PITANJA ZA USMENI

1. Šta je cilj definisanja klase?

Objekti su skupovi informacija koje se tretiraju kao jedinstven entitet. Da bi se kreirao objekat, potrebno je prethodno imati klasu.

Klase definišu objekat, odnosno one sadrže atribute i metode koje objekat treba da ima. Klase, predstavljaju neku vrstu šablona, prema kome se kreiraju objekti.

Klasa u prirodnom poretku definiše neke zajedničke osobine i ponašanja entiteta koje se ubrajaju pod njenu klasifikaciju, a koji se razlikuju u detaljima vezanim za određene osobine.

2. Šta znate o konstruktoru?

Konstruktori su funkcije koje karakteriše: isto ime kao i ime klase, nemaju povratnu vrednost. Sve ostale karakteristike koje važe za bilo koju funkciju važe i za konstruktore.

Svaki put kada se kreira objekat neke klase zapravo se poziva konstruktor te klase. Veoma često se u konstruktorima vrši inicijalizacija podataka članova klase.

Konstruktori mogu biti bez argumenata (tada se zovu podrazumevani konstruktori ili default konstruktori) ili sa argumentima.

Može postojati samo jedan podrazumevani konstruktor i više konstruktora sa parametrima. Ako ne definišemo podrazumevani konstruktor sam sistem će ga kreirati i telo konstruktora će biti prazno.

3. Kakvo nasleđivanje podržava C#?

Jednostruko nasleđivanje (class A, class B : A)

Hijerarhijsko nasleđivanje (više klasa nadleđuje jednu klasu class A, class B : A, class C : A)

Nasleđivanje od više nivoa (kada klasa naslešuje klasu koja već sama nasleđuje drugu klasu class A, class B : A, class C : B)

Višestruko nasleđivanje interfejsova (c# ne dozvoljava da klasa nasledi više od jedne klase, ali može da nasledi više interfejsova interface A, interface B, class C : A , B)

4. Koje nivoe pristupa možemo definisati atributu klase?

Public, Član je dostupan svima, tj. bilo kojoj klasi, bilo gde u programu.

private, Član je dostupan samo unutar klase u kojoj je definisan.

Ovo je podrazumevani nivo pristupa ako ništa nije eksplicitno navedeno.

protected, Član je dostupan unutar klase u kojoj je definisan i svim izvedenim klasama (nasleđivanje).

internal, Član je dostupan unutar iste skupštine (assembly).

Ovo je korisno kada želite da ograničite pristup na nivo projekta.

protected internal, Član je dostupan unutar iste skupštine i u izvedenim klasama, čak i ako se nalaze u drugoj skupštini.

private protected Član je dostupan unutar klase u kojoj je definisan i u izvedenim klasama, ali samo ako su izvedene klase unutar iste skupštine.

Skupština - assembly

5. Koji elementi klase se ne nasleđuju?

6. Navedite dva razloga zbog kojih bi trebalo da se predefiniše metod koji je već implementiran u nadklasi!

Možda ćete morati da promenite ili dodate funkcionalnost metodu roditeljske klase da bi odgovarao specifičnim potrebama izvedene klase.

Možemo da promenimo fuknciju metode na osnovu toga šta nam treba za izvedenu klasu, ili možemo da dodamo na metodu roditeljske klase

7. Koje metode se ne mogu predefinisati u podklasi?

static, private, konstruktor, metode koje nemaju virtual, abstract ili overrride modifikator

8. Koje metode se moraju predefinisati u podklasi?

Apstraktne metode i metode iz interfejsa

9. Koja je ključna reč za kreiranje klase koja se ne može koristiti u procesu nasleđivanja?

sealed

10. Objasnite pojam polimorfizma!

Pod polimorfizmom u OOP podrazumeva se sposobnost programa da se prilagođava tipu podataka koji se upravo obrađuju.

Sposobnost promenljive da referencira objekte različitih tipova i da automatski poziva odgovarajuću metodu objekta koji se referencira se naziva polimorfizam.

