1. Odaberite tri kljucna mehanizma za ostvarivanje OO programiranja
    - Enkapsulacija
    - Nasljedjivanje
    - Polimorfizam
2. Koji sintaksicki element, odnosno, koja je naredba u programskim jezicima Basic, Cobol i Fortran remetila strukturne principe progamiranja?
    - GOTO
3. Softversko inzinjerstvo se bavi iskljucivo implementacijom softvera.
    - Netacno
4. Od ponudjenih, odaberite objektno orijentisane programske jezike.
    - C++
    - Java
5. Povezite odgovarajuce koncepte, termine i definicje
    - Mehanizam koji povezuje naredbe i podatke sa kojima te naredbe rade…ENKAPSULACIJA
    - Osobina koja omogucava da se jedan nacin pristupa koristi za opstu…. POLIMORFIZAM
    - Proces kojim jedan objekat dobija svojstva drugoga, uz mogucnost…. NASLJEDJIVANJE
6. Za koji od ponudjenih programskih jezika pojava WWW servisa predstavlja kljucni factor za razvoj?
    - Java
7. Odaberite strukturne programske jezike (koji nijesu objektno orijentisani).
    - C
    - Pascal
8. Posmatranje slozenih objekata u vidu cjeline koja se ponasa na odredjeni nacin oznacava se pojmom:
    - Apstrakcija
9. Oznacite prihvatljive definicije softverskog inzinjerstva (SI,SE):
    - SI je primjena sistematskih, disciplinovanih, mjerljivih pristupa razvoju,…..
    - Softversko inzinjerstvo je definisani, korak-po-korak proces,…….
10. Kako se naziva osnovni koncept enkapsulacije u vecini programskih jezika?
    - Klasa

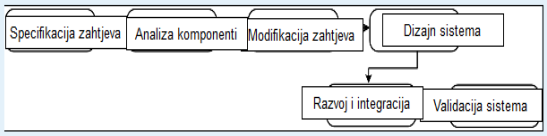
-DRUGA LEKCIJA-

11. Koje su osnovne aktivnosti softverskog inzinjerstva?
    - Specifikacija softvera
    - Validacija softvera
    - Evolucija softvera
    - Razvoj softvera
12. Softverski sistemi koji sluze za kontrolisanje hardvera nazivaju se:
    - Embedded upravljacki sistemi
13. Softversko inzinjerstvo treba da podrzi:
    - Profesionalni razvoj softvera
14. Odaberite apekte uticaja WWW servisa internet na razvoj softverskog sistema
    - Web servisi treba da se razvijaju i isporucuju inkrementalno
    - Visestruko koriscenje je postal dominantan pristup,…..
    - Korisnicki interfejsi su ograniceni funkcionalnostima,……
15. Odrzavanje softvera je skuplje od njegovog razvoja
    - Tacno
16. Izvrsavanje na udaljenom racunaru a pristupanje pomocu PC-a odnosno terminal, kao na primjer, u slucaju e-commerce alata, karakterise:
    - Interaktivne aplikacije zasnovane na transakcijama
17. Programski sistemi koji se pokrecu na lokalnom racunaru (PC) i tipicno sve funkcionalnosti nude offline nazivaju se:
    - Stand-alone aplikacije
18.
    - Evolucija softvera – Promjena softvera kako bi reflektovao izmijenjene zahtjeve kupaca i trzista.
    - Specifikacija softvera – Kupci i inzinjeri definisu softver koji treba napraviti, kao i ogranicenja u njegovom funkcionisanju.
    - Razvoj softvera – Dizajn i kodiranje
    - Validacija softvera – Provjerava da li softver zadovoljava zahtjeve kupaca

-TRECA LEKCIJA-

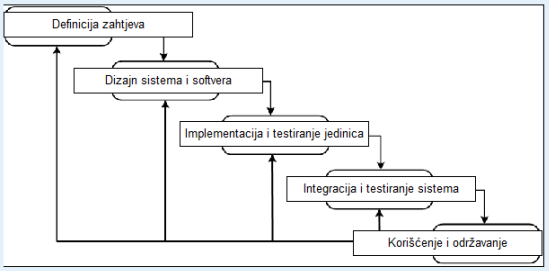
19. Popunite blok-dijagram vezan za inzinjerstvo ponovnog koriscenja.

Popunite blok-dijagram vezan za inženjerstvo ponovnog korišćenja.



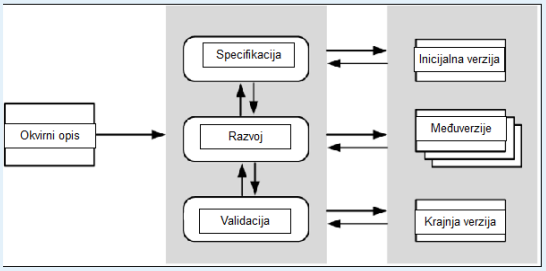
20. Prevlacenjem odgovarajucih tekstualnih blokova, popunite model vodopada,….

