**Blic 1**

1. **Nabroji 3 key-points domain-driven dizajna**

* Model i implementacija oblikuju jedno drugo
* Model predstavlja distilirano znanje
* Model predstavlja temelj jezika koji se koristi

1. **Na koji način konstruktor dojavljuje grešku prilikom njegova izvođenja. Zašto je tako.**

Izbaci exception, na taj način javlja programu da je došlo do greške.

throw new ArgumentException("Reason", "param name");

1. **Navedi koja su 2 osnovna principa OO paradigme**

* Apstrakcija – Prikazuje bit objekta, odnosno sve ono što ga razlikuje od drugih. (Esencijalne karakteristike objekta)
* Enkapsulacija – princip information hidinga. Smanjivanje složenosti odnosno skrivanje nebitnoga sa enkapsulacijom, tim podacima se može pristupiti samo preko sučelja definiranog enkapsulacijom. Pomaže povecati reusability koda

1. **U max 2 rečenice pojasni analysis paralysis.**

Overthinking problema, i izrada kompletnog dizajna prije započinjanja kodiranja. Može dovesti do delaya i poteškoća u začetcima kodiranja. Ima mnogo različitih načina pristupa problemu i traženje najoptimalnijeg ponekad nije optimalno, već koci projekt. Rješenje je iterativan development u konstantan refactoring i feedback od korisnika.

**Blic 2**

1. **Kako se zove pravilo koje govori s kim objekt smije pricati**

Demeterovo pravilo – govori o tome s kim objekt smije pričati (slati poruke)

* Sam sa sobom (this)
* Sa objektima koje je sam stvorio
* Sa objektima koji su mu predani kao parametri
* Sa objektima na koje ima referencu, a koji su njegovi podobjekti

1. **Kako se zove objekt koji ne implementira metode al moze imati clanske varijable**

Apstraktni razred – nedovršen razred, predviđen za daljnje nasljeđivanje i ne može se instancirati

1. **Koji odnos imaju razredi kod nasljedjivanja**

IS-A-KIND-OF odnos

1. **Objasni open-closed principle.**

Softverski entiteti trebaju biti otvoreni za proširenje, ali zatvoreni za modifikaciju. Rješenje pomoću sučelja i apstraktnih razreda

**Blic 3**

1. **Što potpuni(detaljan) use case NE sadržava? (na zaokruživanje)  
   -ponuđeno tipa: glavni scenarij, lista interesa, učesnici, opis user interfacea**

d) Opis user interfacea

Djelovi Use-Casea:

* Stakeholderi i lista interesa
* Glavni scenarij (happy path)
* Preduvjeti i garancija uspjeha (postconditions)
* Ekstenzije (alternativni scenarij)

1. **Napiši svojim riječima liskovino pravilo supstitucije**Ponašanje programa mora ostati nepromijenjeno kada umjesto objekta baznog razreda podmetnemo objekt izvedenog razreda.
2. **Navedi 3 problema kohezije:**

* Mixed domain – Razred sadrži svojstva koja su nedefinirana za neke objekte
* Mixed instance – Razred sadrži element koji direktno opterećuje razred s ekstrinsičnim razredom neke druge domene
* Mixed role – Razred sadrži element koji je direktno opterećen ekstrinsičnim razredom iz iste domene

1. **Izbaci uljeza (na zaokruživanje)  
   -ponuđena 3 ispravna naziva principa + interface defintion principle koji ne postoji**

**Blic 4**

1. **Gradevinski blokovi domain driven designa (6 komada)**

* **Entity** – objekt kojeg definira njegov identitet (sadrži i atribute), mijenjanjem atributa identitet se ne mijenja
* **Value object** – objekt koji sadrži vrijednost, nema identitet
* **Aggregate** – skupina povezanih objekata koje tretiramo kao jedinku prilikom promjene podataka. Definiraju ga korijen (jedan točno određeni entity objekt koji je dio agregata) i granica (definira što se nalazi unutar agregata). Vanjski objekti smiju pristupati samo korijenskim elementima drugih agregata.
* **Domain** **Event** – objekt koji definira event (događaj).
* **Service** – kada operacija konceptualno ne pripada niti jednom objektu implementira se unutar servise
* **Repository** - konceptualni okvir koji enkapsulira rješenje za dohvaćanje objekata. Djeluje kao kolekcija, ali sa poboljšanim mogućnostima postavljanja upita. Dodajemo i brišemo objekte prikladnog tipa, a repozitorij se brine o interakciji s bazom podataka. Na jednom mjestu je definiran kohezivni skup odgovornosti za dohvaćanje objekata.
* **Factory** – programski element odgovoran za kreiranje drugih objekata. Enkapsulira potrebno znanje za kreiranje objekata na jednom mjestu

