Specifikacija i modelovanje softvera
Teorijska pitanja: 2010/2011

1. Kako se naziva prvi korak u procesu razvoja softvera?
   -Izrada specifikacije zahteva

2. Definišite pojam ZAHTEV.
   -Zahtev je svojstvo ili osobina koja se mora iskazati u cilju stvaranja podloge za resavanje nekog problema.

3. Na šta se odnose ZAHTEVI?
   -Zahtevi se odnose na neke aspekte funkcionisanja sistema koji je predmet analize

4. Odakle po prirodi proizilaze zahtevi?
   -Proizilaze iz sistema.Predstavljaju slozene kombinacije zahteva razlicitih ljudi na razlicitim nivoima organizacione strukture u ambijentu u kojem sistem radi.

5. Koje osnovne elemente obuhvata oblast znanja Softverski zahtevi u sklopu SWEBOK?(7)
   -Osnovni softverski zahtevi , Proces zahteva , Iskazivanje zahteva , Analiza zahteva , Specifikacija zahteva , Validacija zahteva , Prakticni zahtevi.

6. Koje pojmove, prema SWEBOK-nomenklaturi obuhvataju osnovni pojmovi softverskih zahteva? (6)
   -Definisanje softverskoh zahteva, Procesni i zahtevi proizvoda, Funkcionalni i nefunkcionalni zahtevi , Emergent properties, Merljive zahteve , Zahtevi sistema i zahtevi softvera.

7. Šta, prema SWEBOK-nomenklaturi obuhvata Proces zahteva? (4)
   -Zahtevi modela , zahtevi korisnika,zahtevi za Podrsku i upravljanje, zahtevi za kvalitet i poboljsanja .

8. Šta, prema SWEBOK-nomenklaturi obuhvata Iskazivanje zahteva? (2)
   -zahtevi izvora, Elicitation Techniques (iznajmljivanje tehnika u bukvalnom prevodu).

9. Šta, prema SWEBOK-nomenklaturi obuhvata Analiza zahteva? (4)
   -Klasifikaciju zahteva, Konceptualno modelovanje, Arhitektonski dizajn i alokacija zahteva, ponovno pregovaranje u vezi zahteva.

10. Šta, prema SWEBOK-nomenklaturi obuhvata Specifikacija zahteva? (3)
    -Definisanje sistema, Specifikaciju sistema, Specifikaciju softvera.

11. Šta, prema SWEBOK-nomenklaturi obuhvata Validacija zahteva? (4)
    -Pregled zahteva, prototipovi, validacija modela, testovi validacije.

12. Šta, prema SWEBOK-nomenklaturi obuhvataju Praktični zahtevi? (5)
    -Ponavljanje procesa zahtava, promena menadzmenta, svojstva zahteva, trazenje skrivenih zahteva, uporedjivanje zahteva po prioritetima.

13. Koja tri elementa obuhvata Iskazivanje zahteva?
    -Pribavljanje , otkrivanje , prikupljanje zahteva.

14. Navedite izvore zahteva. (5) Kakva je uloga Ciljeva sistema?
    -Ciljevi sistema , domensko znanje pregledi najsireg moguceg skupa ucesnika , radni ambijent, organizacioni aspekti. Ciljevi sistema nam predstavljaju cilj koji se mora dostici da bi se sistem smatrao uspesnim.

15. Navedite izvore zahteva. (5) Kakva je uloga Domenskog znanja?
    -Domensko znanje predstavlja znanje projektanata sistema o problemu koje ce sistem koji se izradjuje da resava.

16. Navedite izvore zahteva. (5) Kakva je uloga pogleda najšireg mogućeg skupa učesnika?
    -Sagledavanje sa svih aspekata mogucih korisnika i svih onih koji bi mogli doci u kontak sa sistemom.Na taj nacin se prikupljaju informacije o dodatnim zahtevima.

17. Navedite izvore zahteva. (5) Kakva je uloga radnog ambijenta sistema?
-Potrebno je znati u kojim uslovima se sistem koristi, kako bi bio optimizovan za rad u potrebnim uslovima.

18. Navedite izvore zahteva. (5) Kakva je uloga organizacionih aspekata sistema?
-Dobra organizacija za lako, brzo i efikasno koriscenje sistema.

19. Navedite pet osnovnih tehnika iskazivanja/prikupljanja zahteva.
-Intervjuji – tradicionalna tehnika , scenariji , prototipovi , radni sastanci, posmatranje.

20. Kakva je, sa aspekta prikupljanja zahteva, razlika između KUPACA i KORISNIKA?
-Sistem se modeluje prema korisnicima ne prema kupcima tako da je prilikom specifikacije zahteva razvojni tim u kontaktu iskljucivo sa buducim korisnicima sistema.

21. Navedite osnovne karakteristike Analize zahteva.
-Predstavlja predmet dogovora pri cemu se kljucna svojstva odvajaju od manje vaznih.Cesto podrazumeva modeliranje ponasanja krajnjih korisnika u cilju boljeg razumevanja poslovne uloge i funkcija sistema kloji je predmet analie.Proces je iterativan i zahteva promene, redefinicije i rafinacije u hodu.Proces je iscrpljujuci i zahteva mnogo vremena.

22. Navedite osnovne ciljeve Analize zahteva.
-Otkrivanje i resavanje konflikta izmedju zahteva.Otkrivanje granica sistema i njegovih interakcija sa okruzenjem.Preslikavanje zahteva od strane sistema na zahteve putem softvera.

23. Koja je osnovna razlika između FUNKCIONALNIH i NEFUNKCIONALNIH zahteva?
-Funkcionalni su mogucnosti, a Nefunkcionalni nametnuta ogranicenja koje resenje treba da zadaovoljava.

24. Kojim formalizmima se moraju opisivati zahtevi?
-Zahtevi se moraju iskazivati formalizmima iz korisnickog domena.

25. Navedite najmanje četiri od osam osnovnih osobina zahteva.
1.Mogucnost verifikacije unutar zadatih ogranicenja koja ukljucuju resurse.
2.Promenljivost
3.Iskazuju se formalizmima iz korisnickog domena.
4.Moraju biti potrebni i dovoljni.
5.Moraju biti jedinstveno identifikovani i ukljuceni u proces rukovanja konfiguracijom kroz celokupan zivotni ciklus predmeta primopredaje.
6.Moraju biti jedinstvenoprepoznati u elementima argitekture predmeta primopredaje.
7.Moraju posedovati jedinstveni prioritet.
8.Stepen do koga deluje na sistem.

26. Zbog čega je važno dodeliti jedinstven prioritet svakom zahtevu?
-Da bi se mogla odrediti vaznost svakog dela zahteva za buduce korisnike sistema.

27. Navedite osnovnu kategorizciju prioriteta zahteva?
-Esencijalan, vrlo pozeljan , pozeljan , opcion.

28. Obrazložite značaj mogućnosti praćenja zahteva kroz sve faze životnog ciklusa softvera.
-Pracenje zahteva je bitno jer se mogu javiti promene u zahtevima koje razlicito mogu uticati u razlicitim fazama razvoja softvera. U nekim fazama se novi zahtevi mogu lako dodati, dok u nekim zahtavaju korenite promene ili cak ponovni razvoj od pocetka.

29. Šta je svrha modelovanja zahteva?
-Stvaranje podloga za struktuiranje resenja. Eksperimentisanje u cilju istrazivanja vise verzija resenja. Oslonac na apstrakcije u cilju rukovanja slozeniscu. Skracenje vremena izrade proizvoda. Smanjenje troskova razvoja. Rukovanje rizicima.

