1. Šta predstavlja .NET Frimework?

.NET Framework predstavlja obiman skup klasa koje se koriste u programiranju različitih tipova aplikacija.

2. Kako su grupisane klase?

Sve klase su grupisane u takozvane imenovane prostore (named spaces) radi lakšeg snalaženja i efikasnijeg rada.

3. Šta raditi ako klasa ne postoji definisana u biblioteci .NET Frimework?

Ako ne postoji, može se napisati od početka ili naslediti funkcionalnost postojećih klasa.

4. Ko je Anders Hejzberg?

Nepravedno je govoriti o C#, a ne pomenuti Andersa Hejlsberga, koji je glavni arhitekta i projektant ovog jezika i učesnik u razvoju .NET tehnologije. Rođen u Danskoj 1961. godine, programer koji je početkom osamdesetih godina 20. veka napisao čuveni Turbo Pascal, a kasnije projektovao Delphi.

5. Zašto se C# izdvaja od ostalih programskih jezika iz Visual Studio paketa programa?

C# se izdvaja od drugih i smatra se prirodnim za .NET okruženje. Ovaj programski jezik je nastao kao potpuno nov i nije opterećen kompatibilnošću sa ranijim verzijama. Radno okruženje Visual Studio je u potpunosti napisano baš u C# jeziku. On je moderno strukturiran, potpuno objektno orijentisan, zasnovan na C++ jeziku i zvanično prihvaćen kao standard od strane organizacija ECMA (European Computer Manufacturers Association) i ISO (International Organization for Standardization).

6. Koje mogućnosti prevođenja programskog koda napisanog u C#?

Postoje dve mogućnosti: prevodioci i tumači.

7. Na koji način prevodioci vrše prevoženje programskog koda?

Prevodioci prevode izvorni kôd direktno u mašinski kôd procesora.

8. Na koji način tumači vrše prevoženje programskog koda?

U slučaju tumača se prilikom pokretanja programa prevodi linija po linija - kako se program izvršava.

9. Da li je .Net tumač ili prevodilac i na koji način se ovdje vrši prevođenje programskog koda?

. **NET je tumač**, ali ne vrši prevođenje direktno u mašinski jezik procesora, već u takozvani MSIL (Microsoft Intermediate Language). Po pokretanju programa na scenu stupa JIT (Just In Time) prevodilac, koji dalje prevodi MSIL u mašinski jezik procesora i izvršava ga. Ovakva slojevita arhitektura omogućava portabilnost .NET aplikacija između različitih operativnih sistema u budućnosti.

10. Koja su tri osnovna tipa projekta u C# i objasnite svaku od navedenih?

Tri osnovna tipa projekta su: Windows aplikacija (Windows application) predstavlja standardnu Windows aplikaciju koja se može samostalno pokretati na računaru. **Biblioteka klasa** (Class Library) je biblioteka klasa sa svojim funkcijama i drugim elementima. Ovaj projekat ne može se samostalno pokrenuti, već ga pokreću i koriste drugi tipovi projekata. **Aplikacija konzole** (Console Application) takođe Windows aplikacija, ali bez grafičkog korisničkog interfejsa. Komunikacija se odvija isključivo sa komandne linije. Ostali tipovi predstavljaju samo različite predloške i demo aplikacije uz mogućnost pronalaženja drugih predložaka projekata na Internetu.

11. Šta predstavlja linija **static void Main(string[] args)** u programskom kodu C# jezika i kako de definisano tijelo ove funkcije?

Ona definiše glavnu (Main) funkciju koja se prva pokreće prilikom pokretanja programa. U telu ove funkcije treba upisati kôd koji je potrebno izvršiti. Telo funkcije je definisano početnom { i završnom } velikom zagradom, što predstavlja način na koji se obeležava početak i kraj svih segmenata u C# jeziku.

12. Da li C# razlikuje velika i mala slova?

    Da

13. Pomoću koje alatke se vrši pokretanje aplikacije iz radnog okruženja?

Pomoću funkcijskog tastera F5.

14. Šta radi linija koda : Console.WriteLine("Moja prva C# aplikacija");?

Ova linija koda odgovorna je za ispis teksta. Console objekt predstavlja prozor komandne linije, dok je WriteLine metod ovog objekta kojim se ispisuje tekst.

15. Šta radi linija koda: Console.ReadKey();?

Linija: Console.ReadKey(); postoji samo da bi se napravila pauza. Bez nje bi se otvorio prozor komandne linije, ispisao tekst i prozor bi se odmah zatvorio jer se aplikacija završila. Slično kao i u prethodnoj liniji koristi se objekat Console, a ovog puta metod ReadKey koji čeka da korisnik pritisne taster na tastaturi.

16. Koja je prednost objektno orijentisanog programiranja?

Za razliku od tradicionalnog proceduralnog programiranja koje se sastoji od niza funkcija i njihovih poziva, objektno programiranje je bolje i skladnije oslikava realni svet i omogućava logičnije projektovanje programskih zahteva. **Ono što nam donosi objektno programiranje je skladniji programski model i još neke lepe stvari kao što je ponovna upotrebljivost i nasleđivanje objekata. Jedan objekat je uvek predstavljen jednom klasom.**

17. Šta je klasa?

Klasu treba posmatrati kao nacrt ili plan za izgradnju objekta. Na primer, plan za izgradnju kuće predstavlja klasu, a kuća napravljena po tom planu predstavlja objekat zasnovan na klasi. Sama klasa za sebe nije objekat, već predstavlja samo opis po kome će objekat biti napravljen.

18. Šta je objekat?

Objekt, što je konkretna realizacija klase.

19. Šta znači instancirati?

Znači napraviti objekat na osnovu klase. Kada napravimo objekat na osnovu klase, on se kreira u memoriji računara i tek tada se može koristiti. Na osnovu jedne klase može se napraviti neograničeni broj objekata, odnosno broj koji je tehnički ograničen memorijom računara.

20. Šta je metoda klase?

Učenik obavlja veliki broj školskih aktivnosti koje takođe treba opisati. To je ono što učenik radi u školi i kod kuće, na primer, radi pismeni i domaći zadatak, prisustvuje određenom času, dobija ocenu i slično. Ove radnje koje učenik izvršava i koje se odnose na njega nazivaju se metodima (eng. methods) klase.

21. Šta su svojstva klase?

Uzmimo učenika kao primer. Ako bismo hteli da ga predstavimo u objektnom svetu, šta bi klasa koja ga opisuje trebalo da sadrži? Možemo početi od fizičkih karakteristika. Ime učenika, pol, godina rođenja, boja očiju i kose, razred i slično. Sve pomenuto predstavlja svojstva (eng. properties) klase.

22. Šta su događaji klase?

Učenik takođe na određeni način reaguje na dešavanja u školskom okruženju. Reakcija na dobru ili lošu ocenu, izostanak u slučaju bolesti i slično predstavljaju događaje (eng. events) klase.

23. Fizička implementacija klase.

Svojstva klase su predstavljena različitim varijablama koje su javno vidljive van klase. Korisnik klase će po kreiranju objekta na osnovu klase uspostaviti odgovarajuće vrednosti varijabli i time definisati objekat. Metode klase su predstavljene funkcijama u klasi. Na primer, funkcija „UradiDomaćiZadatak“ koja bi kao ulazne parametre verovatno trebalo da sadrži opis domaćeg zadatka, predmet, vreme kada je zadat i rok izvršavanja. Metod klase može imati i povratnu vrednost koja se obično koristi kao status izvršavanja metoda, odnosno funkcije. Nepisano pravilo jeste da povratna vrednost nula znači uspešno izvršenu radnju, a vrednost različita od nule označava problem definisan tim brojem.

24. Napišite sintaksu za deklaraciju varijable?

vidljivost tip varijable naziv = početna vrednost Na primjer: public string ImeUcenika;

25. Kakve su to privatne, a kakve javne varijable?

Vidljivost definiše odakle se varijabli može pristupiti. Ključna reč „public“ znači da je varijabla javna u celom opsegu u kome je deklarisana – na primer, u celoj klasi i da je takođe vidljiva korisnicima te klase. U drugom slučaju varijabla sa privatnom vidljivošću – dostupna je samo u okviru dela u kome je deklarisana, ali ne i korisnicima klase.

26. Kako se definiše funkcija u C#?

Definiše vidljivost, ime funkcije, tip (koji u slučaju funkcije znači tip vrednosti koju vraća funkcija) uz dodatak deklaracije ulaznih argumenata funkcije ako postoje.

Primer funkcije koja vraća zbir dva broja:

```
public double Zbir (double a, double b) { return a + b; }
```

27. Kako se deklariše funkcija koja nema povrtanu vrijednost?

Funkcije koje nemaju povratnu vrednost deklarišu se kao tip void i u tom slučaju se navodi samo ključna reč return bez parametra.

28. Kako se pišu komentari u programskom jeziku C#?

Dve kose crte // označavaju komentar sve do kraja reda. Osim ove jednolinijske varijante komentara postoji i višelinijski komentar koji počinje parom znakova /* i završava sa */. Sve linije između smatraju se komentarom i ne izvršavaju se.

29. Napišite naredbu za deklaraciju klase.

```
class NazivKlase
        { /* telo klase sa svim potrebnim svojstvima, metodima i događajima */ }
```

30. Napiši naredbu kojom se pravi instanca klase.

NazivKlase NazivVarijable = new NazivKlase();

31. Šta je InteliSense?

Tehnologija koja pruža pomoć prilikom kucanja koda na taj način da nam ponudi nazive naredbi koje se modu koristiti. Naredba se može unijeti i izborom iz spiska ponuđenih naredbi.

32. Kako u C# da zabranimo pogrešne vrednosti i primjenimo zaštićena svojstva?

Da bi ovo ostvarili, moramo posedovati neki mehanizam koji proverava vrednost pre nego što se dodeli varijabli. Ovo se postiže pomoću specijalnih **set** i **get** funkcija. Tehnika je sledeća: umesto da se pravi javna (public) varijabla, treba napraviti privatnu (private) varijablu i radi bolje čitljivosti joj dati slično ime kao što ima i javna varijabla – na primer prRazred (pr od private). Sintaksa je jednostavna, navodi se tip i naziv svojstva koje je sada javno (public) i u okviru njega upišu set i get funkcije.

33. Šta je potrebno napisati u get i set dio?

U get delu, se korisniku zapravo vraća vrednost privatne varijable. U set delu se prvo provjerava da li je unieta vrijednost u zadatim granicama i ako jeste onda se dodeljuje vrednost privatnoj varijabli. Primjer: prRazred = value;

34. Šta se postiže dodjelom vrijednosti **value**?

Promenjiva value je specijalna C# varijabla i predstavlja vrednost koju je korisnik upotrebio prilikom dodele vrednosti svojstvu. I ovde se nalazi ključ za proveru vrednosti pre dodele privatnoj varijabli.

35. Koju proceduru je potrebno ispoštovati ako želimo da napravimo zaštićena svojstva?

- napraviti privatnu varijablu koja ima isto ime kao željeno svojstvo sa nekim prefiksom. (Ovu konvenciju imenovanja treba shvatiti kao preporuku, a ne kao čvrsto pravilo)
- napraviti javno (public) svojstvo sa željenim imenom i tipom istim kao privatna varijabla
- unutar javnog svojstva uneti get i set delove.
- u get delu korisniku pomoću return naredbe vratiti vrednost privatne varijable
- u set delu proveriti vrednost koju korisnik dodeljuje svojstvu (value) i ako je vrednost validna dodeliti je privatnoj varijabli. U suprotnom prikazati poruku o grešci.

36. Koji su logički operatori koji vezuju dva ili više uslova?

To su && (i), || (ili)

37. Kojom klasom su predstavljane greškau .NET okruženju?

Sa klasom Exception .

38. Napiši sintaksu za kreiranje objekta na osnovu klase Exception.

Exception greska = new Exception ("Pogrešna vrednost svojstva Razred");

39. Kojom naredbom se vrši pokretanje ili podizanje greške (eng. raise error)?

Izvršava se naredbom **throw.**

40. Šta se dešava kada se izvrši throw?

Kada se izvrši throw, klasa prestaje sa radom i ovu grešku prenosi na korisnika klase, koji potom treba da je obradi na odgovarajući način.

41. Kako se dijele greške koje nastaju prilikom pisanja koda u C#?

Dijele se na sintaksne i na greške koje se dešavaju za vrijeme izvršavanja programa.

42. Kakve su to sintaksne greške?

Sintaksne greške (eng. syntax errors) – rezultat pogrešno napisane naredbe, nedeklarisane varijable i slično. Ove greške najčešće nisu problematične jer program koji ih sadrži ni ne može da se kompajlira.

43. Kakve su to greške koje se javljaju za vrijeme izvršavanja programa?

Greške u vreme izvršavanja (eng. runtime errors) – kao u našem primeru, greške koje se mogu ali i ne moraju dogoditi za vreme izvršavanja programa. Najčešće su rezultat neproverenih vrednosti varijabli i pogrešnog unosa korisnika. Relativno se lako pronalaze i ispravljaju.

44. Kako obraditi grešku promoću naredbi try i catch?

Svaku liniju ili više njih koje mogu generisati grešku treba smestiti u try blok. U slučaju da se generiše greška, tok programa se automatski preusmerava na catch blok. U njemu pišemo kod koji ispravlja grešku ili bar obaveštava korisnika šta nije u redu. Opciono pomoću Exception varijable možemo proveriti koja greška se dogodila, videti njen opis i zavisno od toga izvršiti odgovarajuću akciju.

45. Koja je sintaksa naredbi try i catch?

U osnovnom obliku sintaksa je sledeća: try { // linije koda koje mogu da generišu grešku } catch (Exception varijabla) { // obrada greške }.

46. Od kojih elemenata se sastoji Windows aplikacija?

To su komandna dugmad, liste, padajuće liste, izbor opcija, meniji i tako dalje.

47. Koja su dva osnovna dela kada radimo Windows aplikacije?

To su **dizajn korisničkog interfejsa** u kome "crtamo" i raspoređujemo kontrole na formi. Ovde takođe vršimo i podešavanje svojstava i ponašanje kontrola. **Dio u kome se vrši kodiranje**, to je dio u kome pišemo kod kao odgovor na akcije korisnika, na primer klik mišem na komandno dugme, izbor iz liste, zatvaranje forme i slično.