formalnom) notacijom

S obzirom na sadržaj,
zahtjeve klasificiramo na:

_____,
_____,
_____.

S obzirom na sadržaj,
zahtjeve klasificiramo na:
funkcionalne,
nefunkcionalne i
zahtjeve domene primjene.

Kojim tipom veza su
aktori povezani s
obrascima uporabe?

Kojim tipom veza su akteri
povezani s obrascima uporabe?

pridruživanja (asocijacije)

Sadržaj scenarija uobičajeno obuhvaća:

- opis početne situacije,
- opis normalnog/standardnog tijeka događaja,
- ---
- informaciju o paralelnim aktivnostima,
- opis stanja gdje scenarij završava.

Sadržaj scenarija uobičajeno obuhvaća:

- opis početne situacije,
- opis normalnog/standardnog tijeka događaja,
- opis što se eventualno može dogoditi krivo (ili opis grešaka ili opis ostalih mogućnosti rada ili opis odstupanja)
- informaciju o paralelnim aktivnostima,
- opis stanja gdje scenarij završava.

Nacrtajte standardnu grafičku oznaku povratne poruke (engl. *reply message*) UML sekvencijskog dijagrama.

Nacrtajte standardnu grafičku oznaku povratne poruke (engl. *reply message*) UML sekvencijskog dijagrama.



U agilnom pristupu razvoja
programske potpore zasnovanom na
radnom okviru Scrum
osobu koja je jedina odgovorna za
upravljanje projektnim dnevnikom
zaostataka (engl. *Product Backlog*)
nazivamo

U agilnom pristupu razvoja programske potpore zasnovanom na radnom okviru Scrum

osobu koja je jedina odgovorna za upravljanje projektnim dnevnikom zaostataka (engl. *Product Backlog*) nazivamo

Vlasnik proizvoda (engl. Product Owner) .

Kod unificiranog procesa razvoja programske potpore, dinamička perspektiva ostvaruje se putem četiri _____, koje se sastoje od jedne ili više _____, a koje rezultiraju _____.

Kod unificiranog procesa razvoja programske potpore, dinamička perspektiva ostvaruje se putem četiri faze, koje se sastoje od jedne ili više iteracija, a koje rezultiraju artefaktima (može se reći i dokumentima ili izdanjima koda).

Princip oblikovanja objektno usmjerene
arhitekture programske potpore koji
potpomaže jasno raspoznati rad
komponente te ujedno smanjuje utjecaj
promjena te
komponente na ostale povezane
nazivamo

Princip oblikovanja objektno usmjerene arhitekture programske potpore koji potpomaže jasno raspoznati rad komponente te ujedno smanjuje utjecaj promjena te komponente na ostale povezane nazivamo

Smanji međuovisnost - engl. *Reduce coupling where possible*.

U objektno usmjerenom oblikovanju pri razradi dijagrama razreda dio vidljivog ponašanja skupa objekata opisuje se

U objektno usmjerenom oblikovanju pri razradi dijagrama razreda dio vidljivog ponašanja skupa objekata opisuje se

sučeljima, engl. interfaces

Liskovin princip zamjene
(engl. *Liskov substitution principle*)
kaže da ako se objekt tipa
_____ postavi u
varijablu tipa _____
_____, program se
mora korektno izvoditi.

Liskovin princip zamjene
(engl. *Liskov substitution principle*)
kaže da ako se objekt tipa
~~podrazreda~~ postavi u
varijablu tipa
~~nadrazreda~~, program se
mora korektno izvoditi.

U procesu oblikovanja
arhitekture programske
potpore strukturiranje
sustava obuhvaća definiciju
podsustava i mehanizama

U procesu oblikovanja
arhitekture programske
potpore strukturiranje
sustava obuhvaća definiciju
podsustava i mehanizama
komunikacije .

U suradničkom alatu GIT
novi sadržaji iz udaljenog
repozitorija povlače se u
_____ repozitorij
naredbom _____.

U suradničkom alatu GIT novi
sadržaji iz udaljenog
repozitorija povlače se u
lokalni repozitorij
naredbom
fetch (priznaje se i pull).

2018

GRUPA B

Propisani strukturirani postupci,
dokumentiranje, _____,
višestruka uporaba
komponenata i formalna
verifikacija modela i ispitivanje
elementi su modernog postupka
oblikovanja sustava.

Propisani strukturirani postupci, dokumentiranje, modeliranje (apstrakcija), višestruka uporaba komponenata i formalna verifikacija modela i ispitivanje elementi su modernog postupka oblikovanja sustava.

Za svojstvo dobre
dokumentacije
programske potpore
vrijedi da



Za svojstvo dobre dokumentacije programske potpore vrijedi da doseže potrebe i ciljeve ključnih dionika.

Problemi zahtjeva
domene primjene su:

Problemi zahtjeva
domene primjene su:

razumljivost i
implicitnost.

Metoda kvalitativnog promatranja i opisivanja ponašanja ljudi u društvu, kod koje su zahtjevi izvedeni temeljem istraživanja kako ljudi stvarno rade naziva se

Metoda kvalitativnog promatranja i opisivanja ponašanja ljudi u društvu, kod koje su zahtjevi izvedeni temeljem istraživanja kako ljudi stvarno rade naziva se etnografija.

Kod UML dijagrama
obrazaca uporabe, vezu

možemo koristiti između
dva aktora kao i između
dva obrasca uporabe.

Kod UML dijagrama
obrazaca uporabe, vezu
generalizacije
možemo koristiti između
dva aktora kao i između
dva obrasca uporabe.

Kod UML sekvencijskih dijagrama, crtkanom (isprekidanom) linijom sa strelicom na jednom kraju prikazujemo poruku.

Kod UML sekvencijskih
dijagrama, crtkanom
(isprekidanom) linijom
sa strelicom na jednom
kraju prikazujemo
povratnu poruku.

Kod unificiranog procesa
razvoja programske potpore,
jezgrena aktivnost
implementacije programske
potpore provodi se u najvećoj
mjeri tijekom faze

Kod unificiranog procesa
razvoja programske potpore,
jezgrena aktivnost
implementacije programske
potpore provodi se u najvećoj
mjeri tijekom faze
izgradnje (engl. construction)

Vlasnik proizvoda,
razvojni tim i

ține Scrum tim.

Vlasnik proizvoda,
razvojni tim i
Scrum vođa
(engl. Scrum master)
čine Scrum tim.

Koncepcijska arhitektura (engl.
Conceptual Architecture)
programske potpore usmjerena
je na pogodnu dekompoziciju
sustava i komunikaciju s

Koncepcijska arhitektura (engl. *Conceptual Architecture*) programske potpore usmjerena je na pogodnu dekompoziciju sustava i komunikaciju s

netehničkim dionicima (uprava, prodaja, korisnici).

Za koji princip oblikovanja objektno usmjerene arhitekture programske potpore jedna od karakteristika je grupiranje modula koji pridonose jednoj dobro definiranoj funkciji?

Za koji princip oblikovanja objektno usmjerene arhitekture programske potpore jedna od karakteristika je grupiranje modula koji pridonose jednoj dobro definiranoj funkciji?