30. Navedite bar četiri od sedam tipova modela koji se koriste pri modelovanju zahteva.
    1.Model tokova podataka
    2.Model tokova kontrola
    3.Model stanja
    4.Model dogadjaja
    5.Model korisnickih interakcija
    6.Model objekta
    7.Model slucajeva koriscenja

31. Koji elementi imaju direktan uticaj na izbor tipa modela. (4)
    -Priroda tipa problema koji se resava.
    -Nivo ekspertskog znanja analiticara
    -Nacin na koji kupac/korisnik iskazuje zahtev
    -Raspolozivost metoda i alata

32. Koja je osnovna karakteristika formalnih modela?
    -Formalni modeli koriste notacije zasnovane na elementima koriscene metodologije modelovanja.

33. Šta se očekuje od modela? (Navedite bar četiri od šest elemenata)
    -Da obezbedi preciznost pri modelovanju.
    -Da poseduje internu i eksternu konzistentnost.
    -Da se moze obrazloziti na jednostavan nacin.
    -Da se moze menjati uz minimalan napor.
    -Da bude sto je moguce jednostavniji ali ne i banalan.

34. Šta se očekuje od zahteva?(3)
    -Bilo koji skup zahteva treba da opise sve delove sistema uljucujuci i njegove granice.
    -Neophodno je poznavati koji su objekti ili entiteti ukljuceni u sistem, kakva je njihova struktura (atributi), na koji nacin su povezani i koje transformacije nad njima obavljaju.
    -Po izolovanju elemenata sistema neophodno je koristiti detaljnije tehnike za predstavljanje specificnosti analiziranog sistema.

35. Šta podrazumeva VIZUALNO MODELOVANJE?
    -Predstavlja modelovanje uz oslonac na neku standardnizovanu graficku notaciju.

36. Šta predstavlja UML?
    -UML je standardni je jezik za vizuelizacuju, specificiranje , konstrukciju i dokumentovanje softverskih proizvoda.
    -UML predstavlja standardni jezik za modelovanje sistema koji sadrze softver.

37. Za šta se koristi UML?
    -Koristi se za vizuelizaciju (graficki jezik), specifikaciju (izrada preciznih nedvosmislenih i potpunih modela), konstukciju (modeli se direktno povezju sa skupom programskih jezika) i dokumentovanje produkta softverskih sistema.

38. Koji četiri elementa čine UML?
    -Pregledi (razliciti aspekti modelovanja sistema)
    -Dijagrami (Graficke predstave sadrzaja razlicitih aspekata sistema)
    -Elementi modela (gradivni elementi dijagrama)
    -Opsti mehanizmi (Dopune, komentari, prosirenja ...)

39. Šta predstavljaju Pogledi kao element UML-a? Navedite osnovne poglede.
    -Pogledi = razliciti aspekti modelovanja sistema:
        -use case
        -funkcionalni pogled korisnika na sistem (Actor)
        -logicki pogled
        -pogled sa aspekta strukture (staticka athitektura sistema/resenja)
        -pogled sa aspekta dinamike (aspekti komunikacije i sinhronizacije u sklopu sistema/resenja)
        -fizicka arhitektura sistema.

40. Šta predstavljaju Dijagrami kao element UML-a? Navedite osnovne dijagrame.
    -Graficke predstave sadrzaja razlicitih aspekata sistema.
        -Dijagram slucajeva koriscenja (use case)
        -Dijagram klasa (Class)
        -Dijagram objekata (Object)
        -Dijagram stanja (State)
        -Dijagram sekvence (Sequence)
        -Dijagram saradnje (Collaboration)
        -Dijagram aktivnosti (Activity)
        -Dijagram delova koda (Component)
        -Dijagram fizicke arhitekture (Deployment)

41. Šta predstavljaju Elementi UML-a? Navedite br 6 od 12 osnovnih elemenata UML-a.
    -Gradivni elementi dijagrama
        -Klasa
        -Objekat
        -Stanje
        -Slucaj koriscenja
        -Cvor
        -Sprega
        -Paket
        -Veza
        -Komponenta
        -Zabeleska
        -Poruka
        -Akcija

42. Šta predstavljaju Opšti mehanizmi UML-a?
    -Dopune, komentari, prosirenja i sl.

43. Koliko nivoa poseduje metamodel UML-a?
    -Cetiri nivoa : Meta-metamodel , metamodel , model ,Korisnicki objekti.

44. Koja je osnovna namena USE-CASE dijagrama?
    -Izolovanje i opis funkcionalnih zahteva prema modelovanom sistemu/resenju.
    -Formiranje jasnog i konzistentnog opisa onoga sto sistem treba da radi.
    -Stvaranje podloga za formiranje sistemskih testova u cilju verifikacije i validacije sistema/resenja.
    -Stvaranje podloge za povezivanje funkcionalnih zahteva sa klasama implementiranim u sistemu/resenju.

45. Šta predstavlja slučaj korišćenja?
    -Slucaj koriscenja predstavlja specifikaciju nuza akcija, ukljucujuci i razliciti varijante, koje sistem obavlja u interakciji sa ucesnicima.

46. Kako se definiše učesnik (akter)?
    -Neko ili nesto u interakciji sa sistemom (razmenjuju informacije sa sistemom)
    -Akter predstavlja ulogu u odnosu na sistem, ne nekog pojedinacnog korisnika sistema.

47. Šta predstavlja dijagram slučajeva korišćenja? Koja četiri elementa on poseduje?
    -Dijagram slucajeva koriscenja prikazuje odnose izmedju ucesnika i slucajeva koriscenja u sklopu sistema.
    -Sadrzi :
        -Sistem koji se modeluje
        -Ucesnike sa kojima je sistem u interakciji
        -Slucajeve koriscenja ili servise koje sistem pruza
        -Veze izmedju gore nabrojanih elemenata

48. Šta se podrazumeva pod granicom sistema?
    -Linija razdvajanja izmedju sistema i okruzenja
    -Svi slucajevi koriscenja se nalaze unutar granica sistema
    -Svi ucesnici se nalaze van granica sistema

49. Kakva je razlika između veze sadržavanja i veze proširivanja?
-Koristimo <<extend>> kada modelujemo prosiranja ili varijacije nekog slucaja koriscenja koji postoji kao autonomna celina
-Koristimo <<include>> kada zelimo da podelmo zajednicko ponasanje izmedju dva ili vise slucajeva koriscenja

50. Gde se, sa aspekta sistema, nalaze slučajevi korišćenja?
-Svi slucajevi koriscenja se nalaze unutar granica sistema.

51. Gde se, sa aspekta sistema, nalaze učesnici?
Svi ucesnici se nalaze van granica sistema.

52. Ko inicira slučaj korišćenja?
-Slucaj koriscenja se uvek inicira od strane aktera.

53. Koji tip veze je jedino dozvoljen između učesnika?
-Kada vise ucesnika u sklopu svojih uloga ucestvuje u sklopu neke opste uloge njihov odnos se opisuje generalizacijom.

54. Šta predstavlja opis slučaja korišćenja?
-Dokument koji sadrzi detaljnu specifikaciju slucaja koriscenja
-Sadrzaj moze biti obican tekstualni opis ili struktuirani tekstualni opis.

55. Koji tipovi veza su dozvoljeni između slučajeva korišćenja?
-Generalizacija : generalizovan slucaj koriscenja opisuje ponasanje zajednicko za grupu specijalizovanih slucajeva koriscenja
-Ukljucivanje (inclusion) : slucaj koriscenja je deo drugog slucaja koriscenja
-Prosirivanje (extension) : slucaj koriscenja moze prosiriti drugi slucaj koriscenja

56. Kada se između slučajeva korišćenja koristi veza generalizacije?
-Koristi se kada vise slucajeva koriscenja poseduje neku zajednicki pod-zadatak ali svaki od njih to posmatra iz svog ugla.