Prevlačenjem odgovarajućih tekstualnih blokova, popunite model vodopada prikazan na slici:



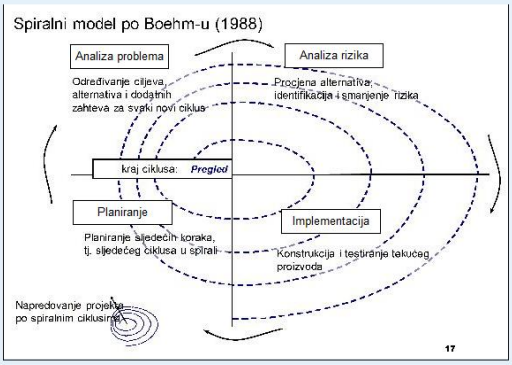
21. Prevlacenjem popunite prazne blokove dijagrama modela inkrementalnog razvoja

Prevlačenjem popunite prazne blokove dijagrama modela inkrementalnog razvoja.

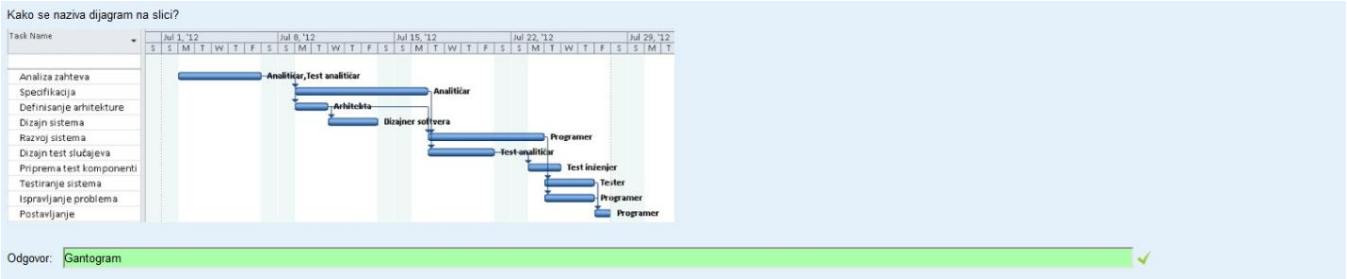


22. Dopunite sljedeci model razvoja softvera zadatim blokovima:

Dopunite sljedeći model razvoja softvera zadatim blokovima:



23. Kako se naziva dijagram na slici?



24. Povezite termine i odgovarajuća objašnjenja

- Obuhvata period od definicije zahtjeva do prestanka upotrebe – Zivotni ciklus softvera,
- Koherentni (povezani, spojeni) set aktivnosti koji vodi proizvodnji softverskog proizvoda – Softverski proces,
- Opisuje procese iz razvoja, koriscenja i održavanja softverskog proizvoda u toku njegovog zivotnog ciklusa – Model zivotnog ciklusa

25. Povezite aktivnosti i odgovarajuce motivacije svake od aktivnosti

- Softver mora evoluirati, kako bi mogao zadovoljiti promjenljive potrebe njegovih kupaca (narucilaca) – Evolucija softvera,
- Napravljeni softver mora biti provjeren, tj.mora biti izvrsena njegova validacija kako bi se obezbijedilo da on radi ono što je kupac želio – Validacija softvera,
- Softver sa zadatim specifikacijama mora biti napravljen – Dizajn i implementacija softvera,
- Funkcionalnost softvera i ograničenja njegovih operacija moraju biti definisani – Specifikacija softvera

26. Softverski proces je:

- koherentni (povezani, spojeni) set aktivnosti koji vodi proizvodnji softverskog proizvoda

27. Model razvoja softvera kod kojeg se stvara preklapanje aktivnosti specifikacije, razvoja i validacije, a sistem se razvija kao serija verzija, naziva se:

- Inkrementalni

28. Model razvoja softvera koji je zasnovan na postojanju značajnog broja upotrebljivih komponenti i kod kojeg je proces razvoja je orjentisan na integraciju ovih komponenti naziva se:

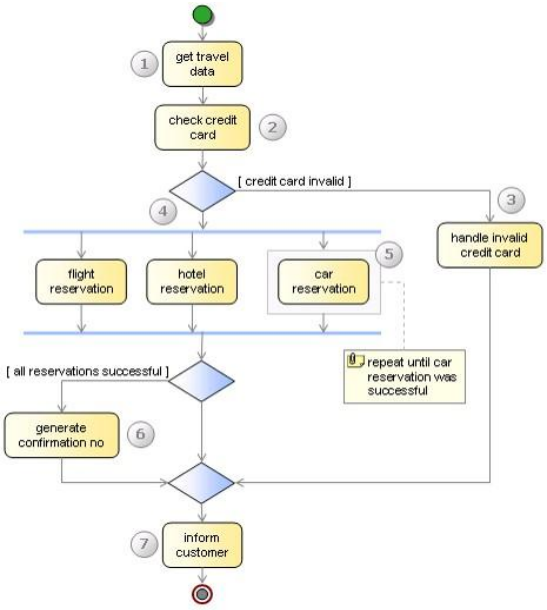
- Inznjerstvo ponovnog korišćenja softvera