Povećanje kohezije

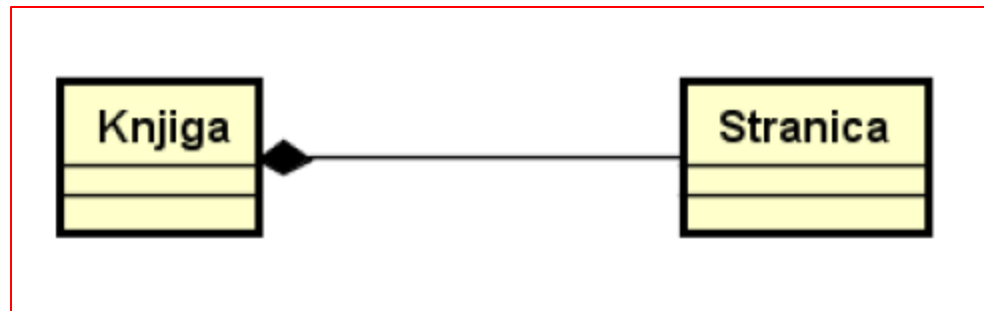
(engl Increase cohesion where possible)

Dobri dokumenti oblikovanja
arhitekture, kao pomoć izradi bolje
programske potpore, pišu se s
gledišta _____ i služe kao
sredstvo komunikacije

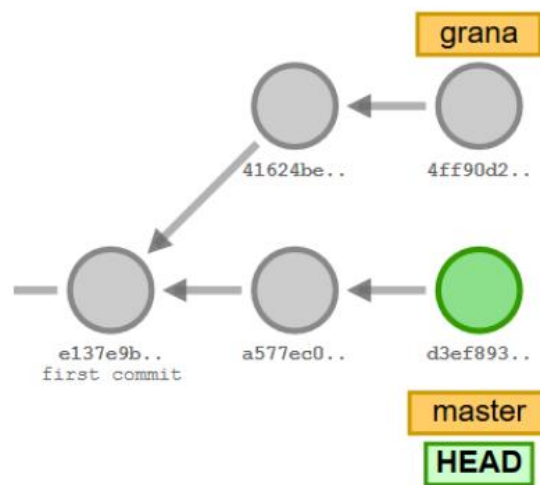
Dobri dokumenti oblikovanja
arhitekture, kao pomoć izradi bolje
programske potpore, pišu se s gledišta
čitatelja i služe kao sredstvo
komunikacije
tima za implementaciju/budućih osoba
zaduženih za održavanje, razvoj ili
promjene/osobama uključenih u
oblikovanje povezanih sustava ili
pod sustava.

Nacrtajte grafičku oznaku sadržavanja UML dijagrama razreda kojom se naglašava ovisnost razreda o životnom ciklusu drugog razreda i prikažite primjer dva razreda u takvoj vezi.

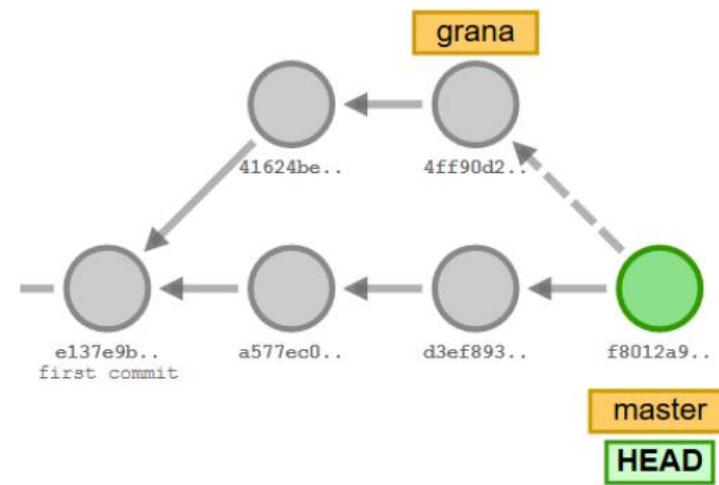
Nacrtajte grafičku oznaku sadržavanja UML dijagrama razreda kojom se naglašava ovisnost razreda o životnom ciklusu drugog razreda i prikažite primjer dva razreda u takvoj vezi.



Kojom Git naredbom repozitorij pohranjuje promjene iz grane „grana” u granu „master”, tj. prelazi iz stanja sa slike a) u stanje sa slike b)?



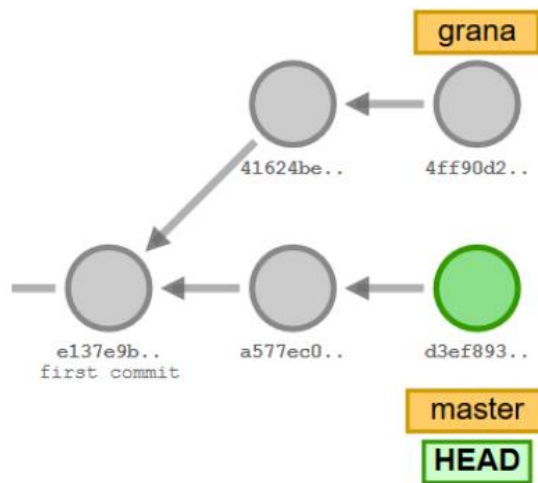
slika a)



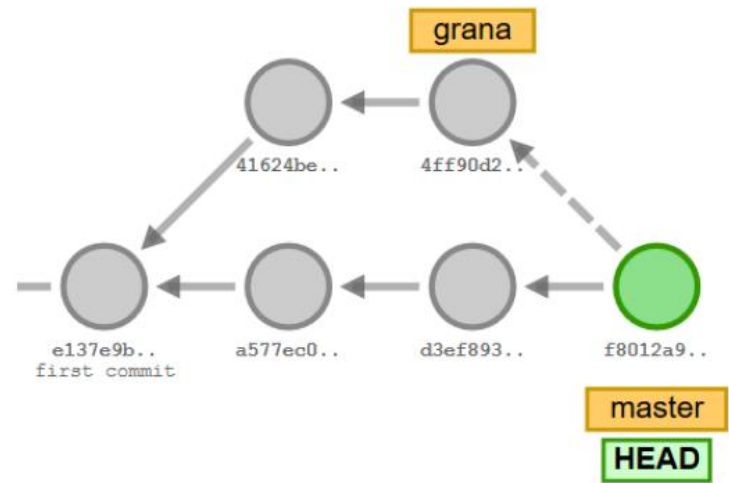
slika b)

Kojom Git naredbom repozitorij pohranjuje promjene iz grane „grana” u granu „master”, tj. prelazi iz stanja sa slike a) u stanje sa slike b)?

git merge grana




slika a)




slika b)

PROGI-2013,2014,2015-A



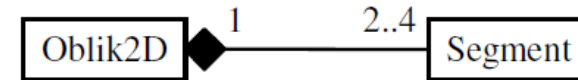
1. (1 bod) Za spiralni model inženjerstva zahtjeva **ne** vrijedi:

- A. Koriste se iteracije ciklusa svih generičkih aktivnosti.
 - B. U kasnijim iteracijama izrađuju se prototipovi radi validacije.
 - C. Zahtjevi sustava određuju se u ranijim iteracijama.
 - D. Rane iteracije imaju fokus na poslovnom modelu zahtjeva.
 - E. Zahtjevi se kroz iteracije specificiraju s različitom razinom detalja.
- 

1. (1 bod) Za spiralni model inženjerstva zahtjeva **ne** vrijedi:

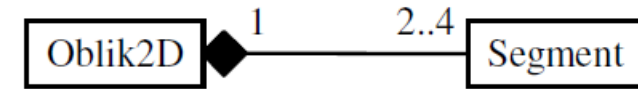
- ☒ A. Koriste se iteracije ciklusa svih generičkih aktivnosti.
- ☐ B. U kasnijim iteracijama izrađuju se prototipovi radi validacije.
- ☐ C. Zahtjevi sustava određuju se u ranijim iteracijama.
- ☐ D. Rane iteracije imaju fokus na poslovnom modelu zahtjeva.
- ☐ E. Zahtjevi se kroz iteracije specificiraju s različitom razinom detalja.