57. Da li specijalizovani slučaj korišćenja može posedovati vlastiti skup učesnika? (1. Da 2. Ne)
-Specijalizovani slucaj koriscenja moze integrirati sa novim ucesnicima.

58. Da li specijalizovani slučaj korišćenja može proširiti preduslove? (1. Da 2. Ne)
-Moze prosiriti preduslove i post-uslove.

59. Kada se između slučajeva korišćenja koristi veza sadržavanja?
Kada skup slucajeva koriscenja poseduje zajednicko ponasanje, koje je moguce modelovati jednim slucajem koriscenja, tada je moguce koristiti ovaj tip veze.

60. Koliko puta se može izvršiti sadržani slučaj korišćenja?
-Ako X<<sadrzi>>Y indicira da se izvrsavanjem procesa X uvek podrazumeva izvrsavanje procesa Y najmanje jednom.

61. Da li sadržani slučaj korišćenja mora očuvati post-uslove? (1. Da 2. Ne)
-Ne.

62. Kada se između slučajeva korišćenja koristi veza proširivanja?
-Sluzi za definisanje tacke prosirenja nekog slucaja koriscenja. Koristimo <<extend>> kada modelujemo prosirenje ili varijacije nekog slucaja koeiscenja koji postoji kao autonomna celina.

63. Da li proširen slučaj korišćenja mora eksplicitno deklarisati sve svoje tačke proširenja? (1. Da 2. Ne)
-Da.

64. Šta je stereotip sa aspekta UML notacije?
-Konstrukcija koje prosiruje UML recnik.
-Dodaje nova znacenja postojecim entitetima.
-ogranicen je sa <<>> delimiterima.

65. Šta je namena UML dijagrama aktivnosti?
-Namena dijagrama aktivnosti je modelovanje toka aktivnosti u sklopu sa postupkom koji je deo neke slozenije aktivnosti. Kod projekata koji koriste modele slucajeva koriscenja dijagram aktivnosti se moze iskoristiti za detaljnije modelovanje posmatranog slucaja koriscenja. Dijagrami aktivnosti se mogu koristiti nezavisno od slucajeva koriscenja za modelovanje funkcija na nivou poslovnog ponasanja.

66. Na šta se fokusira UML dijagram aktivnosti?
-Posto se modeluje postupak, dijagram aktivnosti se fokusira na redosled aktivnosti i uslove koji upravljaju tokom procesa koji se opisuje.

67. Šta predstavlja aktivnost u sklopu UML dijagrama aktivnosti?
-Predstavlja skup operacija neophodnih za obavljanje nekog pojedinacnog posla pri cemu dolazi do iniciranja akcija koje uticu na stanje sistema ili davanje povratnih informacija vrednosti neophodnih za sintezu upravljanja u sklopu posla.

68. Šta predstavljaju plivačke staze u sklopu UML dijagrama aktivnosti?
-Aktivnosti se grupisu u vertikalne ili horizontalne zone oivicene isprekidanim linijama. Svaka zona predstavlja pojas odgovornosti, koji obicno implementira skuo klasa ili objekata. Npr jedna plivacka staza moze predstavljati objekte vise klasa koji konkretno obavljaju jedinstvenu aktivnost.

69. Šta definišu UML dijagrami interakcija?
-Desinisu nacin odvijanja upravljanja razmatranjem odnosa izmedju objekata.

70. Koji su osnovni UML dijagrami interakcija?
-Dijagram sekvence i dijagram saradnje.

71. Šta predstavlja UML dijagram sekvence?
-Deo UML-a koji omogucava graficko predstavljanje hronoloskog ponasanja sistema.

72. Šta se opisuje pomoću UML dijagrama sekvence?
-Opisuje se zivotni vek objekta u defunisanom vreenskom periodu.

73. Šta prikazuje horizontalna osa dijagrama sekvence?
-Uloge objekta koje odslikavaju pojedinacne objekte u okviru modelovane saradnje.

74. Navedite osnovne elemente UML dijagrama sekvence?
-Objekti, zivotne linije , ucesnici , fokus kontrole , poruke.

75. Šta modeluje poruka u sklopu UML dijagrama sekvence?
-Poruka modeluje komunikaciju izmedju objekata. Ona prenosi informacije u cilju obezbedjenja izvrsenja aktivnosti. Prijem poruke pbicno rezultuje nekom reakcijom. Poruka se moze smatrati pokretacem.

76. Šta predstavlja rekurzivna poruka u sklopu UML dijagrama sekvence?
-Poruka koju objeat sam sebi salje.

77. Navedite osnovne korake pri preiranju dijagrama sekvence. (7)
1.Ustanoviti kontakst interakcije (sistem, podsistem, operacije ili klasa, jedan scenario korisnickih funkcija, saradnja klasa)
2.Ustanociti scenu interakcije (identifikacija objekata koji ucestvuju u interakciji)
3.Ustanoviti liniju zivota za svaki objekat
4.Rasporediti poruke u vremenskom redosledu (od vrha ka dnu, izmedju linija zivota objekata uz definisanje kompozicije poruka)
5.Po potrebi obeleziti fokus kontrole na liniju zivota svakog objekta
6.Po potrebi specificirati vremenska ili prostorna ogranicenja (dodatna obelezja poruka koje se razmenjuju)
7.Po potrebi svakoj poruci dodati preduslove postuslove

78. Šta predstavlja UML dijagram saradnje?

-Deo UML-a koji omogucuje graficko predstavljanje interakcije izmedju objekata, kod koga objekti mogu biti razmesteni bilo gde na dijagramu.

79. Opišite osnovne razlike između UML dijagrama sekvenc i UML dijagrama saradnje.

-Dijagram sekvence jasno pokazuje redosled poruka dok se sa dijagrama saradnje redosked poruka tesko uocava. Dijagram sekvence ima veliki broj opcija za specificiranje detalja, dok dijagram saradnje ima manji broj notacionih mogucnosti.Dijagram saradnje je ekonomican sa aspekta koriscenja prostora dok dijagram sekvence raste u desnu stranu.

80. Koja je osnovna snaga UML dijagrama sekvence?

-Jasno prikazuje redosled poruka i poseduje veliki broj opcija za specificiranje detalja.

81. Koja je osnovna snaga UML dijagrama saradnje?

-Ekonomican sa aspekta koriscenja prostora i ima fleksibilno dodavanje objekta u dve dimenzije.

82. Koja je osnovna slabost UML dijagrama sekvence?

-Prosiruje se u desnu stranu i zauzima horizontalni prostor na dijagramu.

83. Koja je osnovna slabost UML dijagrama saradnje?

-Tesko se uocava redosled poruka i ima manji broj notacionih mogucnosti.

84. Na čemu je osnovni naglasak kod UML dijagrama saradnje?

-Dijagram saradnje naglasava strukturnu organizaciju objekata koji salju i primaju poruke.

85. Navedite bar pet od osam osnovnih osobina dobro strukturiranog UML dijagrama interakcije.

1.Fokusiran je na jedan aspekt dinamike sistema koji modeliramo.
2.Sadrzi samo neophodne elemente za razumevanje tog aspekta sistema.
3.Usmeren je na vlastit nivo apstrakcije i koristi samo one vizualne dopune koje pospesuju razumljivost.
4.Ukljucuju svu neophodnu ali ne i najmanju mogucu semantiku.
5.Imenovanje diagrama interakcije odgovara njegovoj nameni.
6.Adekvatno koristi dijagram sekvence i saradnje.
7.Pregledan je, koristi boje za markiranje zvaznih osobina interakcije.
8.Koristi samo jednostavnja grananja i to samo ako, sa sapekta modela, dovode do smanjenja neodredjenosti (za te namene je pogodniji dijagram aktivnosti).

