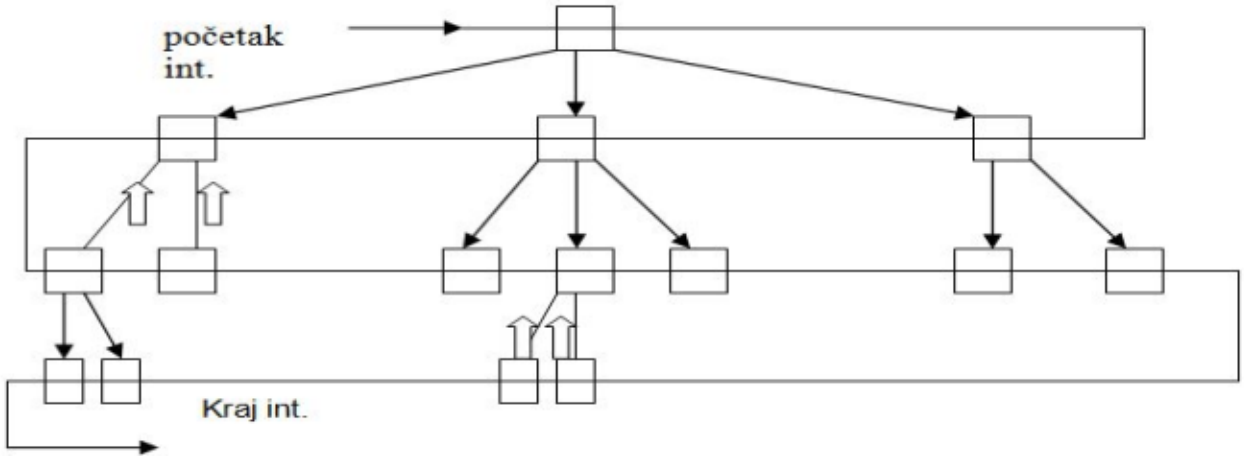
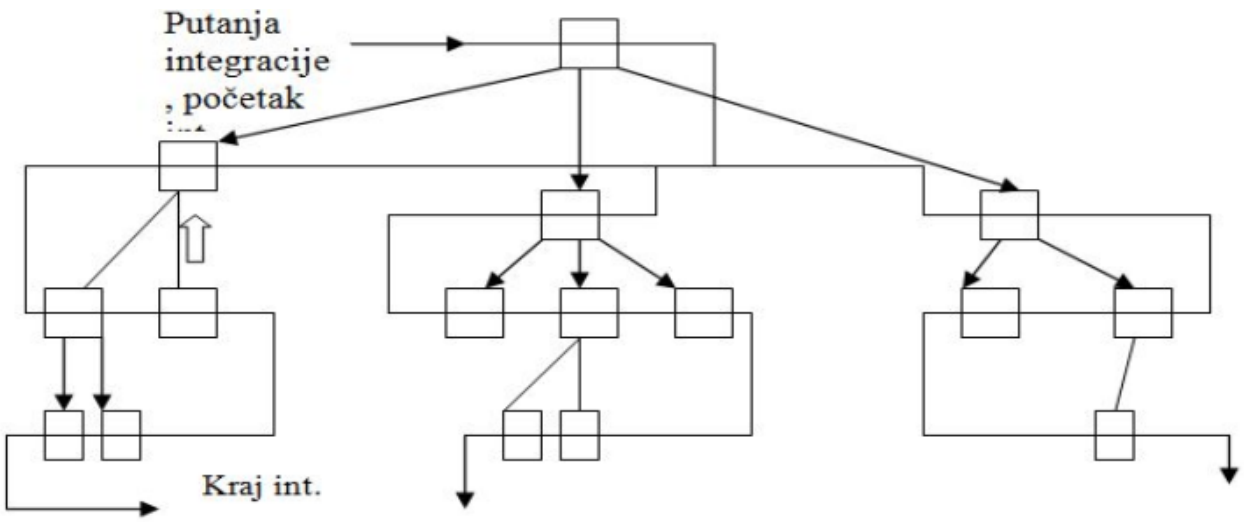
SE211-Ispitna pitanja