29. Procesi kod kojih je planiranje aktivnosti inkrementalno nazivaju se:
    - Agilni procesi
30. Kod Spiralnog modela po Boehm-u, softverski proces je reprezentovan spiralom. Svaka petlja u spirali se može podijeliti na četiri sekcije
    - Analiza problema i postavljanje ciljeva,
    - Analiza rizika,
    - Razvoj i validacija,
    - Planiranje

-CETVRTA LEKCIJA-

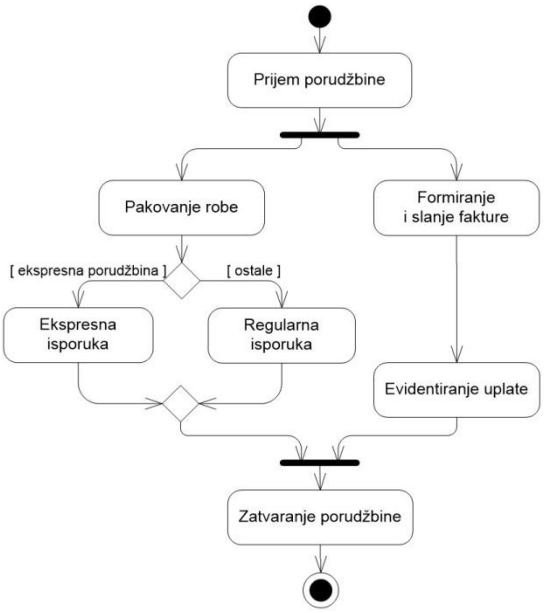
31. Odaberite stavke koje se odnose na aktera u Use case modelu:
    - Spoljni korisnik
    - Interni korisnik koji ima određenu ulogu u sistemu
    - Informacioni system, aplikacija,
    - Hardverski uređaj,
    - Drugi sistemi koji su u interakciji sa posmatranim sistemom

32. Sta se od navedenog odnosi na dijagram aktivnosti?
    - Detaljnu specifikaciju use case modela,
    - Detaljno opisuje slučajeve korišćenja, aktivnosti, scenarija, ponašanja,
    - Opisjue šta sistem radi, ali ne i kako sistem radi
33. Koji UML dijagram je prikazan na donjoj slici?
    - Dijagram aktivnosti



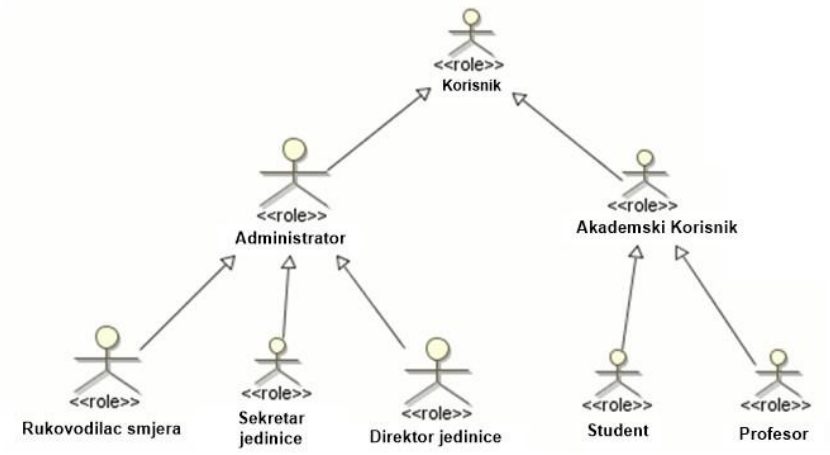
34. Za dijagram na slici odabrati tacne konstatacije:

- Pakovanje robe i Formiranje i slanje fakture su aktivnosti ili akcije,
- Pakovanje robe i Formiranje i slanje fakture se uvijek izvršavaju,
- Dio u obliku predstavlja kontrolni čvor



35. Sta je predstavljeno na donjoj slici?

- Primjer generalizacije



36. Odaberite pitanja koja mogu pomoci u odredjivanju aktera, kod Use-Case modela.

- Ko je zainteresovan za određeni zahtjev?,
- Gdje se u organizaciji koriste use case-ovi?,
- Ko će koristiti ili imati koristi od upotrebe aplikacije?,
- Ko će kreirati, ažurirati i brisati podatke?,
- Ko će pružati podršku i održavati sistem?,

- Da li ovaj sistem koristi spoljašnje resurse?,
- Da li jedna osoba ima nekoliko različitih uloga?,
- Da li nekoliko osoba ima istu ulogu?,
- Da li sistem uspostavlja interakciju sa naslijeđenim sistemom?