86. Kakva je uloga UML dijagrama stanja?

-Pokazuje nam :
- -zivotni ciklus posmatranog objekta
- -dogadjaje koji uzrokuju prelaze stanja
- -akcije koje si posledica prelaza stanja

87. Šta predstavlja tranzicija u sklopu UML diajgrama stanja? Koje elemente ona poseduje?(5)

-Relacija izmedju dva stanja.Poseduje sledece elemente :
- -izvorisno stanje – stanje na koje deluje tranzicija
- -pobudni dogadjaj – njegov prijem pobudjuje tranziciju
- -zastitni uslov – uslov pobude tranzicije
- -akcija – izvrsivo nedeljivo procesiranje koje moze direktno delovati na objekat i indirektno uticati na stanja koja su dostizna iz izvorisnog stanja
- -odredisno stanje – stanje koje je aktivno po zavrsetku tranzicije

88. Šta predstavlja SPECIFIKACIJA DIZAJNA?

-Predstavlja korak u kome se analiza zahteva prevodi u arhitekturu sistema koji je predmet projektovanja.

89. Šta definiše UML specifikacija dizajna?

-Definise nacin realizacije (implementacije) slucajeva koriscenja.

90. Čime se vrši implentacija slučajeva korišćenja?

-Saradnjom klasa.

91. Navedite osnovne elemente UML specifikacije dizajna. (3)
-Segregacija – razlaganje funkcionalne specifikacije na module
-hijerarhijsko organizovanje modula
-definisanje tokova podataka izmedju modula
-definisanje strukture spoljasnjih skladista podataka
-definisanje prstupnih puteva spoljasnjim skladistima podataka
-definisanje struktura podataka koje se razmenjuju medju modulima
-izrada kljucnih algoritama
-definisanje unutrasnje strukture svakog algoritma

92. Navedite četiri osnovna tipa veza u sklopu UML dijagrama klasa.
-Asocijativnost, zavisnost, generalizacija, realizacija

93. Šta modeluje asocijacija između klasa?
-Opisuje grupu veza sa zajednickom strukturom i semantikom i modeluje semanticku vezu izmedju klasa.

94. Kakva je razlika između AGREGACIJE i KOMPOZICIJE?
-I agregacija i kompozicija modeluju znacenje deo-celina ili deo-deo preko koje su objekti koji predstavljaju delove celine povezani sa objektom koji predstavlja celinu.Kod agregacije delovi ne dele sudbinu celine, dok kod kompozicije delovi moraju da dele sudbinu celine.

95. Kada je neophodno koristiti ASOCIJATIVNE KLASE pri UML specifikaciji dizajna?
-Asocijativna klasa je neophodna kod relacija više prema više i ona jedino može da sadrži informacije koje su jedinstvene za relaciju između te dve klase.

96. Šta podrazumeva veza ZAVISNOSTI?
-Podrazumeva vezu koja opisuje medjuzavisnost elemenata modela kod koje promena u sklopu jednog (nezavisnog) elementa uzrokuje promenu kod drugog (zavisnog) eementa.

97. Šta podrazumeva veza RAFINACIJE?
-Podrazumeva vezu izmedju jedne iste stvari na razlicitim nivoima apstrakcije.

98. Kakva je suštinska razlika u prirodi veza ZAVISNOSTI i RAFINACIJA?
-Veza zavisnosti se uspostavlja na istom nivou apstrakcije dok se rafinacija moze uspostaviti na razlicitim nivoima apstrakcije.

99. Šta predstavlja veza generalizacije u sklopu UML specifikacije dizajna?
-Predstavlja vezu izmedju klasa u kojoj jedna od njih ima opsta svojstva dok je druga njena specijalizacija.Podredjeni element je u potpunoj saglasnosti sa nadredjenim i poseduje sve osobine nad-klase (roditejska klasa - predak), ali moze posedovati i dodatne osobine koje definise pod-klasa (potomak). Veza generalizacije je jedino moguca izmedju klasa.

100. Navedite osnovne aspekte dizajna softverskih sistema. (4) Šta obuhvata KONCEPTUALNI DIZAJN?
1.Konceptualni dizajn
2.Tehnicki dizajn
3.Arhitektonski dizajn
4.Detaljni dizajn
Konceptualni dizajn predstavlja preciznu specifikaciju iz koje korisnik nedvosmisleno i jednoznacno dobija informaciju o tome sta ce da radi softver.

101. Navedite osnovne aspekte dizajna softverskih sistema. (4) Šta obuhvata TEHNIČKI DIZAJN?
1.Konceptualni dizajn
2.Tehnicki dizajn
3.Arhitektonski dizajn
4.Detaljni dizajn
Tehnicki dizajn omogucava projektantima i programerima razumevanje stvarno potrebnog hardvera i softvera za resavanje problema.

102. Navedite osnovne aspekte dizajna softverskih sistema. (4) Šta obuhvata ARHITEKTONSKI DIZAJN?
1.Konceptualni dizajn
2.Tehnicki dizajn
3.Arhitektonski dizajn
4.Detaljni dizajn
Arhitektonski dizajn se bavi izborom strategije, resavanje i modularizacijom sistema.

103. Navedite osnovne aspekte dizajna softverskih sistema. (4) Šta obuhvata DETALJNI DIZAJN?
1.Konceptualni dizajn
2.Tehnicki dizajn
3.Arhitektonski dizajn
4.Detaljni dizajn
Detaljan dizajn se bavi formulacijom detaljnih algoritama i strukture podataka neophodne za implementaciju sistema.

104. Kakva je uloga Dekompozicije i modularnosti u sklopu arhitekture rešenja?
-Dekompozicijom se dolazi do unutrasnje strukture slozenih sistema. Modularnost predstavlja karakteristiku koja govori da je resenje sastavljeno od modula koji kooperiraju u cilju implementacije funkcija sistema.

105. Šta je osnov konstrukcije u slučaju dekompozicije bazirane na modulima?
-U sklopu ovog pristupa konstrukcija se bazira na alokaciji funkcija komponentama pocevsi od najviseg nivoa apstrakcije pa do elementarne (atomicke) funkvionalmosti.

106. Šta je osnov konstrukcije u slučaju dekompozicije bazirane na podacima?
-Kod ovog pristupa dizajn se rukovodi spoljasnjim strukturom podataka. Najvisi nivo apstrakcije obuhvata generalnu strukturu podataka dok najnizi nivoi opisuju elemente podataka.

107. Šta je osnov konstrukcije u slučaju dekompozicije bazirane na događajima?
-Kod ovog pristupa akcenat se stavlja na dogadjaje i rukovanje dogadjajima odnosno nacim na koji dogadjaji menjaju stanje sistema.

108. Šta je osnov konstrukcije u slučaju dekompozicije bazirane na odnosu ulaz/izlaz?
-Kod ovog pristupa akcenat se stavlja na ulaze i izlaze sistema (sistem se posmatra kao crna kutija).

109. Šta je osnov konstrukcije u slučaju dekompozicije bazirane na objektima?
-Kod ovog pristupa akcenat se stavlja na objekte sa kojima sistem radi, njhove apstrakcije (klase) i veze medju njima.