10. (1 bod) Što definira veza prikazana na UML dijagramu:



- A. U sustavu se u bilo kojem trenutku nalazi najviše jedan objekt instanciran iz razreda `Oblik2D`,
- B. U sustavu se u bilo kojem trenutku može nalaziti samostalno od dva do četiri objekata nastalih iz razreda `Segment`.
- C. Ako se obriše jedan objekt razreda `Segment`, mora se obrisati i pripadajući objekt razreda `Oblik2D`.
- D. Svaki objekt razreda `Segment` nasljeđuje attribute razreda `Oblik2D`.
- E. Svaki objekt razreda `Oblik2D` je povezan s dva do četiri objekta razreda `Segment` (nakon inicijalizacije).

10. (1 bod) Što definira veza prikazana na UML dijagramu:

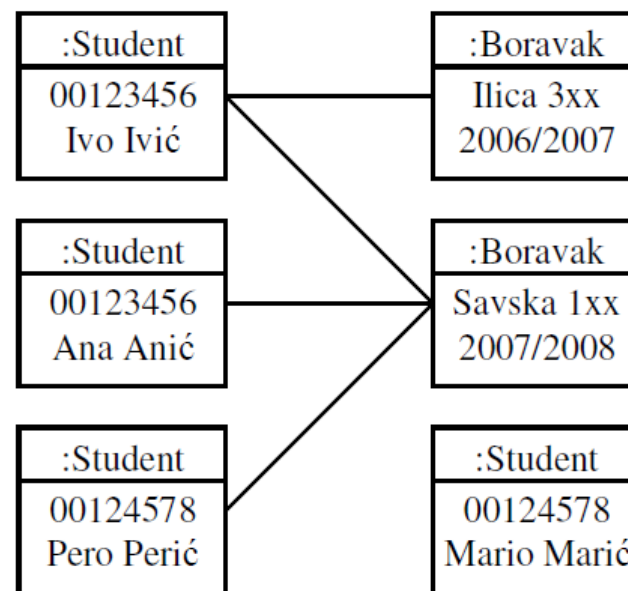


- A. U sustavu se u bilo kojem trenutku nalazi najviše jedan objekt instanciran iz razreda `Oblik2D`,
- B. U sustavu se u bilo kojem trenutku može nalaziti samostalno od dva do četiri objekata nastalih iz razreda `Segment`.
- C. Ako se obriše jedan objekt razreda `Segment`, mora se obrisati i pripadajući objekt razreda `Oblik2D`.**
- D. Svaki objekt razreda `Segment` nasljeđuje attribute razreda `Oblik2D`.
- E. Svaki objekt razreda `Oblik2D` je povezan s dva do četiri objekta razreda `Segment` (nakon inicijalizacije).

11. (1 bod) Dijagram objekata na slici nastao je iz dijagrama razreda. Kakva je brojnost (višestrukost) u dijagramu razreda [Student]---[Boravak] ?

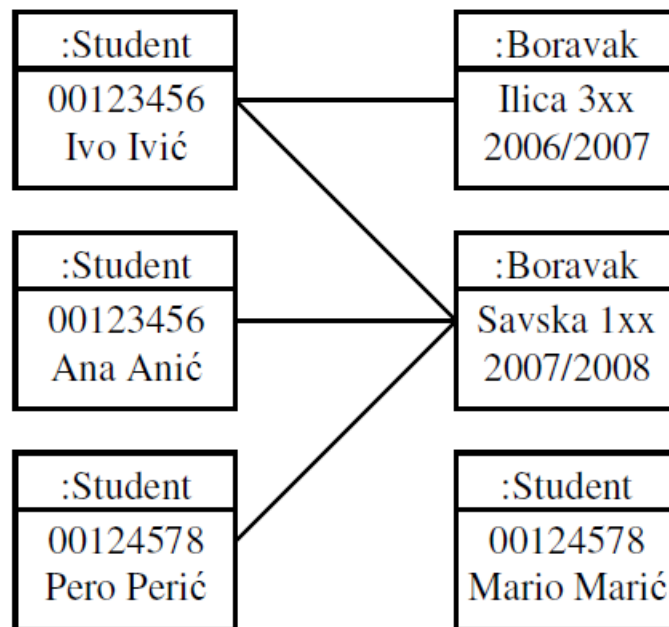
Svaka višestrukost veza prikazana na dijagramu vrijedi i općenitije, tj. ako npr. vrijedi 3---x tada vrijedi i N---x.).


- A. 1---1
- B. *---1
- C. 1..N---1..N
- D. 1..N---*
- E. N---N




11. (1 bod) Dijagram objekata na slici nastao je iz dijagrama razreda. Kakva je brojnost (višestrukost) u dijagramu razreda [Student]---[Boravak] ?
Svaka višestrukost veza prikazana na dijagramu vrijedi i općenitije, tj. ako npr. vrijedi 3---x tada vrijedi i N---x.).

- A. 1---1
- B. *---1
- C. 1..N---1..N
- D. 1..N---***
- E. N---N







13. (1 bod) U okviru oblikovanja UML dijagrama razreda potrebno je odrediti pridruživanja (engl. *association*) između razreda. Pri tome **najviše** valja koristiti:

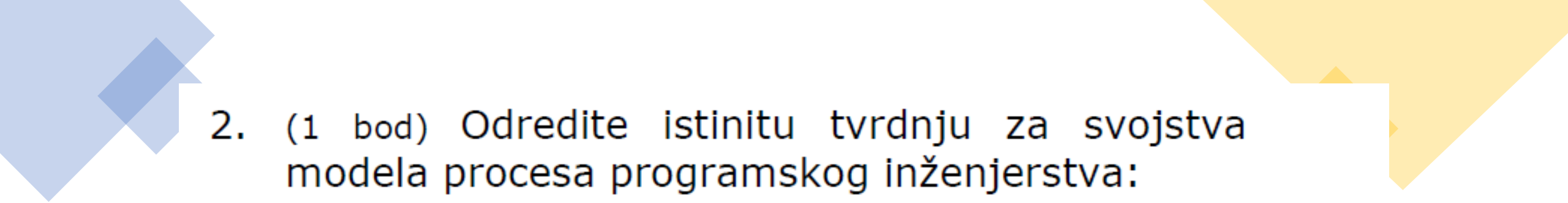

- A) UML dijagrame obrazaca uporabe (engl. *Use Case*).
 - B) UML sekvencijske dijagrame.
 - C) Imenice i imeničke izraze u specifikaciji programske potpore.
 - D) Akcije i navede ih se kao pridruživanja.
 - E) Funkcionalne zahtjeve temeljem analize zahtjeva.
- 

13. (1 bod) U okviru oblikovanja UML dijagrama razreda potrebno je odrediti pridruživanja (engl. *association*) između razreda. Pri tome **najviše** valja koristiti:



- A) UML dijagrame obrazaca uporabe (engl. *Use Case*).
- B) UML sekvencijske dijagrame.
- ☒ C) Imenice i imeničke izraze u specifikaciji programske potpore.
- D) Akcije i navede ih se kao pridruživanja.
- E) Funkcionalne zahtjeve temeljem analize zahtjeva.