1. Navesti aktivnosti u procesu konstruisanja softvera.
2. Ukratko opisati aktivnosti u procesu konstruisanja softvera.
3. U čemu se ogleda iterativni koncept konstruisanja softvera?
4. Šta predstavljaju softverski standardi?
5. Zbog čega je potrebno da programi budu dobro dokumentovani?
6. Kako se postiže dobra vizuelna struktura programa?
7. Šta čini internu dokumentaciju softvera?
8. Opisati pravilno korišćenje varijabli kod programiranja.
9. Šta je važno kada su u pitanju tipovi podataka kod programiranja?
10. O čemu treba voditi računa kod upotrebe petlji?
11. Kako se određuje kompleksnost potprograma?
12. Navesti osnovne koncepte strukturnog (struktuiranog) programiranja.
13. Navesti osnovne koncepte proceduralnog programiranja.
14. Koje su aktivnosti karakteristične za životni ciklus softvera?
15. Navesti modele životnog ciklusa softvera i opisati jedan po izboru.
16. Opisati OO model životnog ciklusa softvera.
17. Šta se ubraja u pripreme za konstruisanje softvera?
18. Navesti i ukratko opisati ključne odluke za konstruisanje softvera.
19. Kako izgleda konstruisanje softvera kod malih, a kako kod velikih projekata?
20. Upravljanje konstruisanjem.
21. Upravljanje konfiguracijom.
22. Navesti osnovne karakteristike dizajnerskih alata i njihovu svrhu korišćenja.
23. Šta omogućuje dobar IDE editor?
24. Šta su refaktoreri?
25. Šta se podrazumeva pod konstrukcionim testiranjem?
26. Navesti tehnike testiranja softvera.
27. Opisati Black box tehniku testiranja softvera.
28. Opisati White box tehniku testiranja softvera.
29. Opisati Gray box tehniku testiranja softvera.
30. Navesti nivoe testiranja softvera.

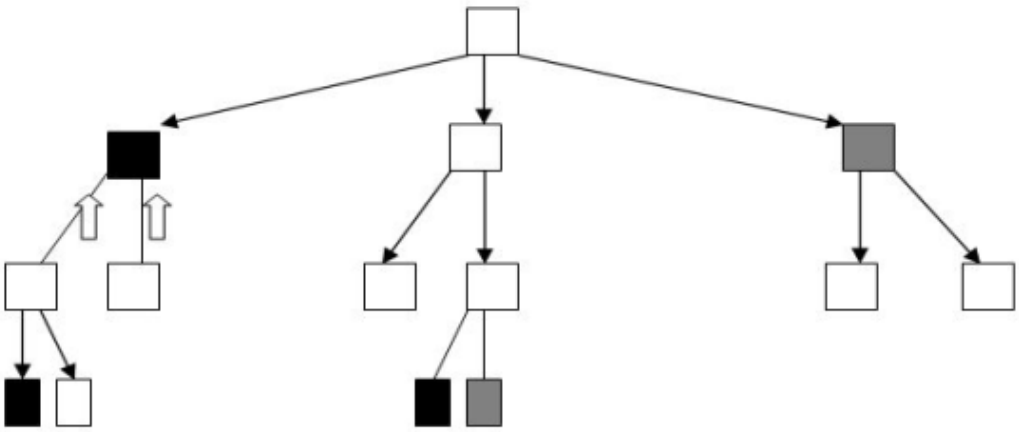
31. Opisati jedinično testiranje i navesti alate.
32. Šta je karakteristično za itegraciono testiranje?
33. Kada se sprovodi sistemsko testiranje?
34. Nabrojati alate za podršku testiranju i objasniti jedan po izboru.
35. Uporediti ručno i automatizovano testiranje softvera.
36. Navesti i opisati kategorije grešaka u programu.
37. Nabrojati tehnike za pronalaženje i korigovanje grešaka.
38. Navesti alate za debagiranje.
39. Nabrojati komande alata za debagiranje i opisati dve po izboru.
40. Kako se realizuje strategija inkrementalne integracije?
41. Navesti alternativne strategije integracije.
42. Uporediti faznu i inkrementalnu integraciju.
43. Šta je prikazano na slici? Opisati ukratko.



44. Sledeća slika prikazuje jednu od strategija integracije. Koju? Opisati.



45. Na slici je prikazana antirizična integracija. Kako ona funkcioniše?



46. Opisati funkcionalnu integraciju.
47. Koja su dva osnovna prilaza u izvršavanju integracionog testiranja? Objasniti ukratko.
48. Navesti korake koje je potrebno preduzeti pri integracionom testiranju.
49. Objasniti hijerarhijsku strukturu softvera (pet nivoa).
50. Kakva treba da je komunikacija između pojedinih podsistema? Zašto?
51. Navesti i ukratko opisati faze dizajna.
52. Na primeru objasniti dizajn podataka.