110. Kada kažemo za sistem da je modularan?
-Ako se svaka aktivnost u sistemu odvija uz podrsku samo jedne komponente sa jasno definisanim ulazima i izlazima.

111. Kada za komponentu kažemo da je dobro definisana?
-Ako i samo ako si svi njeni ulazi esencijalni sa aspekta funkcionalnosti koju podrzava a svi njeni izlazi posledica neke od njenih (privatnih) akcija.

112. Navedite četiri osnovne karakteristike kvalitetnog dizajna. Šte definiše nezavisnost komponenti?
1.Nezavisnost komponenti
2.Identifikacija i rukovanje izuzecima
3.Otpornost na greske i rukovanje greskama
4.Smanjenje slozenosti

113. Šta predstavlja snaga veza među modulima?
1.Nepovezani – nizak nivo zavisnosti
2.Veza preko podataka – parametri
3.Veza preko srukture podataka
4.Veza preko kontrola – parametri obezbedjuju prenos kontrole
5.Veza preko zajednickih elemenata – globalne reference ili globalne vrednosti
6.Veze preko sadrzaja – jedna komponenta modifikuje drugu – visok nivo zavisnost

114. Šta predstavlja unutrašnja funkcionalna snaga modula?

1.Funkcionalna – jedan modul jedna funkcija

2.Sekvencijalna kohezije – jedan modul sadrzi vise funkcija koje se sekvencijalno koriste tako sto se izlaz iz jedne prosledjuje na ulaz druge.

3.Komunikaciona – modul sadrzi vise funkcija koje su povezane preko zajednickog repozitorijuma eksternog za modul.

4.Procedurna – modul sadrzi vise funkcija koje se moraju izvrsiti u utvrdjenom redosledu.

5.Temporalna – modul sadrzi vise funkcija cije izvrsenje vremenski povezano.

6.Logicka – ima smisla da budu zajedno.

7.Koicidentna – slucajno su zajedno.

115. Kakva je razlika između GREŠKE i IZUZETKA?

-Izuzetak su pojave koje predvidimo i znamo kako da reagujemo kada se dese,dok kod greske neznamo razlog desavanja i moze doci i do kriticnih problema za softver.

116. Na koja od tri načina je moguće opslužiti identifikovani izuzetak?

-Ponovni pokusaj – prvobitno stanje se restuira i ponovi servis uz koriscenje druge strategije.

-Korekcija – prvobitno stanje se restuira, izvrsi se korekcija nekog od elemenata servisa i ponovi servis uz koriscenje iste strategije.

-Izvestaj o neuspehu – prvobitno stanje se restuira, izvesti se kompnenta za rukovanje greskama i servis se blokira.

117. Šta podrazumeva PREVENCIJA GREŠAKA?

-Dizajn uparen sa testiranjem i verifukacijom.

118. Šta podrazumeva IZBEGAVANJE OTKAZA?

-Nadzor i sprecavanje otkaza.

119. Šta podrazumeva DETEKCIJA GREŠAKA?

-Otkrivanje gresaka na osnovu manifesacije otkaza.

120. Šta podrazumeva KOREKCIJA-ISPRAVKA GREŠAKA?

-Uklanjanje gresaka, oporavak od posledice otkaza, testiranjem verifikacija i ponovno aktiviranje.

121. Šta podrazumeva OTPORNOST NA GREŠAKE?

-Podrazumeva dizajniranje softvera tako da izoluje stetu uzrokovanu otkazom, dinamicki reorganizuje servise, obezbedi oporavak ili redundantni servis i nastavi podrsku korisnicima.

122. U čemu se sastoji SMANJENJE SLOŽENOSTI?

-Pojednostavljuje strukture softverskog sistema i strukture podataka do nivoa potpune funkcionalnosti uz minimalnu slozenost. Predstavlja izuzetan izazov za fazu konstrukcije softvera u kojoj cesto dolazi do infiltriranja i kumuliranja neracionalnosti i neefikasnosti kao posledica ad-hok intervencija.

123. Navedite četiri esencijalna zahteva prema softverskom sistemu?

1.Ispravno funkcionisanje (ako sistem ne funkcionise ostali elenemti nisu bitni)

2.Pouzdanost, raspolozivost, bezbednost i integritet

3.Standardnizacija., integracija, konzistentnost i prenosivost

4.Vreme isporuje i troskovi

124. Navedite osnovne postulate kvalitetnog GUI dizajna.(6) Šta se podrazumeva pod KORISNIČKOM ORIJENTISANOŠĆU?

1.Korisnicka orijentisanost

2.Konzistentnost

3.Personalizacija i podesavanje

4.Podrska ekspeimentisanju i oporavku

5.Povratna indikacija

6.Estetski aspekti i mogucnosti koriscenja

Korisnicka orijentisanost ne podrazumeva da program u celosti zamenjuje posao korisnika vec da korisnik interaktivno obavlja posao : korisnickim akcijama(meniji), preuzimanjem kontrole(dijalozi,servisi), povratna informacija (feedback).

125. Navedite osnovne postulate kvalitetnog GUI dizajna.(6) Šta se podrazumeva pod KONZISTENTNOŠĆU?
1.Korisnicka orijentisanost
2.Konzistentnost
3.Personalizacija i podesavanje
4.Podrska ekspeimentisanju i oporavku
5.Povratna indikacija
6.Estetski aspekti i mogucnosti koriscenja

Pod konzistentnosti se podrazumeva da je interfejs izradjen po preporukama postojecih masovno koristenih okruzenja (Microsoft,Apple), takodje da koristi ista obelezavanja i fontove za istu grupu operacija, postuje konvencije imenovanja i oznacavanja.

126. Navedite osnovne postulate kvalitetnog GUI dizajna.(6) Šta se podrazumeva pod PERSONALIZACIJOM I PODEŠAVANJEM?
1.Korisnicka orijentisanost
2.Konzistentnost
3.Personalizacija i podesavanje
4.Podrska ekspeimentisanju i oporavku
5.Povratna indikacija
6.Estetski aspekti i mogucnosti koriscenja

Personalizacija predstavlja opciju kojom korisnik sam podesava korisnicki interfejs prema svojim afinitetima. Podesavanja se vezuju za prilagodjavanje softvera razlicitim grupama korisnika.

127. Navedite osnovne postulate kvalitetnog GUI dizajna.(6) Šta se podrazumeva pod PODRŠKOM EKSPERIMENTISANJU i OPORAVKU?
1.Korisnicka orijentisanost
2.Konzistentnost
3.Personalizacija i podesavanje
4.Podrska ekspeimentisanju i oporavku
5.Povratna indikacija
6.Estetski aspekti i mogucnosti koriscenja

Najbitniji aspekat kod novih korisnika. Omogucuje im nesmetano eksperimentisanje kroz program. Realizuje se viselevelno undo naredbom koja vraca siste u prethodno stanje do pocetnog. U pojedinim softverima nije pozeljna implementacija undo naredbe(uzimanje novca iz bankomata).

128. Navedite osnovne postulate kvalitetnog GUI dizajna.(6) Šta se podrazumeva pod POVRATNOM INDIKACIJOM?
1.Korisnicka orijentisanost
2.Konzistentnost
3.Personalizacija i podesavanje
4.Podrska ekspeimentisanju i oporavku
5.Povratna indikacija
6.Estetski aspekti i mogucnosti koriscenja

Povratna indikacija obavestava korisnika o radu sistema nakon sto je on izdao naredbu. Projektant treba da obezbedi vizuelne ili aditivne efekte kojim se naznacava da je sistem zauzet.Potrebno je proceniti vreme izvrsavanja radnje i prema njemu postaviti odgovarajucu naznaku indikacije.