- 
1. (1 bod) Kao dioniku u ocjeni kakvoće programskog produkta kupcu je najvažnije:
 - A. Lakoća oblikovanja.
 - B. Lakoća održavanja.
 - C. Lakoća ponovne uporabe dijelova.
 - D. Nijedno od navedenog.
 - E. Sve navedeno.
- 




1. (1 bod) Kao dioniku u ocjeni kakvoće programskog produkta kupcu je najvažnije:
- A. Lakoća oblikovanja.
 - B. Lakoća održavanja.
 - C. Lakoća ponovne uporabe dijelova.
 - ☒ D. Nijedno od navedenog.
 - E. Sve navedeno.

- 
2. (1 bod) Odredite istinitu tvrdnju za svojstva modela procesa programskog inženjerstva:
- A. Iteracije su primjenjive samo na unificirani procesni model.
 - B. Najmanje novog kôda generira se u modelu zasnovanom na komponentama.
 - C. RUP (engl. Rational Unified Process) je najstariji model.
 - D. Implementacija promjena je najlakša u vodopadnom modelu.
 - E. Evolucijski model generira najbolju strukturu.
- 

2. (1 bod) Odredite istinitu tvrdnju za svojstva modela procesa programskog inženjerstva:
- A. Iteracije su primjenjive samo na unificirani procesni model.
 - ☒ B. Najmanje novog kôda generira se u modelu zasnovanom na komponentama.
 - C. RUP (engl. Rational Unified Process) je najstariji model.
 - D. Implementacija promjena je najlakša u vodopadnom modelu.
 - E. Evolucijski model generira najbolju strukturu.

- 
3. (1 bod) Koje se tri aktivnosti ponavljaju tijekom razvoja u evolucijskom modelu?
- A. Specifikacija, oblikovanje, ispitivanje.
 - B. Specifikacija, razvoj, validacija.
 - C. Analiza, oblikovanje, implementacija.
 - D. Analiza, implementacija, ispitivanje.
- 

3. (1 bod) Koje se tri aktivnosti ponavljaju tijekom razvoja u evolucijskom modelu?
- A. Specifikacija, oblikovanje, ispitivanje.
 - ☒ B. Specifikacija, razvoj, validacija.
 - C. Analiza, oblikovanje, implementacija.
 - D. Analiza, implementacija, ispitivanje.

- 
- 
4. (1 bod) Dobar obrazac uporabe u inženjerstvu zahtjeva prvenstveno specificira:
- A. Zahtjeve okoline.
 - B. Mogućnosti razvojnog tima.
 - C. Detaljne opise algoritma zahtjeva.
 - D. Najvažnije funkcionalne zahtjeve.
 - E. Sve navedeno.
- 

4. (1 bod) Dobar obrazac uporabe u inženjerstvu zahtjeva prvenstveno specificira:
- A. Zahtjeve okoline.
 - B. Mogućnosti razvojnog tima.
 - C. Detaljne opise algoritma zahtjeva.
 - ☒ D. Najvažnije funkcionalne zahtjeve.
 - E. Sve navedeno.

5. (1 bod) Za sekvencijske dijagrame vrijede sljedeće tvrdnje (moguće više točnih odgovora):

- A. Prikazuju stanja objekata.
- B. Dobri su za grafički prikaz složenih algoritama.
- C. Dobri su za praćenje stanja sustava.



☒ D. Prikazuju interakciju u sustavu.

☒ E. Dobri su za prikaz komunikacije među objektima.

6. (1 bod) Koje od navedenih tvrdnji vrijede za odnos agregacije u UML dijagramu (moguće više točnih odgovora):
- A. Agregat predstavlja "je dio od" IsPartOf odnos.
 - B. Kad se uništi agregat, uništavaju se i svi njegovi dijelovi.
 - C. Agregacija je slabija od kompozicije.
 - D. Agregat predstavlja „je“ IS-A odnos.

6. (1 bod) Koje od navedenih tvrdnji vrijede za odnos agregacije u UML dijagramu (moguće više točnih odgovora):

- ☒ A. Agregat predstavlja "je dio od" IsPartOf odnos.
- ☐ B. Kad se uništi agregat, uništavaju se i svi njegovi dijelovi.
- ☒ C. Agregacija je slabija od kompozicije.
- ☐ D. Agregat predstavlja „je“ IS-A odnos.

- 
7. (1 bod) Za princip oblikovanja koji preporuča zadržavanje razine apstrakcije (engl. *keep the level of abstraction as high as possible*) vrijedi:
- A. Traži poznavanje svih detalja podsustava.
 - B. Povećava složenost prikaza problema.
 - C. Nije pogodan za ciljanu implementaciju u objektnoj okolini.
 - D. Osigurava odgodu razmatranja detalja.
 - E. Nijedno od navedenog.
- 

7. (1 bod) Za princip oblikovanja koji preporuča zadržavanje razine apstrakcije (engl. *keep the level of abstraction as high as possible*) vrijedi:
- A. Traži poznavanje svih detalja podsustava.
 - B. Povećava složenost prikaza problema.
 - C. Nije pogodan za ciljanu implementaciju u objektnoj okolini.
 - ☒ D. Osigurava odgodu razmatranja detalja.
 - E. Nijedno od navedenog.

9. (1 bod) Što definira veza prikazana na UML dijagramu:




- A. U sustavu se u bilo kojem trenutku nalazi najviše dva objekta instancirana iz razreda Zgrada.
- B. U sustavu se u bilo kojem trenutku može nalaziti od dva do četiri objekata nastalih iz razreda Stan bez objekta razreda Zgrada.
- C. Ako se obriše jedan objekt razreda Stan, mora se obrisati i pripadajući objekt razreda Zgrada.
- D. Svaki objekt razreda Stan nasljeđuje atribute razreda Zgrada.
- E. Svaki objekt razreda Zgrada je povezan s dva do četiri objekta razreda Stan (nakon inicijalizacije).


9. (1 bod) Što definira veza prikazana na UML dijagramu:



- A. U sustavu se u bilo kojem trenutku nalazi najviše dva objekta instancirana iz razreda Zgrada.
- B. U sustavu se u bilo kojem trenutku može nalaziti od dva do četiri objekata nastalih iz razreda Stan bez objekta razreda Zgrada.
- C. Ako se obriše jedan objekt razreda Stan, mora se obrisati i pripadajući objekt razreda Zgrada.**
- D. Svaki objekt razreda Stan nasljeđuje attribute razreda Zgrada.
- E. Svaki objekt razreda Zgrada je povezan s dva do četiri objekta razreda Stan (nakon inicijalizacije).



10. (1 bod) Ako razred nasljeđuje apstraktni razred, onda mora implementirati sve njegove apstraktne metode da bi postao konkretan.