37. Veza izmedju aktera i Use case-a u UML Use Case modelu prikazuje se relacijom:
    - Asocijacije
38. Na relaciju generalizacije kod UseCase UML dijagrama se odnose sljedece konstatacije:
    - Ukazuje na opšti use case i ili akter, i njemu određenije (konkretnije) use case-ove (ili aktere),
    - Veza opšteg i specifičnog use case-a koji nasljeđuje opis opšteg use case-a
39. Na relaciju prosirivanja kod UseCase UML dijagrama se odnose sljedece konstatacije:
    - Ukazuje se na alternativni, opcioni ili onaj use case koji se događa pod određenim uslovima
40. Na relaciju realizacije kod UseCase UML dijagrama se odnose sljedece konstatacije:
    - Povezuje specifikaciju i implementaciju te specifikacije,
    - Pokazuje detaljniju realizaciju ili implementaciju apstraktnog use case-a
41. Na relaciju ukljucivanja kod UseCase UML dijagrama se odnose sljedece konstatacije:
    - Ukazuje koja funkcionalnost obavezno prethodi,
    - Ukoliko više use-case-ova koristi istu obaveznu funkcionalnost (npr. login), tada se ona može izdvojiti u poseban use case i include relacija je usmjerena od osnovnog do izdvojenog use case-a
42. Na relaciju zavisnosti kod UseCase UML dijagrama se odnose sljedece konstatacije:
    - Ukazuje na to da use case zahtijeva druge use case-ove za svoje određeniju specifikaciju ili implementaciju
43. Specifikacija i modelovanje korisnickih zahtjeva obavlja se pomocu:
    - Use case dijagrama,
    - Dijagrama aktivnosti
44. Na koju fazu razvoja softvera se odnose Use Case dijagrami?
    - Specifikacija softvera

45. Sta predstavlja Use Case, odnosno, na sta se on odnosi?

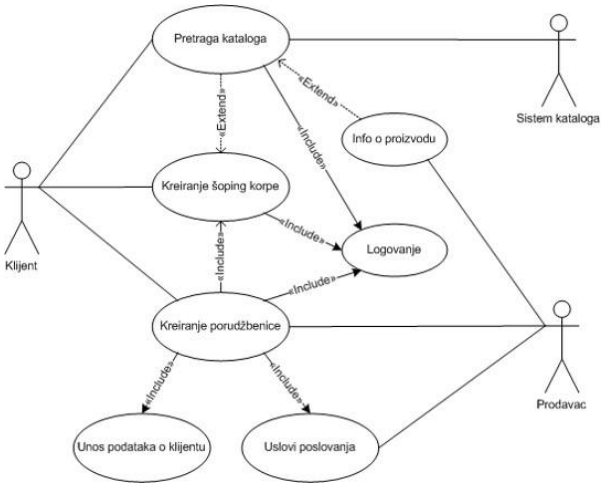
- Funkcionalnost sistema,
- Željeno ponašanje sistema koji se razvija,
- Sekvencu aktivnosti koje sistem izvršava a koje su od posebne važnosti za aktera,
- Šta sistem radi, ali ne i kako sistem radi,
- Dogovor između naručioca softvera i razvojnog tima oko toga šta sistem treba da radi

46. Odaberite uloge use-case UML modela (koje se konstatacije odnose na ovaj UML model).

- Identifikuje korisnike sistema i šta sistem treba da radi,
- Obezbjeđuje komunikaciju sa krajnjim korisnicima i ekspertima

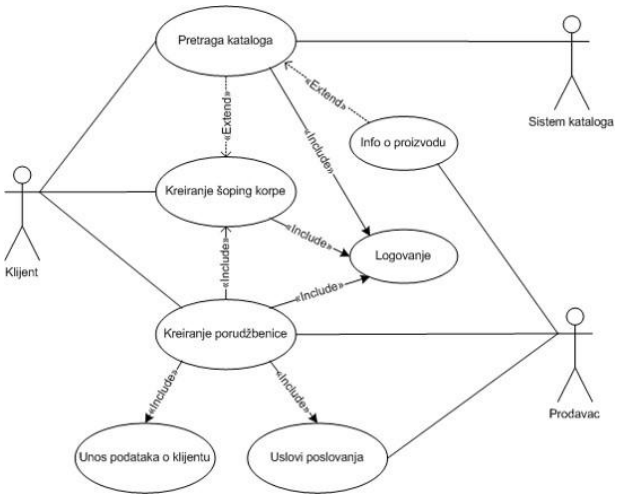
47. Koji UML dijagram je prikazan na donjoj slici?