129. Navedite osnovne postulate kvalitetnog GUI dizajna.(6) Šta se podrazumeva pod ESTETSKIM ASPEKTIMA I MOGUĆNOŠĆU KORIŠĆENJA?
1.Korisnicka orijentisanost
2.Konzistentnost
3.Personalizacija i podesavanje
4.Podrska ekspeimentisanju i oporavku
5.Povratna indikacija
6.Estetski aspekti i mogucnosti koriscenja

Pod estetikom se podrazumeva vizuelna pojava gui interfejsa, koja podleze skupu kriteriijuma. Pod mogucnoscu koriscenja podrazumeva se : lakoca, jednostavnost, efikasnost, pouzdanost i produktivnost pri koriscenju softverskog sistema.
(Lepota je u oku onog koji posmatra!).

130. Šta predstavlja ARHITEKTURA SOFTVERA?

-Predstavlja opis podsistema i komponenti softverskog sistema zajedno sa njihovim medjusobnim vezama. Podsistemi i komponente mogu biti specificirane iz vise uglova sa ciljem ilustrovanja funkcionalnosti i nefunkcionalnosti osobina softverskog sistema. Softverska arhitektura sistema predstavlja proizvod koji je posledica aktivnosti proektovanja softvera.

131. Šta obuhvata ARHITEKTURA SOFTVERA?

1.Opis gradivnih elemenata softverskog sistema
2.Interakcija medju elementima
3.Sablone koji upravljaju kompozicijom elemenata softverskog sistema.
4.Ogranicenja vezana za sablone.
5.Sistem se moze konstruisati kao kompozitni element pri dizajnu drugih sistema.

132. Šta definiše ARHITEKTONSKI STIL?

-Arhitektonski stil definise familiju sistema uz oslonac na sablone koju grade organizacionu strukturu. Arhitektonski stil definise:

-recnik komponenti i tipove konektora
-ogranicenja vezana za nacin povezivanja komponenti
-jedan ili vise semantickih modela koji specificiraju kako se opsta svojstva sistema mogu izgraditi na bazi svojstava sastavnih delova.

133. Koja je osnovna karakteristika HETEROGENIH ARHITEKTURA?

-Kombinacija vise stilova. Komponente hijerarhijkog sistem mogu internu strukturu razviti na bazi razlicitih metoda. Konektori mogu biti dekomponovani na druge sisteme (npr. cevi mogu biti interno implementirane kao FIFO redovi cekanja). Pojedinacne komponente mogu koristiti mesavinu aritektonskih konentora.

Primer : Unix pipes-and-filters sistem – sistem datoteka se ponasa kao repozitorjum, - prima kontrole preko inicijalnih prekidaca – sa drugim konponentama integrira preko cevi.

134. Navedite osnovne prednosti DOGAĐAJIMA UPRAVLJANIH SISTEMA.

-Dekompozicija problema.
-razdvojeno je izracunavanje od kordinacije
-Odrzavanje sistema i ponovno koriscenje
-nema statickih zavisnosti u domenu imenovanja
-evolucija sistema je jednostavnija
-integracija je jednostavnija
-Persformansa
-pozivi se mogu paralelizovati

135. Navedite osnovne mane DOGAĐAJIMA UPRAVLJANIH SISTEMA.

-Dekompoziciju problema
-nema upravljanja redosledom pozivanja
-razmena podataka
-tesko je obezbediti ispravno funkcionisanje (testiranje je kompleksno)
-Odrzavanje sistema i ponovno koriscenje
-zahteva centralnu evidenciju koja sadrzi informacije o tome ko sta zna
-Persformansa
-indirekcija/komunikacija uticu na smanjenje persformanse

136. Koje elemente obuhvata savremeni pristup u razvoju softvera?

-Obuhvata: dizajn šablone, komponente, arhitekture softvera i radna okruženja (frameworks).

137. Koje elemente obuhvata tradicionalni pristup u razvoju softvera?

-Obuhvata: programsku paradigmu (varijable i strukture) i strukturu problema (klase, objekti i tipovi podataka).

138. Šta je osnovna princip programske paradigme i šta ona specificira?

-Osnovni princip je princip programiranja koji obuhvata promenljive i strukture programa. Podrazumeva razvoj produkata čiji je osnovni zadatak da obezbede izračunavanje nečega. Programska paradigma specificira različite načine programiranja.

139. Kakva je uloga programske paradigme u brzoj izradi softvera?

-Programska paradigma je vezana za proces automatske konstrukcije softvera i najviše se koristi u procesu konstrukcije prevodilaca i interpretera.

140. Kakva je uloga strukture problema u procesu brzog razvoja softvera?

-Ovaj nivo apstrakcije omogućava razmatranje krupnijih gradivnih elemenata programskog rešenja, njihovo predstavljanje i automatsku konverziju na elemente programske paradigme na bazi metoda programiranja (objektno, strukturalno, funkcionalno i sl.). Modelovanje statičke strukture problema odnosno rešenja stvara podlogu za modelom upravljanu konstrukciju programskog koda.

141. Na koje osnovne tri grupe se mogu podeliti šabloni? Objasnite ulogu arhitektonskih šablona.

-Arhitektonski sablon
-Dizajn sablon
-Konceptualni sablon
-Programski sablon
-Idiomi

Arhitektonski sablon predstavlja osnovnu strukturu organizacje softverskih sistema.

142. Na koje osnovne tri grupe se mogu podeliti šabloni? Objasnite ulogu dizajnerskih šablona.

-Arhitektonski sablon
-Dizajn sablon
-Konceptualni sablon
-Programski sablon
-Idiomi

Dizajn sablon obezbedjuje semu za rafiniranje podsistema, komponenti softverskog sistema, ili njihovih veza. Opisuje uobicajne strukture koje se cesto javljaju kod komponenti koje saradjuju u cilji resavanja opstih problema (generickih) u sklopu posmatranog (posebnog) konteksta.

143. Na koje osnovne tri grupe se mogu podeliti šabloni? Objasnite ulogu idioma.

-Arhitektonski sablon
-Dizajn sablon
-Konceptualni sablon
-Programski sablon
-Idiomi

Idiom je sablom niskog nivoa karakteristican za korisceni programski jezik. Idiom opisuje nacin implementacije pojedinacnih aspekata komponenti ili njihovih veza, uz oslonac na svojtva koriscenog programskog jezika.

144. Da li su šabloni vezani isključivo za objektnu platformu?Obrazložite odgovor.

-Nisu. Sabloni su predlozeni formati za kojima se moze predstaviti resenje tj to su uputstva za izradu proizvoda slicnog onome koji vec postoji i proverono je dobar.Sablon opisuje neki problem sa kojim se stalno suocavamo i daje sustinski princip resavanja problema.

145. Da li šabloni nude samo jedno moguće rešenje?Obrazložite odgovor.

-Ne. Sabloni kao kosturi resenja nude sustinski princip resavanja a pojedinacni uzroci resenja ne moraju nikad da se ponove. Sabloni predstavljanju specifikaciju problema koji se cesto javlja a ne daju njegovo konkretno resenje.

146. Da li šabloni predstavljaju implementaciju rešenja?Obrazložite odgovor.

-Ne. Sabloni daju potencijalno moguca resenja. Opisuju kada, zasto i kako neko na osnovu njih moze kreirati implementaciju.

147. Da li svako rešenje predstavlja šablon?Obrazložite odgovor.

-Ne. Resenje mora postovati trojno pravilo tj mora da se identifikuje u najmanje tri razlicita problema i resenja. Tek tada se vrsi verifikacija fenomena i resenje postaje sablon.