1. **Sto je tocno za repozitorij (Moze biti vise tocnih odgovora)**

* Brine se o konzistentnosti - agregat
* Kreiranje objekata – tvornica
* **Interakcija s bazom -> Repository (točno)**
* Stvaranje korijenskih elemenata (ne znam, mislim da je samo Interakcija s bazom tocno)

1. **Sto** **je netocno za agregat**

* Elementi unutar agregata ne smiju imati reference na korijene drugih agregata – Netočno

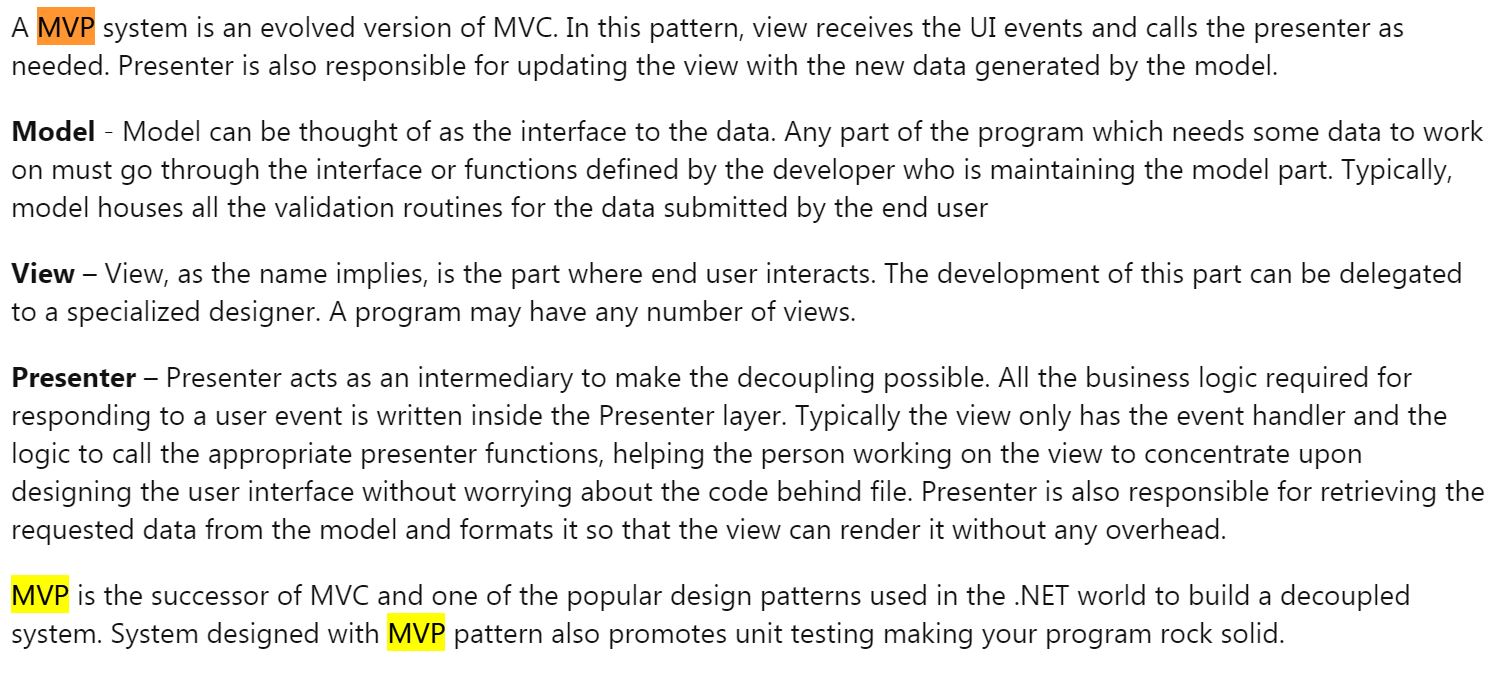
1. **Sto nije tipicna uloga objekta**

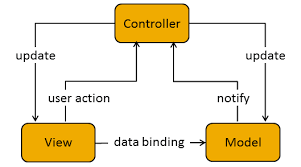
* View

**Blic 5**

1. **Koji sve testovi postoje? (zaokruživanje, više točnih)**
   1. **Unit** **Testing** – testiranje funkcionalnosti malih dijelova izvornog koda
   2. **Integration** **testing** – testiramo kako komponente integriraju, moze biti na razini klase, modula..
   3. **Functional** **testing** – testiranje koda na granici njegovog javnog API-ja. Efektivno, testiramo use case-ove aplikacije
2. **a) Koju modernu verziju MVC-a koristimo?**

* MVP

  
b) bio je crtež sa čovječuljkom, kvadratić View koji ima nacrtani kružić za interface i i kvadratić Controller, i trebalo je nacrtati strelice gdje su potrebne.



…mozda ima nekakve veze

1. **Navedite 3 koraka u TDD?**

* Arrange, Act, Assert

1. **Uloga observera u MVC-u ?**

* Promijena u View-u se mora notify-ati Modelu i Controlleru.