- A. Točno.
 - B. Netočno.
 - C. Ako ima više metoda, onda mora implementirati barem jednu.
- 

10. (1 bod) Ako razred nasljeđuje apstraktni razred, onda mora implementirati sve njegove apstraktne metode da bi postao konkretan.

A. Točno.

☒ B. Netočno.

C. Ako ima više metoda, onda mora implementirati barem jednu.

12. (1 bod) Navedite generičke aktivnosti procesa programskog inženjerstva:

A. _____

B. _____

C. _____

D. _____

12. (1 bod) Navedite generičke aktivnosti procesa programskog inženjerstva:



- A. Specifikacija
- B. Oblikovanje i implementacija
- C. Validacija i verifikacija
- D. Evolucija

13. (1 bod) Prilikom izlučivanja i analize zahtjeva, koriste se UML dijagrami obrazaca uporabe, uz koje se u projektnoj dokumentaciji obično navodi i detaljan opis scenarija. Navedite najmanje 5 elemenata koje se pišu u okviru detaljnog opisa scenarija.


- A. _____
- B. _____
- C. _____
- D. _____
- E. _____

13. (1 bod) Prilikom izlučivanja i analize zahtjeva, koriste se UML dijagrami obrazaca uporabe, uz koje se u projektnoj dokumentaciji obično navodi i detaljan opis scenarija. Navedite najmanje 5 elemenata koje se pišu u okviru detaljnog opisa scenarija.

- A. Naziv
- B. Namjena
- C. Opis
- D. Glavni aktor
- E. Preduvjeti





1. (1 bod) Zahtjevi koji ne sadržavaju konflikte ili kontradikcije u opisima zahtijevanih funkcionalnosti zovu su:


- A. Razumljivi
 - B. Adaptibilni
 - C. Provjerljivi
 - D. Kompletni
 - E. Konzistentni
- 

1. (1 bod) Zahtjevi koji ne sadržavaju konflikte ili kontradikcije u opisima zahtijevanih funkcionalnosti zovu su:

- A. Razumljivi
- B. Adaptibilni
- C. Provjerljivi
- D. Kompletni
- E. Konzistentni



2. (1 bod) Akronimi MDSD, MBASE, MBE, MDE, MDD, MDA, DDD označavaju metodologije oblikovanja programske potpore zasnovane na:

- A. Modelima
 - B. Inženjerstvu
 - C. Komponentama
 - D. Modularnosti
 - E. Domenskom znanju
- 

2. (1 bod) Akronimi MDSD, MBASE, MBE, MDE, MDD, MDA, DDD označavajo metodologije oblikovanja programske podpore zasnovane na:



A. Modelima*

B. Inženjerstvu


C. Komponentama

D. Modularnosti

E. Domenskom znanju



3. (1 bod) Oblikovanje sustava zasnovano na ponovnoj ili višestrukoj uporabi elemenata sustava (engl. reuse-oriented development) glavno je obilježje kojeg modela procesa programskog inženjerstva?

- A. Vodopadnog
 - B. Unificirani proces
 - C. Komponentnog
 - D. Agilnog
 - E. Evolucijskog
- 

3. (1 bod) Oblikovanje sustava zasnovano na ponovnoj ili višestrukoj uporabi elemenata sustava (engl. *reuse-oriented development*) glavno je obilježje kojeg modela procesa programskog inženjerstva?



A. Vodopadnog

B. Unificirani proces


C. Komponentnog*

D. Agilnog

E. Evolucijskog



4. (1 bod) U spiralni pristup iteracijama razvoja programske potpore ne pripada sektor:

- A. Postavljanja ciljeva
 - B. Studije izvedivosti
 - C. Planiranja
 - D. Razvoja i validacije
 - E. Procjene i smanjivanja rizika
- 

4. (1 bod) U spiralni pristup iteracijama razvoja programske potpore ne pripada sektor:



A. Postavljanja ciljeva

B. Studije izvedivosti


C. Planiranja

D. Razvoja i validacije



E. Procjene i smanjivanja rizika




5. (1 bod) Što je od navedenog značajka Unificiranog procesa (engl. Unified Process) modela procesa programskog inženjerstva?

- A. Svaka aktivnost se provodi samo u jednoj određenoj fazi razvoja
 - B. Korišćenje malih inkremenata
 - C. Malo dokumentacije
 - D. Modeli se dokumentiraju dijagramima
 - E. Ad-hoc određivanje zahtjeva klijenata
- 

5. (1 bod) Što je od navedenog značajka Unificiranog procesa (engl. *Unified Process*) modela procesa programskog inženjerstva?
- A. Svaka aktivnost se provodi samo u jednoj određenoj fazi razvoja
 - B. Korištenje malih inkremenata
 - C. Malo dokumentacije
 - D. Modeli se dokumentiraju dijagramima*
 - E. Ad-hoc određivanje zahtjeva klijenata



6. (1 bod) Koja od sljedećih tvrdnji ne vrijedi za UML-sekvencijske dijagrame?

- A. Dobri su za praćenje stanja sustava
 - B. Prikazuju objekte kroz životne crte
 - C. Prikazuju interakciju u sustavu
 - D. Mogu prikazivati aktore i objekte
 - E. Dobri su za prikaz komunikacije među objektima
- 

6. (1 bod) Koja od sljedećih tvrdnji **ne** vrijedi za UML-sekvencijske dijagrame?



A. Dobri su za praćenje stanja sustava*

B. Prikazuju objekte kroz životne crte


C. Prikazuju interakciju u sustavu

D. Mogu prikazivati aktore i objekte

E. Dobri su za prikaz komunikacije među objektima



7. (1 bod) Princip oblikovanja programske
potpore kod kojeg se teži minimizaciji
komunikacije komponenata:

- A. Smanji komunikaciju
 - B. Povećaj koheziju
 - C. Smanji koheziju
 - D. Zadrži razinu apstrakcije
 - E. Smanji međuovisnost
- 

7. (1 bod) Princip oblikovanja programske potpore kod kojeg se teži minimizaciji komunikacije komponenata:

A. Smanji komunikaciju



B. Povećaj koheziju*

C. Smanji koheziju


D. Zadrži razinu apstrakcije

E. Smanji međuovisnost*

Priznaju se rješenja A ili B





8. (1 bod) Princip oblikovanja programske potpore kod kojeg se, između ostalog, izbjegava uporaba najnovijih tehnologija kao i rijetko upotrebljavanih dijelova knjižnica naziva se:


- A. Oblikuj konzervativno
 - B. Oblikuj po ugovoru
 - C. Oblikuj za fleksibilnost
 - D. Planiraj zastaru
 - E. Oblikuj za prenosivost
- 

8. (1 bod) Princip oblikovanja programske potpore kod kojeg se, između ostalog, izbjegava uporaba najnovijih tehnologija kao i rijetko upotrebljavanih dijelova knjižnica naziva se:

- A. Oblikuj konzervativno
- B. Oblikuj po ugovoru
- C. Oblikuj za fleksibilnost
- D. Planiraj zastaru*
- E. Oblikuj za prenosivost





9. (1 bod) Za objekt u UML-dijagramu objekata vrijedi tvrdnja:


- A. Objekti imaju definiciju metoda.
 - B. Objekti imaju vrijednost atributa.
 - C. Simbol objekta je pravokutnik s tri prelinca.
 - D. Objekti imaju označenu vidljivost atributa.
 - E. Objekti imaju definiciju atributa.
- 

9. (1 bod) Za objekt u UML-dijagramu objekata vrijedi tvrdnja:

- A. Objekti imaju definiciju metoda.
- B. Objekti imaju vrijednost atributa.*
- C. Simbol objekta je pravokutnik s tri prelinca.
- D. Objekti imaju označenu vidljivost atributa.
- E. Objekti imaju definiciju atributa.