148. Koje osobine treba da zadovoljava neko rešenje da bi moglo biti proglašeno za šablon?

-Mora biti pverifikovano kao ponavljajuce za problem koji se cesto ponavlja (trojno pravilo).

149. Koje esencijalne elemente poseduje svaki šablon? Šta predstavlja kontekst?

1.Kontekst
2.Problem
3.Resenje
Kontekst opisuje ponavljajuci skup scenarija u kojima je moguce primeniti posmatrani sablon.

150. Koje esencijalne elemente poseduje svaki šablon? Šta predstavlja problem?

1.Kontekst
2.Problem
3.Resenje
Problem se odnosi na skup prinuda tj ciljeva i ogranicenja koji se javljau u sklopu konteksta. U opstem slucaju problem daje odgovor na pitanja kada je moguce primeniti sablon.

151. Koje esencijalne elemente poseduje svaki šablon? Šta predstavlja rešenje?

1.Kontekst
2.Problem
3.Resenje
Resenje opisuje elemente koji predstavljaju sastavne delove sablona, veze medju njima, odgovornosti i modelitete saradnje.

152. Navedite dve osnovne grupe softverskih šablona. U čemu se one razlikuju?

1.Genereativni sabloni
2.Negenerativni sabloni
Generativni sabloni sluze za uoblicavanje arhitekture i za generisanje sistema (ili delova istog), tj oni pomazu pri generisanju drugih sablona.Negenerativni sabloni su pasivni i deskriptivni, posmatramo ih u sklopu postojecih sistema,ne daju nam tumacenje kako doci do novog sablona.

153. Kakva je razlika između klasnih i objektnih dizajn šablona?

-Razlika je u domenu:

- Ļ klasni šabloni koji se fokusiraju na relacije između klasa i podklasa (class patterns)
- Ļ objektni šabloni koji se fokusiraju na relacije između objekata (object patterns)

154. Navedite osnovne klase dizajn šablona. Obrazložite namenu kreacionih dizajn šablona.

-Kreacioni, strukturalni i šabloni ponašanja. Kreacioni (gradivni) šabloni apstrahuju proces instanciranja. Pomoću njih sistem postaje nezavisan od načina pravljenja, sastavljanja i predstavljanja objekata. Gradivni šabloni omogućavaju veliku fleksibilnost u smislu šta se pravi, ko to pravi, kako se pravi i kada.

155. Navedite osnovne klase dizajn šablona. Obrazložite namenu strukturalnih dizajn šablona.

-Kreacioni, strukturalni i šabloni ponašanja. Strukturalni šabloni se bave načinom na koji se klase i objekti sastavljaju u veće strukture.

156. Navedite osnovne klase dizajn šablona. Obrazložite namenu dizajn šablona ponašanja.

-Kreacioni, strukturalni i šabloni ponašanja. Šabloni ponašanja bave se algoritmima i raspodelom odgovornosti među objektima. Opisuju prirodu složenog toka kontrole koji se teško prati u vreme izvršavanja.

157. Kakva je namena i koji problem rešava dizajn šablon ***Factory Method?***

-Definise interfejs za kreiranje nekog objekta ali podklasi prepusta odluku o tome koju klasu ce instancirati. Resava problem dinamickog kreiranja uz oslonac na mehanizam nasledjivanja.

158. Navedite varijante implementacije dizajn šablon ***Factory Method?*** U čemu se one razlikuju?

1.Kada zelimo da kominiciramo sa instancom neke klase a jedino posedijemo svest o interfejsu ili o samoj roditeljskoj klasi.
2.Kada zelimo lokalizovati instanciranje objekta neke klase u cilju bolje kontrole koda.
Razlikuju se u ideju o koriscenju objekta i resavanju njegove vidljivosti.

159. Šta omogućava parametrizovani ***Factory Method?***

-Omogucuje da dellarisani proizvodni metod kreira vise vrsta proizvoda zavisno od vrednosti prosledjenog parametra.Koristi konstruktor sa parametrima.

160. Navedite osnovne prednosti dizajn šablona ***Factory Method?***

-Instanciranje konkretnih objekata je lokalizovano. Programski kod nije jako spregnut sa tipom konkretnih instanci. Klasa moze delegirati kreiranje konkretnih objekata koje koristi na svoje klase. U slucaju paralelnih hijerarhija mozemo lokalizovati znanje o tome koje klase su u vezi.

161. Navedite osnovne mane dizajn šablona ***Factory Method?***

-Posto je dizajn obrazac zasnovan na nasledjivanju (najcesce) Dolazi do pojave velikog broja klasa , jer svaki proizvod zahteva novog kreatora.

162. Navedite srodne dizajn šablone za ***Factory Method?***

-Abstract factory
-Template Method
-Prototip

163. Kakva je namena i koji problem rešava dizajn šablon ***Abstract Factory?***

-Generalizacija obrazca Factory method. Resava problem vise platformi ; jedan interfejs za kreiranje familije proizvoda.Klijent ne mora da zna konkretne implemlementacije.

164. Šta se dobija a šta gubi primenom dizajn šablona ***Abstract Factory ?***

-Dobijamo :
-Instanciranje familije konkretnih objekata je lokalizovano.
-Programski kod nije jako spregnit sa tipom konkretnih instanci.
-Promena familije konkretnioh instanci je moguca izmenom programskog koda na mestu gde se instancira konkretna fabrika objekata.
-U odredjenim slucajevima moguca je dinamicka promena fabrike objekata.
-Gubimo :
-Zbog nasledjivanja dobijamo relativno velik broj klasa.
-Svaki novi proizvod zahteva novog kreatora i svaka nova familija proizvoda zahteva novu fabriku i nove konkretne proizvode.

165. Kakva je namena i koji problem rešava dizajn šablon ***Builder?***

-Resava problem parsiranja slozenog modela ili opisa i kreiranje jednog od vise mogucih ciljnih objekata. Razdvaja konstrukciju slozenih objekata od njihovog modela tako da isti proces konstrukcije moze da kreira razlicite reprezentacije. Obezbedjuje potpuno razdvajanje podataka od nacina prikazivanja.

166. Šta omogućava primena ***Builder*** dizajn šablona?

-Omogucuje menjanje unutrasnje predstave proizvoda
-Izoluje kod za izgradnju od koda za predstavljanje
-Omogucuju finiju kontrolu procesa izgradnje.

167. Navedite srodne dizajn šablone za ***Builder*** dizajn šablon***?***

1.Absract factory
2.Composite

168. Kakva je namena i koji problem rešava dizajn šablon ***Prototype?***

-Specifira vrste objekata koje je potrebno kreirati uz oslonac na instancu-protorip, o kreora novi objekat kopiranjem protopipa. Prototip spada u objek-kreacione sablone. Instanciranje se obicno elegira drugom objektu.

169. Kada je pogodno primeniti dizajn šablon ***Prototype?***

-Koristimo ga da bi se izbeglo kreiranje vise podklasa u cilj instanciranja vise vrsta istog objekta. Klijent jedino treba da bude u interakciji sa osnovnim prototipom koji onda utvrdjuje vrstu i poziva poeraciju kloniranja konkretnog prototipa.

170. Koje su prednosti primene dizajn šablona ***Prototype?***

-Skriva konkretne klase koje generisu proizvod.
-Dozvoljava klijenutu da se skoncentrise na razvoj klasa koje su specificirane za razvijanu aplikacijuj bez rizika da ih kasnije mora menjati
-Dodavanje i ukljnjanje produkta u fazi izvrsavanja.

