PITANJA ZA USMENI

1. Šta je cilj definisanja klase?

2. Šta znate o konstruktoru?

* Da bi se omogućila inicijalizacija objekta, u klasi se definiše posebna metoda koja se implicitno (automatski) poziva kada objekat nastaje – Konstruktor – klasa može da ima više konstruktora.
* 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#?

* Jedna klasa može biti izvedena iz neke druge klase i kao rezultat nasleđivanja osobine osnovne tj. roditeljske klase se prenose i na izvedenu klasu (sreću se i termini klasa potomak, sin itd).
* Izvedena klasa ne nasleđuje konstruktor osnovne klase.
* Ako je definisan konstruktor u osnovnoj klasi potrebno je definisati i konstruktor u izvedenoj klasi.
* Da bi pozvali konstruktor osnovne klase u konstruktoru izvedene klase mora stajati eksplicitni poziv.

4. Koje nivoe pristupa možemo definisati atributu klase?

* Public – ovi članovi nazivaju se javnim, dostupni su spolja.
* Private – članovi klase su nedostupni izvan klase, nazivaju se privatnim.

5. Koji elementi klase se ne nasleđuju?

* Konstruktori, destruktori i metoda operator= osnovne klase se ne nasledjuju.

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

7. Koje metode se ne mogu predefinisati u podklasi?

8. Koje metode se moraju predefinisati u podklasi?

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

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.

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.

12. Koje prednosti ima korišćenje interface-a?

* Interfejsi su apstraktne klase (čiste apstraktne klase) koje sadrže samo potpuno virtuelne metode. Njegova jedina svrha je da deklariše skup metoda a ne da ih realizuje.
* Ne sadrže polja! Elementi interfejsa ne mogu biti static.
* Elementi interfejsa su implicitno public abstract (virtual).

13. Šta je izuzetak?

* Izuzetak je greška koja se otkriva tek pri izvršavanju programa.
* Try, Catch, Finally.

14. Objasnite prednost korišćenja stilova u XAML fajlu!

* Stilovi u XAML se koriste zbog GUI (korisnički interfejs), kako bi izgledao što bolje i primamljivije.
* Stilizovati se može sve: StackPanel, Grid, ComboBox, TextBox, TextBlock, RadioButten, CheckBox…
* Može se postaviti veličina i tip fonta, boja, pozadinska boja, širina, visina, margine, padding…

15. Šta znate o Grid kontejneru?

* Grid može da sadrži više redova i kolona, mi definišemo visinu reda i širinu kolone (apsolutna količina piksela ili u procentu dostupnog prostora ili automatski), redovi i kolone automatski prilagode svoju veličinu u zavisnosti od sadržaja.
* Grid će jednostavno uzeti sve kontrole koje stavite u nju, razvući ih da iskoriste maksimalni raspoloživi prostor i postaviti je jednu na drugu.

16. Šta znate o StackPanel kontejneru?

* StackPanel ne omota sadržaj (no wrap), već ga proteže (stratch) u nekom smeru
* <StackPanel Orientation="Horizontal"> </StackPanel> - jedan pored drugog.
* <StackPanel Orientation="Vertikal"> </StackPanel> - jedan ispod drugog.