10. (1 bod) Metode razreda, koje se deklariraju unutar razreda ključnom riječi (1), pozivaju se preko (2).

- A. (1) abstract (2) naziva razreda
 - B. (1) abstract (2) naziva sučelja
 - C. (1) static (2) naziva razreda
 - D. (1) static (2) reference na objekt
 - E. (1) const (2) naziva razreda
- 

10.(1 bod) **Metode razreda**, koje se deklariraju unutar razreda ključnom riječi (1), pozivaju se preko (2).

A. (1) `abstract` (2) naziva razreda

B. (1) `abstract` (2) naziva sučelja



C. (1) `static` (2) naziva razreda*

11.(1 bod) Liskovin princip zamjene (engl. *Liskov substitution principle*) kaže da ako se u varijablu tipa _____ stavi objekt tipa _____ , program se mora korektno izvoditi.


11.(1 bod) Liskovin princip zamjene (engl. *Liskov substitution principle*) kaže da ako se u varijablu tipa Nadrazreda, stavi objekt tipa podrazreda , program se mora korektno izvoditi.

12.(1 bod) Vrsta pridruživanja na UML-dijagramima razreda kod koje se, ako je uništen objekt agregata, nužno uništavaju i dijelovi tog agregata, naziva se _____.

12.(1 bod) Vrsta pridruživanja na UML-dijagramima razreda kod koje se, ako je uništen objekt agregata, nužno uništavaju i dijelovi tog agregata, naziva se _____ *Kompozicija* _____ .



13.(1 bod) Na UML-dijagramima obrazaca
uporabe, veza _____
može se crtati između dva aktora kao i
između dva obrasca uporabe.



13.(1 bod) Na UML-dijagramima obrazaca
uporabe, veza *Generalizacije* _____
može se crtati između dva aktora kao i
između dva obrasca uporabe.

MI 19/20 A

1. (1 bod) Navedite barem tri značajke dobrog programskog proizvoda.

mora osigurati traženu funkcionalnost i performanse, prihvatljivost za korisnika, pouzdanost, mogućnost lakog održavanja.

2. (1 bod) Navedite barem tri primjera dionika: .

kupac, klijent, rukovoditelj razvoja, razvojni
inženjer__ .

3. (1 bod) Navedite sve načine zapisivanja zahtjeva sustava.

strukturiranim prirodnim jezikom, posebnim jezicima za oblikovanje sustava, dijagramima i matematičkom notacijom.

4. (1 bod) Prilikom utvrđivanja prioriteta zahtjeva, zahtjevi koji se analiziraju u obliku naprednih mogućnosti i koji su pogodni za prikaz mogućnosti budućeg razvoja nazivaju se zahtjevima ____.

Niskog prioriteta.

5. (1 bod) Veza koja može biti između dva aktora na UML dijagramima obrazaca uporabe naziva se ____.

generalizacija

6. (1 bod) Nacrtajte poruku za uništavanje objekta na UML sekvencijskim dijagramima.



7. (1 bod) Kod unificiranog procesa razvoja programske potpore, jezgrena aktivnost ispitivanja (engl. *test*) programske potpore provodi se u više faza, ali u najvećoj mjeri tijekom faze

__izgradnje (engl. *construction*)__

- 8. (1 bod) Ekstremno programiranje i čisti (engl. *lean*) razvoj neki su od primjera ____ metoda procesa razvoja programske potpore.

- agilnih

9. (1 bod) Skup stavaka iz projektnog dnevnika zaostataka koji je određen za implementaciju u nekom sprintu naziva se

- __dnevnik sprinta (ili sprint dnevnik) (engl. *sprint backlog*)

- 10. (1 bod) Kod klasifikacije modela arhitekture, dinamički procesni model (engl. *component and-connector view type*) prikazuje

- __komponente u izvođenju__(engl. *runtime*).

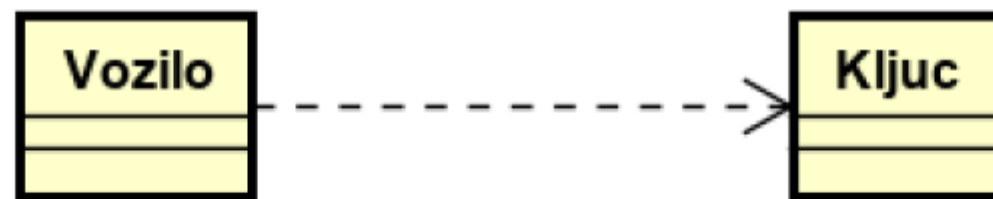
- 11. (1 bod) Princip dobrog oblikovanja programske potpore koji predviđa ispunjavanje preduvjeta, završnih uvjeta i invarijanti (engl. *invariants*) prilikom pozivanja neke metode naziva se

- Oblikuj po ugovoru (engl. *Design by Contract*)____.

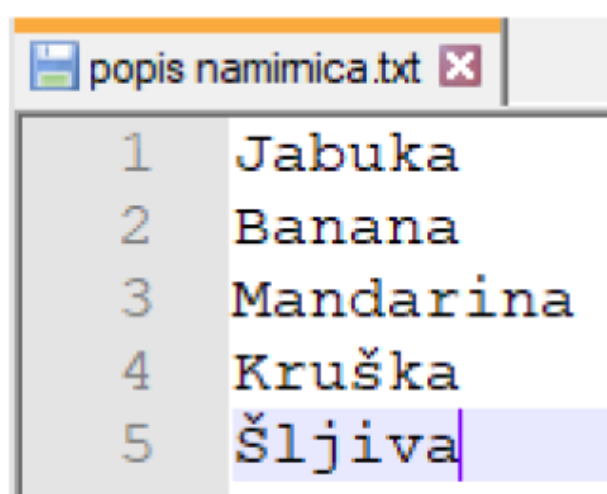
- 12. (1 bod) Nacrtajte dijagram razreda za sljedeći kod:

- `public class Kljuc {`
- `...`
- `}`
- `public class Vozilo{`
- `public void otključaj(Kljuc k){`
- `k.ocitaj();`
- `}`
- `}`

Rj.

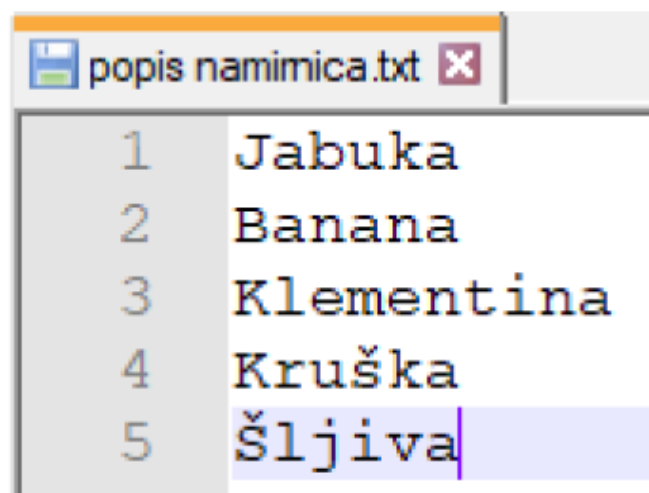


13. (1 bod) Trenutno se nalazimo na grani master. Osim nje, postoji i grana develop. Obje grane sadrže istu datoteku „popis namirnica.txt“ različitog sadržaja (prikazano na donjoj slici). Napišite kako će izgledati datoteka na trenutnoj grani ukoliko pokrenemo naredbu „git merge develop“.



```
1 Jabuka
2 Banana
3 Mandarina
4 Kruška
5 Šljiva
```

grana master



```
1 Jabuka
2 Banana
3 Klementina
4 Kruška
5 Šljiva
```

grana develop

Rj.

Jabuka

Banana

<<<<<< HEAD

Mandarina

=====

Klementina

>>>>>> develop

Kruška

Šljiva

2019 B

-
- 1. (1 bod) Navedite barem tri poteškoće u razvoju programske potpore.

-
- heterogenost razvoja, ograničeno vrijeme isporuke, povećanje pouzdanosti, česte izmjene zahtjeva i složenost razvijenog programskog sustava .

- 2. (1 bod) Objasnite u kakvom su odnosu učinkovitost programske potpore i lakoća korištenja.

-
- Proturječje: povećanje učinkovitosti specijalizacijom čini programsku potporu manje razumljivom i može smanjiti mogućnost održavanja ili ponovnog korištenja. Povećanje lakoće korištenja (npr. uključivanje uputa tijekom rada) može smanjiti učinkovitost.

- 3. (1 bod) Zahtjevi visoke razine apstrakcije koji se pišu prirodnim jezikom i crtaju grafičkim dijagramima nazivaju se

- korisnički

- 4. (1 bod) Recenzija zahtjeva, izrada prototipa sustava i generiranje ispitnih slučajeva su tehnike

- Validacije zahtjeva

-
- 5. (1 bod) Vrsta veze koju koristimo na UML dijagramima obrazaca uporabe za prikaz opcionalne, dodatne funkcionalnosti nekog baznog obrasca uporabe naziva se

- extend

-
- 6. (1 bod) Kod UML sekvencijskih dijagrama, punom linijom s punom strelicom na jednom kraju, kao što je prikazano na slici dolje, označavamo ____ poruku

- sinkronu

-
- 7. (1 bod) Kod unificiranog procesa razvoja programske potpore, jezgrena aktivnost modeliranja poslovnog procesa provodi se u najvećoj mjeri tijekom faze _____ .

- početka (engl. *inception*)

- 8. (1 bod) Ekstremno programiranje koristi međusobno provjeravanje tijekom implementacije programske potpore koje se naziva ____

- programiranje u paru__ (engl. *pair programming*).

- 9. (1 bod) Navedite barem tri elementa sprinta po Scrum metodologiji:

-
- sastanak planiranja Sprinta, __dnevni Scrum (engl. daily Scrum, stand-up__), posao razvoja, sastanak revizije Sprinta i retrospektiva Sprinta. (priznati i dnevnik sprinta)

- 10. (1 bod) Kod klasifikacije modela arhitekture, model alociranih elemenata (engl. *allocation view type*) dokumentira _____.

- odnos programske potpore i razvojne/izvršne okoline

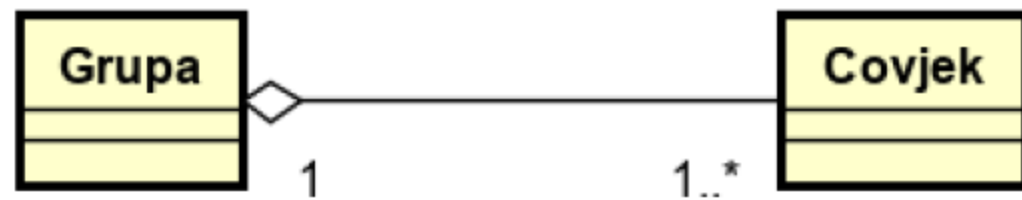
- 11. (1 bod) Princip dobrog oblikovanja programske potpore koji posebno napominje da se ne upotrebljava izravno umetanje podataka ili konfiguracija u izvorni programski kôd (engl. *hard code*) naziva se _____

- oblikuj za fleksibilnost

12. (1 bod) Nacrtajte dijagram razreda za sljedeći kod:

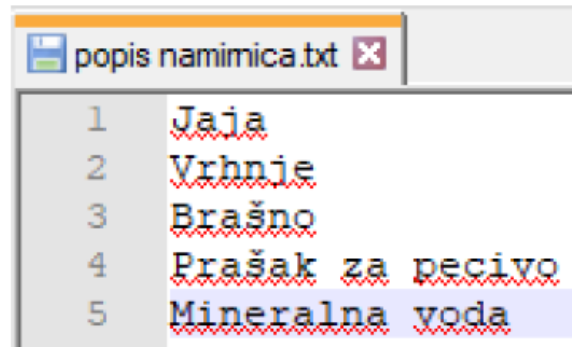
```
public class Grupa{  
    private List<Covjek> clanovi;  
    public Grupa (List<Covjek> clanovi){  
        this.clanovi = clanovi;  
    }  
}  
  
public class Covjek{  
    private grupa;  
}
```

Rj.



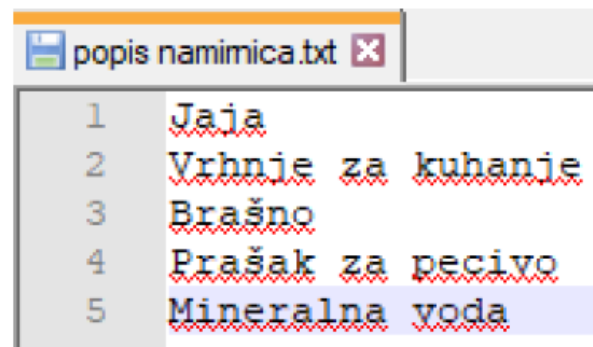
(asocijacija->0.5 bodova)

13. (1 bod) Trenutno se nalazimo na grani develop. Osim nje, postoji i grana master. Oboje grane sadrže istu datoteku „popis namirnica.txt“ različitog sadržaja (prikazano na donjoj slici). Napišite kako će izgledati datoteka na trenutnoj grani ukoliko pokrenemo naredbu „git merge master“.



```
1 Jaja
2 Vrhnje
3 Brašno
4 Prašak za pecivo
5 Mineralna voda
```

grana master



```
1 Jaja
2 Vrhnje za kuhanje
3 Brašno
4 Prašak za pecivo
5 Mineralna voda
```

grana develop

- Rj.
- Jaja
- <<<<<< HEAD
- Vrhnje za kuhanje
- =====
- Vrhnje
- >>>>>> master
- Brašno
- Prašak za pecivo
- Mineralna voda

MI 18/19 A

-
- 1. (1 bod) Složenost, usklađenost, promjenjivost i nevidljivost su _____problemi razvoja programske potpore.

- suštinski (inherentni, nezaobilazni)

-
- (1 bod) Arhitektura programske podrške mora rezultirati dokumentacijom koja ima tri svojstva:

-
- dobra___, ___ispravna___ i ___uspješna___.

-
- 3. (1 bod) Problemi zahtjeva domene primjene su:

- Razumljivost i implicitnost

-
- 4. (1 bod) Model ubrzanog razvoja (engl. *Agile modeling*) programske potpore posebno je pogodan za _____ projekata a karakterizira ga _____ interakcija s klijentima.

- brzi razvoj manjih i srednjih
- stalna

- 5. (1 bod) Kod UML dijagrama obrazaca uporabe, višestrukosti se mogu pisati na kojoj vrsti veza?

- Asocijacije

-
- 6. (1 bod) Pravokutnikom na životnoj liniji (engl. *lifeline*) objekta prikazuje se ____.

- aktivnost objekta

-
- 7. (1 bod) Kod unificiranog procesa razvoja programske potpore, jezgrena aktivnost analize i oblikovanja programske potpore provodi se u najvećoj mjeri tijekom faze ____.

- razrade (engl. Elaboration)

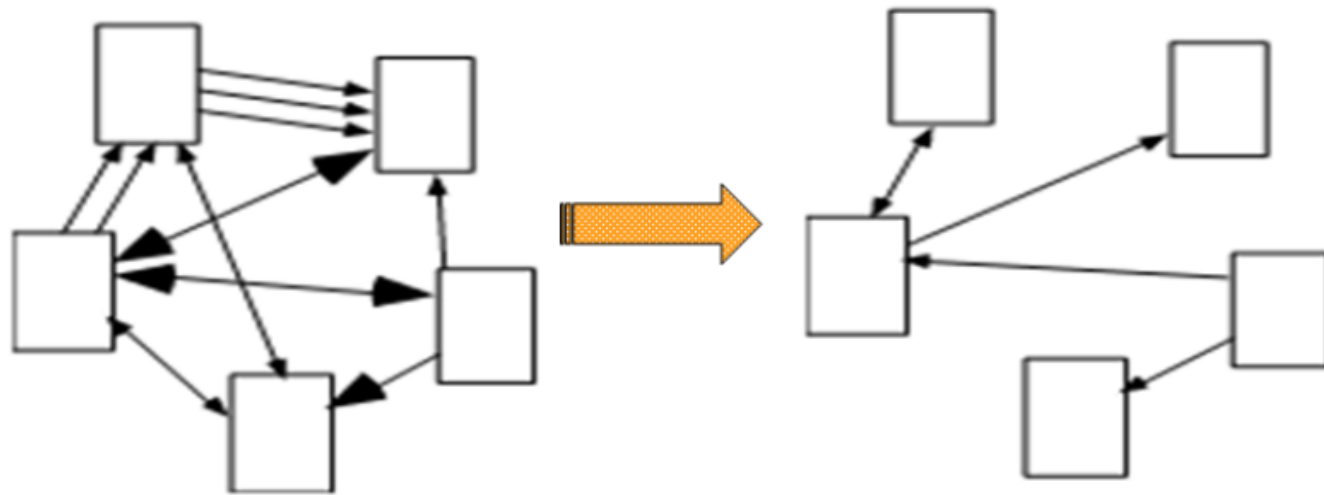
- 8. (1 bod) Projektni dnevnik (engl. *product backlog*), dnevnik sprinta (engl. *sprint backlog*) i ____ su Scrum artefakti

- inkrement

-
- 9. (1 bod) Konceptijska arhitektura (engl. *Conceptual Architecture*) programske potpore usmjerena je na pogodnu dekompoziciju sustava i komunikaciju s _____.

- netehničkim dionicima (uprava, prodaja, korisnici)

10. (1 bod) Primjena kojeg principa oblikovanja objektno usmjerene arhitekture programske potpore vodi do transformacije arhitekture prema slici?



- ____Smanjivanje međuovisnosti (engl____Reduce coupling where possible)

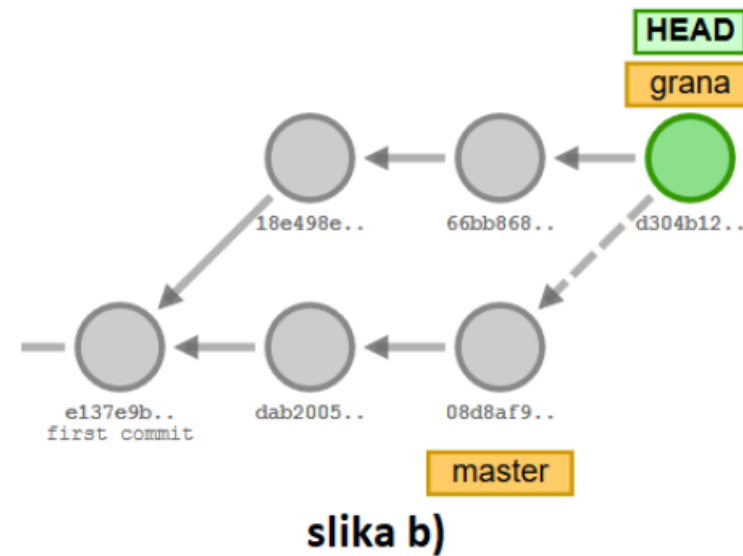
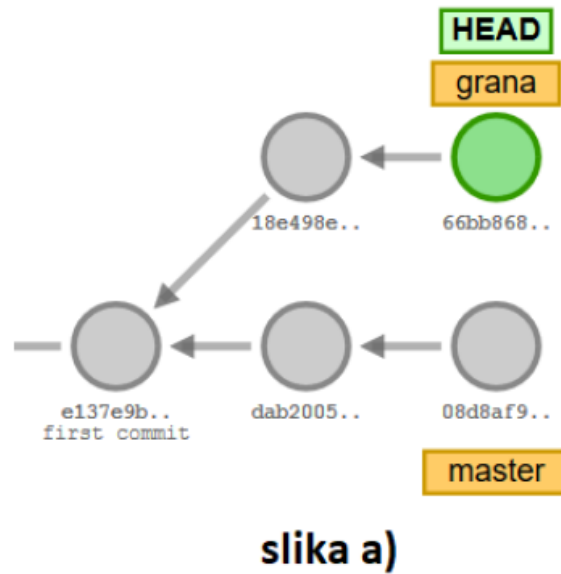
- 11. (1 bod) Dobri dokumenti oblikovanja arhitekture, kao pomoć izradi bolje programske potpore, pišu se s gledišta _____ i služe kao sredstvo komunikacije _____.

- Čitatelja
- tima za implementaciju/budućih osoba zaduženih za održavanje, razvoj ili promjene/osobama uključenih u oblikovanje povezanih sustava ili podsustava

- 12. (1 bod) Nacrtajte grafičku oznaku sadržavanja UML dijagrama razreda kojom se naglašava ovisnost razreda o životnom ciklusu drugog razreda i prikažite primjer dva razreda u takvoj vezi.



13. (1 bod) Kojom Git naredbom repozitorij pohranjuje promjene iz grane „master“ u granu „grana“, tj. prelazi iz stanja sa slike a) u stanje sa slike b)?



- Git merge master

by mačka