Primer polimorfizma: ako se obrađuje skup geometrijskih figura treba izračunati njihove površine. Površine različitih figura (krug, pravougaonik, trougao, ...) računaju se na različite načine.

Polimorfizam obezbeđuje da u programu ne treba za svaku pojedinačnu figuru ispitivati kojoj vrsti pripada već se prepoznavanje figure vrši automatski.

Polimorfizam se zasniva na sledećem konceptu: metoda koja je deklarisana u baznoj klasi može da se implementira na više različitih načina u različitim izvedenim klasama.

11. Objasnite pojam apstraktne klase!

Ključna reč abstract omogućuje nam da stvorimo klase odnosno članove klase čije je namena da definišu svojstva koja moraju implementirati izvedene klase tj. ne-apstraktne klase.

Apstraktne klase se ne mogu instancirati. Njihova svrha je da pruže zajedničku definiciju bazne klase koju nasleđuje više klasa.

Apstraktne klase mogu definisati i apstraktne metode.

Apstraktne metode nemaju implementaciju. Imaju samo tačku-zarez umesto normalne definicije u vitičastim zagradama.

Izvedene klase apstraktne klase, ako i same nisu apstraktne, moraju implementirati sve apstraktne metode.

12. Koje prednosti ima korišćenje interface-a?

Podržava višestruko nasleđivanje (jedna klasa može da nasledi više interfejsova)

Apsstrakcija omogućava programeru da sakrije sve osim relevantnih podataka o objektu kako bi smanjio složenost i povećao efikasnost. Mnogi jezici, poput C#, postižu apsstrakciju koristeći interfejse.

Interfejs će samo specificirati šta klasa može da radi, a ne kako će proces biti implementiran.  
Interfejs se koristi za postizanje fleksibilne povezanosti i potpune apsstrakcije.

13. Šta je izuzetak?

Prilikom izvršavanja C# koda mogu se javiti različite greške: greške u kodiranju koje je napravio programer, greške zbog pogrešnog unosa ili druge nepredvidljive stvari.

Kada dođe do greške, C# obično stane i generiše poruku o grešci. Tehnički termin za ovo je: C# će izazvati izuzetak (throw an exception).

14. Objasnite prednost korišćenja stilova u XAML fajlu!

Ako imamo više nekih elemenata (na primer button) koji moraju da imaju iste stilove (na primer boju teksa boju pozadine i marginu) možemo da definišemo style za sve button-e i tako smanjujemo količinu pisanja koda i istovremeno šansu za greške

15. Šta znate o Grid kontejneru?

Za grid treba da definišemo koliko želimo kolona i redova

<Grid.ColumnDefinitions>

<ColumnDefinition></ColumnDefinition>

<ColumnDefinition></ColumnDefinition>

<ColumnDefinition></ColumnDefinition>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition></RowDefinition>

<RowDefinition></RowDefinition>

<RowDefinition></RowDefinition>

</Grid.RowDefinitions>

Širinu kolona, ondosno visinu redova možemo da odredimo uz pomoć width:\*

Na primer ako imamo tri kolone i svaka ima width:\* one će biti iste širine, ali ako jedna ima 2\* ona će biti duplo šira od druge dve (druge dve ostaju međusobno iste). Kada se saberu zauzimaće ceo prozor (ne računajući marginu)

Elemente u gridu možemo da rasporedimo po kolonama i redovima (npr. Grid.Row="0" Grid.Column="0")

16. Šta znate o StackPanel kontejneru?

Stackpanel „produžava” svoje elemente automatski na maksimalnu širinu, tako da možemo da slažemo elemente jedne na druge.

Podrazumevana orijentacije je od gore ka dole (orientation=”vertical”) ali možemo da ga prebacimo da ide od leva ka desno (orientation=”horizontal”)

Za elemente stackpanela možemo da koristimo verticalalignment i horizontalalignment (left right center top bottom)