53. Objasniti arhitektonski dizajn kroz primer.
54. Objasniti interfejsni dizajn kroz primer.
55. Konceoti OO dizajna.
56. Artifakti OO analize i dizajna.
57. Šta se podrazumeva pod detaljnim dizajnom?
58. Šta sadrže dokumenti dizajna?
59. Vrste modela kod komponentnog dizajna.
60. Koji su ciljevi dizajna?
61. Navesti nekoliko tehnika koje se koriste kod dizajna.
62. Objasniti koncept „rafiniranja“.
63. Kakva je to vertikalna, a kakva horizontalna podela u okviru modularne tehnike dizajna?
64. Prednosti i nedostaci top-down i bottom-up dizajna.
65. Opisati reupotrebu softvera.
66. Navesti korake koji se preduzimaju kod dizajna klase.
67. Kako izgleda procedura konstruisanja klase?
68. Definisanje vidljivosti klase.
69. Šta podrazumeva dizajn visokokvalitetnih klasa?
70. Kohezivnost klase.
71. Šta predstavlja povezanost modula?
72. Zbog čega je potrebno kreirati klase?
73. Koji su razlozi za eliminaciju klase?
74. Zbog čega su potrebne apstraktne klase? Navesti primer.
75. Mapiranje klasnih dijagrama i sekvencijalnih dijagrama.
76. Šta prikazuje implementacioni klasni dijagram?
77. Kakva je veza između implementacionog sekvencijalnog dijagrama i programa?
78. Konzistentnost klasnog dijagrama i sekvencijalnog dijagrama – objasniti ukratko.
79. Od čega se sastoji konstruisanje potprograma?
80. Šta čini visoki dizajn potprograma?
81. PPP (Pseudocode Programming Process) – objasniti na primeru.
82. Navesti alternative za PPP.

83. Tehnike refaktorisanja programa tj. potprograma.
84. Navesti i ukratko objasniti korake pri kodiranju potprograma.
85. Doterivanje programskog koda.
86. Koji su razlozi za formiranje potprograma?
87. Koje su karakteristke lošeg, a koje dobrog (kvalitetnog) potprograma?
88. Šta obuhvata optimizacija kodiranja?
89. U čemu je značaj dizajna algoritama?
90. Šta je predmet razmatranja kod detaljnog dizajna?
91. Navesti i ukratko opisati atribute dizajna.
92. Na osnovu kog kriterijuma možemo razmatrati kvalitet dizajna?
93. Klasifikacija kuplovanja.
94. Koji je cilj merenja (metrika) OO softvera?
95. Opisati WMC metriku.
96. Opisati DIT metriku.
97. Opisati NOC metriku.
98. Opisati CBO metriku.
99. Opisati RFC metriku.
100. Šta je refaktorisanje programa i koje su njegove prednosti?
101. Navesti nekoliko tehnika refaktorisanja programa.
102. Kada treba izvršiti refaktorisanje programa?
103. Šta uključuje refaktorisanje na nivou potprograma?
104. Šta uključuje refaktorisanje na nivou klase?
105. Bezbednost refaktorisanja.
106. Navesti i ukratko opisati vrste kohezija potprograma.
107. Koje su to neprihvatljive vrste kohezija? Zašto?
108. Navesti preporuke za parametre potprograma.
109. Koji se alati koriste za refaktorisanje programa ili potprograma?
110. Objasniti upotrebu assert funkcija.
111. Šta je karakteristično za robustni program?
112. Izuzeci, rukovanje izuzecima.
113. Čime se bavi validacija?
114. Kakvo je to defanzivno programiranje?
115. Kako zaštititi program od loših ulaznih podataka?

116. Objasniti postavljanje “barikada” u program.
117. Kojim tehnikama se obezbeđuje kvalitet izvornog koda?
118. Navesti metode za detekciju grešaka i objasniti ukratko dve metode po izboru.
119. Koje tehnike se koriste pri konstrukcionoj verifikaciji softvera?
120. Kako se može obezbediti kvalitet softvera?
121. Opisati statičku analizu programa.
122. Šta se podrazumeva pod dinamičkom analizom programa?
123. Koje osobine utiču na kvalitet proizvoda (softvera)?
124. Dati pregled konstrukcionih tehnologija.
125. Opisati primenu tabelarno-bazirane metode.
126. Šta spada u code-tuning tehnike?
127. Kako se može povećati brzina programa bez upotrebe code-tuning tehnika?
128. Opisati ukratko kako treba postupiti pri otkrivanju da je neki program spor.