171. Koje su mane primene dizajn šablona ***Prototype?***

-Svaka podklasa mora implementirati Clone() operaciju sto nije uvek jednostavno (kada klasa vec postoji to je jako tesko).

172. Navedite srodne dizajn šablone za ***Prototype*** dizajn šablon***?***

1.Abstract factory
2.Factory method
3.Composite
4.Decorator

173. Kakva je namena i koji problem rešava dizajn šablon ***Singletone?***

-Obezbedjuje da klasa ima samo jednu instancu i pruza globalno dostupnu tacku preko koje se pristupa toj instanci.

174. Koje su prednosti primene dizajn šablona ***Singletone?***

-Kontrolisani pristup jednom primerku
-Smanjenje prostora imena
-Omiogucava unapredjenje operacija i predstavljanja

175. Koje su mane primene dizajn šablona ***Singletone?***

-Posto obezbedjuje jedinstvenu pristupni tacku servisa, dolazi do skrivenih zavisnosti potencijalno bilo gde u kodu.Takodje veze nisu vidkjive analizom interfejsa klasa koje koriste singeltone.Jedna klasa preuzima sve odgovornosti.Testiranje.Objekat nosi stanje.

176. Navedite osnovne šablone koji se mogu implementirati uz oslonac na dizajn šablon ***Singletone?***

1.Abstract factory
2.Builder
3.Prototype

177. Kakva je namena i koji problem rešava dizajn šablon ***Adapter?***

-Konvertuje interfejs klase u drugi koji klijentsja klasa ocekuje.Omogucuje da klase koje nisu dizajnirane da saradjuju posredstvom njega obavljaju saradnju.

178. Kakva je razlika između klasnog i objektnog Adaptera?

-Klasni adapter prilagodjava adaptiranu i ciljnu klasu.Ukoliko klasa poseduje hijerarhiju naslednika adapter nadjacava adaptirane klase, tj nema adaptiranja podklasa.Objektni adapter adaptira osnovnu podklasu i sve njene podklase, poseduje mehanizam za pristup podklasama.

179. Navedite osnovne načine implementacije Adaptera. U čemu se oni razlikuju?

-Kreira se klasa koja poseduje zeljeni interfejs i ucini se da ona kominucira sa klasom koja poseduje drugaciju interfejs.Ovo se ostvaruje :
-mehanizmom nasledjivanja
-Kreiranjem odgovarajuce kompozicije objekata

Mehanizam nasledjivanja podrazumeva izvodjenje nove klase iz one koja nam odgovara i dodavanje metoda neophodnih da bi novoizvedena klasa zadovoljila zahteve zeljenog interfejsa.

Kreiranjem odgovarajuce kompozicije objekata podrazumeva se ukljucivanje klase unutar nove kalse a zatim kreiranje metoda koje prevode pozive unutar nave klase.

180. Navedite srodne dizajn šablone za ***Adapter*** dizajn šablon***.***

1.Bridge
2.Decorator
3.Proxy

181. Kakva je namena i koji problem rešava dizajn šablon ***Bridge?***

-Razdvaja apstrakciju od njene implementacije cime omogucuje njihovu nezavisnu izmenu.

182. Navedite osnovne situacije u kojima je moguće primeniti dizajn šablon ***Bridge.***

-Kada je potrebno da neka apstrakcija poseduje vise razlicitih implementacija. Rasporedjivanje podataka po nitima.

-Kada se zeli izbeci trajno vezivanje apstrakcije i njenih implementacija

-Kada je potrebno prosirivati apstrakciju i njene implementacije novim podklasama

-Kada promena u implementaciji ne sme da utice na klijente

-Kada se zeliti smanjiti brij kalsa u hijerarhiji

-Kada se u implementaciji deli vise objekata a da klijent toga nije svestan.

183. Navedite osnovne posledice primenite dizajn šablon ***Bridge.***

-Razdvajenje interfejsa od implementacije

-implementacija nije trajno vezana sa interfejsom

-implementacija apstrakcije se moze odgoditi do trenutka izvrsavanja

-objekat moze promeniti svoju implementaciju u fazi

-Jednostavno prosirivanje

-hijerarhija apstrakcija i implementacija se mogu nezavisno prosirivati.

-Skrivanje implementacionih detalja od klijenta

184. Obrazložite značaj razdvajanja interfejsa od implementacije.

-Pogodnosti su te sto nasledjivanjem interfejsa mozemo ga razlicito implementrati u zavisnosti od konkretne potrebe.

185. Navedite srodne dizajn šablone za ***Bridge*** dizajn šablon***.***

1.Abstract Factory

2.Adapter

186. Kakva je namena i koji problem rešava dizajn šablon ***Composite?***

-Rukovanje hijerarhijama koje predstavljaju strukturu „celina-deo-celina“. Uniformno rukovanje elementarnim i slozenim komponentama.

187. Navedite osnovne posledice primenite dizajn šablon ***Composite.***

-Definise hijerarhije klasa koje se sastoje od primitivnih i slozenih objekata.

-Primitivni objekti se mogu udruživati u složenije objekte, koji se rekurzivno mogu dalje strukturirati.

-Kad god klijentski kod očekuje primitivni objekat bez posledica može preuzeti i kompozitni objekat.

-Klijent se pojednostavljuje

-Klijent može tretirati kompozitne strukture i primitivne objekte UNIFORMNO.

-Klijent nema znanje (i ne treba da brine) da li radi sa Listom ili sa Sastavom

-Celokupno resenje moze postati generickije nego sto je potrebno

-Mana jednostavnog dodavanja komponenti je da se usložnjava sprečavanje uključivanja nekih komponenti u sastav:

-U tom slučaju su neophodne PROVERE U FAZI IZVRŠAVANJA.

188. Kakva je namena i koji problem rešava dizajn šablon ***Decorator?***

-Rukovanje hijerarhijama koje predstavljaju strukturu „celina-deo-celina“. Uniformno rukovanje elementarnim i slozenim komponentama.

189. Koje su prednosti primene dizajn šablona ***Decorator?***

-Fleksibilnije od statičkog nasleđivanja

-Odgovornosti je moguće dodavati i ukidati u fazi izvršavanja.

-Također je moguće jedno te isto svojstvo dodati VIŠE PUTA!

-Izbegava se uvođenje klasa koje opisuju opcije i dodatna svojstva na više nivoe hijerarhije.

-Nudi mogućnost dodavanja odgovornosti po potrebi.

-Umesto pokušaja obezbeđivanja podrške za sve moguće kombinacije proširenja u sklopu jedne složene klase, moguće je krenuti od jednostavne klase i DEKORISATI je po potrebi.

190. Koje su mane primene dizajn šablona ***Decorator?***

-Dekorater i njegova komponenta NISU IDENTIČNI

-Sa aspekta IDENTITETA OBJEKATA dekorisana komponenta nije identična originalnoj komponenti.

-Veliki broj jednostavnih objekata

-Sistemi koji su sastavljeni od velikog broja jednostavnih objekata koji svi liče jedan na drugog.

-Jednostavno se podešava ali ga je teško razumeti i TESTIRATI.

191. Kakva je namena i koji problem rešava dizajn šablon ***Facade?***

-Obezbedjuje jedinstvenost interfejsa za skup podsistema.

192. Kakva je namena i koji problem rešava dizajn šablon ***Flyweight?***

-Oslonac na mehanizam deljenja u cilju efikasnog podrzavanja rada sa velikim brojem jednostavnih objekata.

193. Kakva je namena i koji problem rešava dizajn šablon ***Proxy?***

-Resava problem distribuirane obrade tako sto obezbedjuje lokalnu predstavu jedne te iste komponente u razlicitom adresnom prostoru.