- Use-Case dijagram



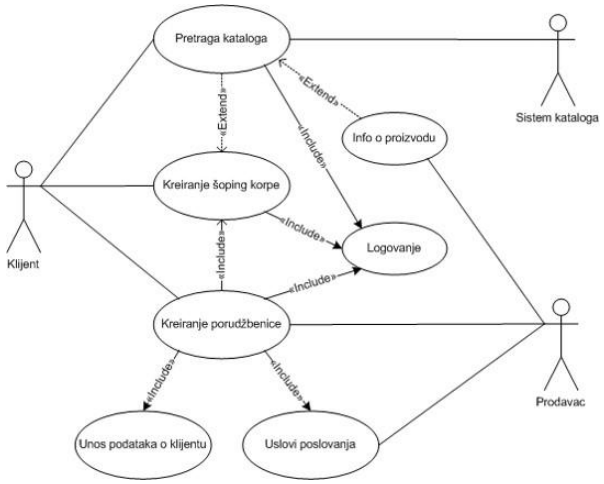
48. Veza izmedju Klijenta i Pretrage kataloga je:

- Relacija asocijacije

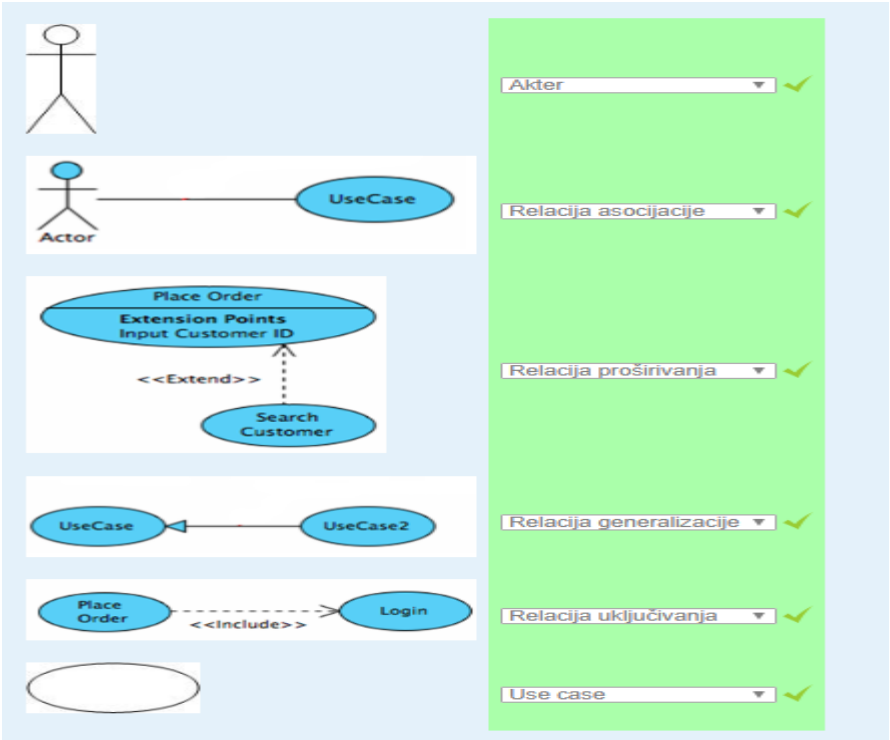


49. Za dijagram prikazan na donjoj slici vazi:

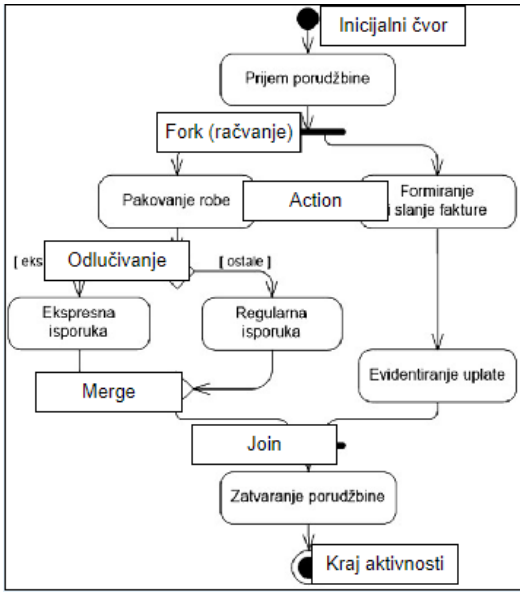
- Use case *Kreiranje šoping korpe* proširuje use case *Pretraga kataloga*
- Use case *Pretraga kataloga* uključuje use case *Logovanje*



50. Povezite elemente UML use case dijagrama sa odgovarajucim objasnjenjem

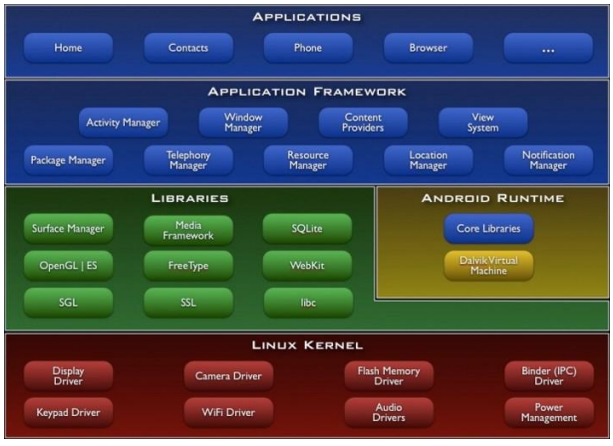


51. Prevlacenjem oznacite djelove dijagrama aktivnosti.



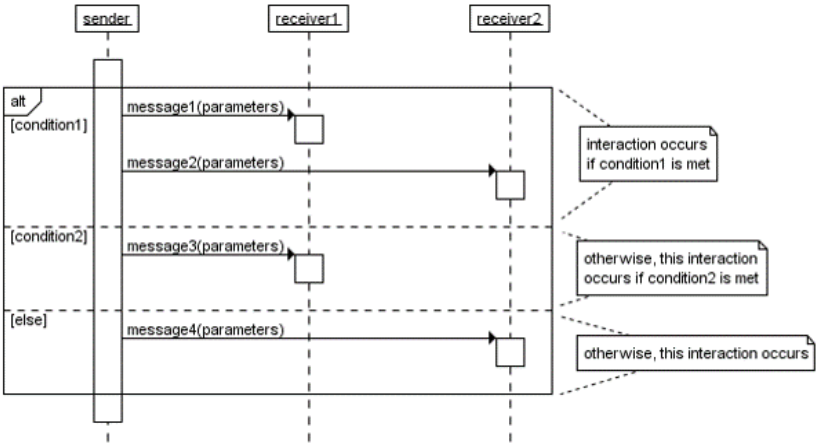
-PETA LEKCIJA-

52. Sta se od navedenog odnosi na analizu sistema?
    - Fokus na razumijevanje problema,
    - Ponašanje,
    - Cilj je razumijevanje problema i početak razvoja vizuelnog modela o tome šta treba da se izgradi,
    - Logički dizajn
53. Na slici je prikazana:
    - Android arhitektura

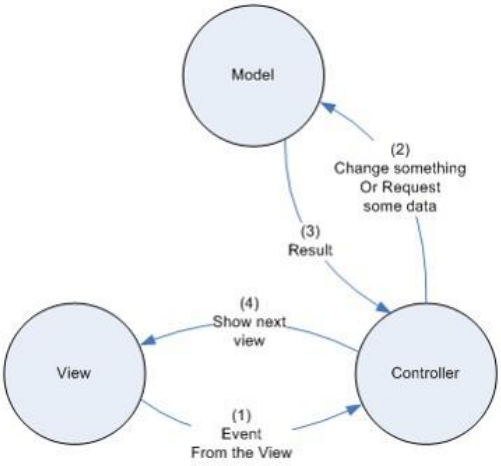


54. Skup strateskih odluka o strukturi softverskog sistema, koji je prikazan kao kolekcija komponenti koje ispunjavaju zeljene funkcionalnosti sistema, optimizujuci pri tome kvalitet, performance, bezbjednost i upravljivost cjelokupnog sistema.
    GORE JE NAVEDENA DEFINICIJA:
    - Arhitekture softvera

55. Na slici je prikazan:
    - Dijagram sekvenci

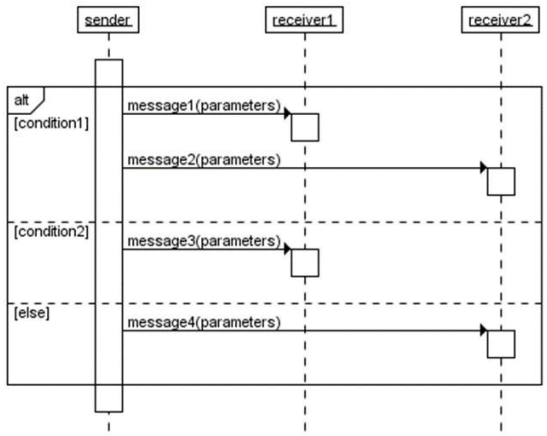


56. Sta se od navedenog odnosi na projektovanje (dizajn) sistema?
    - Fokus na razumijevanje rješenja,
    - Performanse,
    - Cilj je prerađivanje (analitičkog) modela radi razvoja modela dizajna koji će omogućiti prelazak u fazu kodiranja,
    - Fizički dizajn sistema
57. Kod MVC arhitekture, interfejs preko kojeg korisnik komunicira sa sistemom je obezbijedjen preko:
    - View

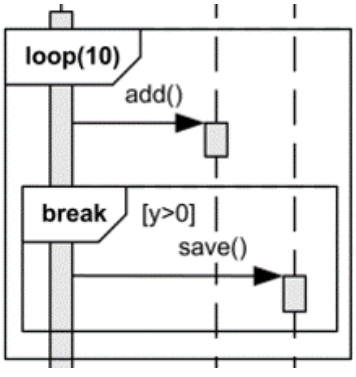


58. Sadrzaj use case-a se modeluje:
    - Dijagramom klasa
59. Interakcije use case-a se modeluju:
    - Dijagramom sekvenci,
    - Dijagramom komunikacije (collaboration)

60. Skup principa koji nude apstraktni okvir sistema koji olaksavaju podjelu i ponovnu upotrebu, pruzajuci pritom rjesenja za ucestale problem poznati su pod nazivom:
    - Sabloni arhitekture
61. Povezite odgovarajuce definicije i termine vezane za sekvencijalne UML dijagrame.
    - Poruke kod kojih posiljalac nije prikazan (nije poznat ili nije relevantan)—PRONADJENE PORUKE
    - Poruke koje se salju samo ako je neki uslov ispunjen---USLOVNE INTERAKCIJE
    - Posiljalac ceka dok primalac ne zavrsi obradu poruke---SINHRONE PORUKE
    - Poruke kojima je potrebno zanemarljivo malo vremena da stignu do primaoca---TRENUTNE PORUKE
    - Posiljalac ne ceka primaoca da zavrsi obradu poruke---ASINHRONE PORUKE
62. Sta vazi za sljedeci kombinovani fragment:
    - Svaka poruka se salje pod posebnim uslovom

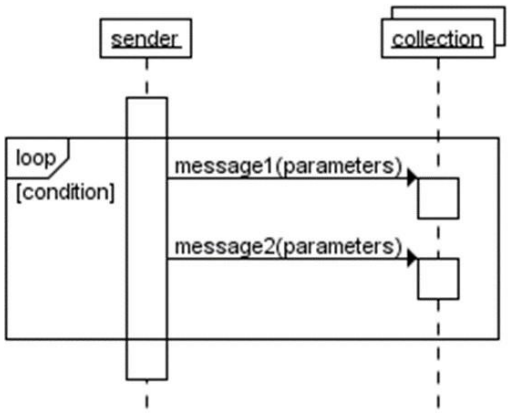


63. Sta vazi za sljedeci kombinovani fragment:
    - Kad je ispunjen uslov y>0 prekida se petlja koja je pozvala unutrašnji fragment



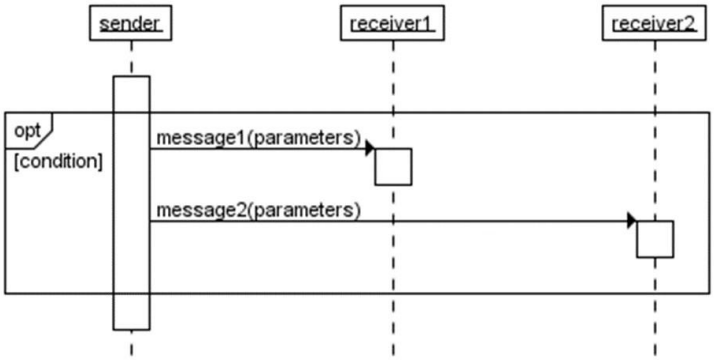
64. Sta vazi za kombinovani fragment:

- Slanje poruka se ponavlja iterativno dok je uslov ispunjen



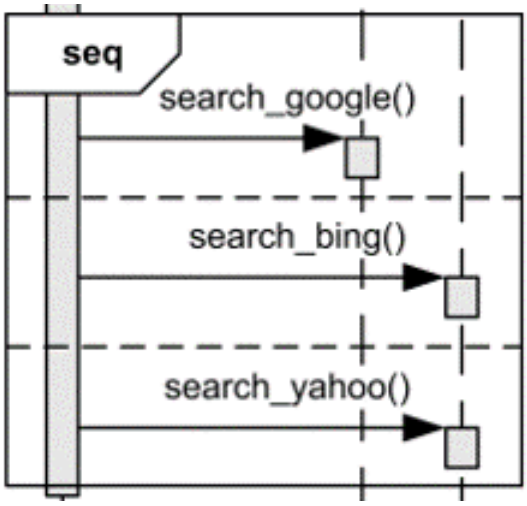
65.Sta vazi za sljedeci kombinovani fragment:

- Poruke se salju pod istim uslovom

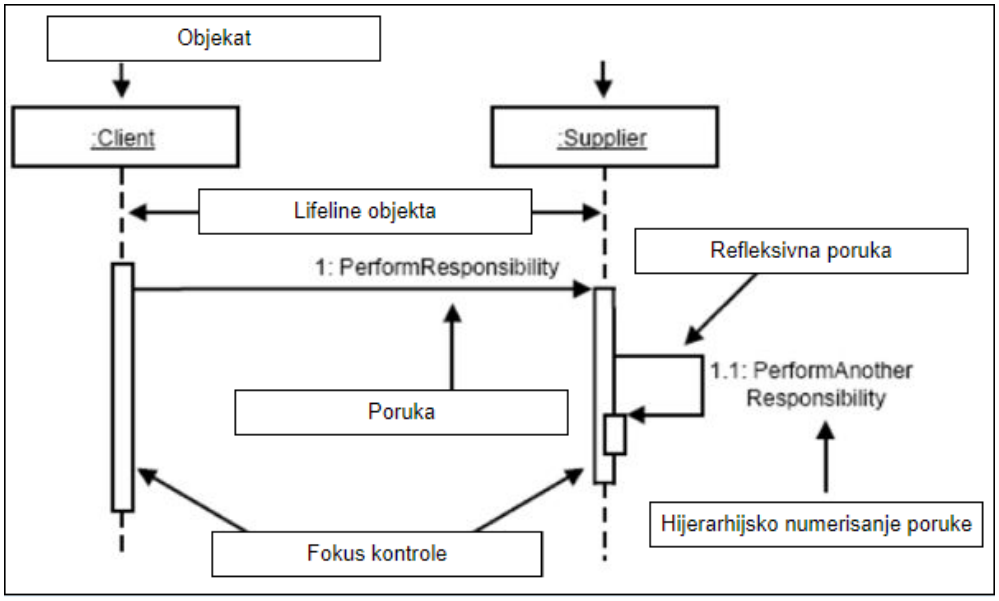


66. Sta vazi za sljedeci kombinovani fragment:

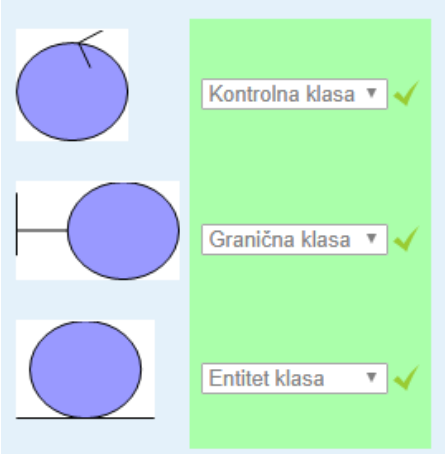
- Redosljed poruka nije obavezan



67. Prevlacenjem oznacite elemente prikazanog dijagrama:



68. Sta oznacavaju navedeni simboli?

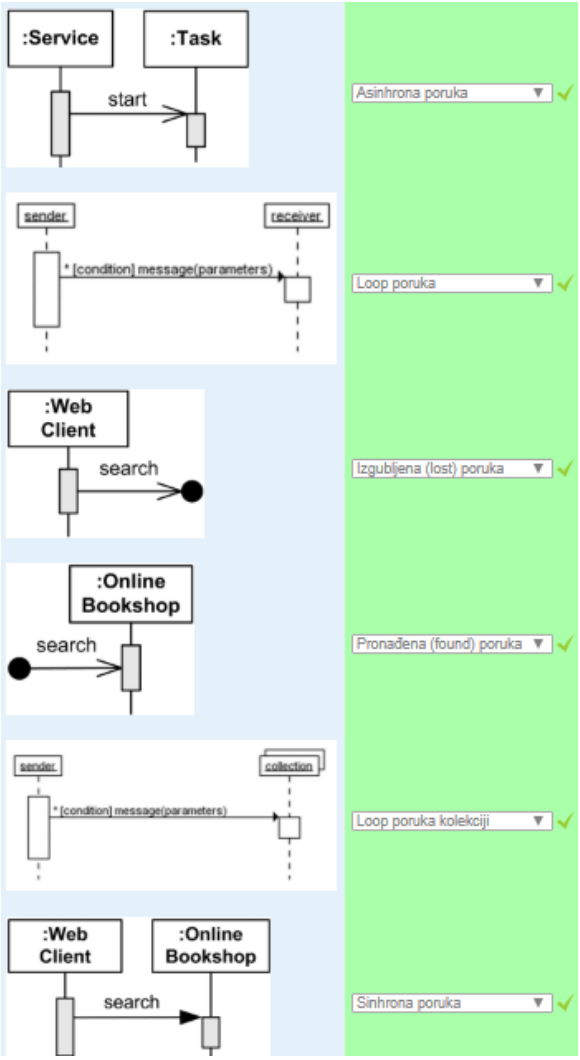


69. Dopunite sljedeci dio koda, tako da je uskladjen sa zadatim dijagramom sekvenci

```
class A {

B b;
   void f(){
       b.g();
       b. k() ✓;
   }
}
class B{
   void g(){
       h() ✓;
   }
   void h(){ }
   void k(){ }
}
```

70. Povezite graficke elemente i odgovarajuca objasnjenja:



71. Popunite donju sliku prevlacenjem teksta u odgovarajuce blokove: