1. **string haqida gapirib bering.**

Sring o’zgarmas (immutable), o’zida matn saqaliydi, reference type, string bilan ishlash xotiradan ko’p foydalanishga olib keladi yani xotiradan ko’p joy egallaydi.

Agar stringniga o’zgartirish kiritilsan oxtiradan yangi mazil ochilib saqlandi va bu xotiradan har stringda o’zgartishi kiritildan ko’p joyni olishiga sabab bo’ladi

1. **stringBuilder haqida gapirib bering.**

StringBuilder o’zgaruvchan (mutable), bu string bilan ishlashfa samarali bo’lib bunda o’zgartirshlar qilinganda yangi ochmasda o’zini o’zgartirib ketadi bu stringda ishlashda osonlashtiradi va xotirani tajaydi tezroq ishlaydi, bu aslida class bo’lib bundan obyekt olish ishlanadi

1. **array vs list**

Fiksatsiyalangan hajmli va bir xil turdagi elementlarni saqlash uchun ishlatiladi. Listning hajmi dinamik bo’ladi unda qo’shish o’zgartish ammalani bejarish mumkin, List xotiradan ortiqcha yuklanishlarni olib keladi, Array tezroq ishliydi va ketma ket joylashgan xotira talab qiladi. Array uzunligi anniq bo’lishi shart

1. **ref vs out**

ref va out parametrlar qiymatlarini o‘zgarishiga imkon beruvchi usullardir. ref: Parametrga qiymat berilishi shart, u metod ichida o‘zgarishi mumkin. out: Parametrni metod ichida qiymat berish orqali ishga tushirish talab qilinadi. ref qiymatni o‘zgartirish uchun ishlatiladi, out esa metoddan bir nechta qiymatni qaytarishda foydali.

1. **static**

Static classga tegishli bo’lib static class Ichida faqat static methodlar bo’ladi, static classda obyekt olib bo’lmaydi, class nomi bilan ishlatiladi, nonstatic Ichida hammasi bo’lishi mumkin, nonstatic ichidagi static methodi class nomi bilan ishlatamiz

1. **class**

Class bu obyeklat yaratishi uchun qolibdir uni Ichida method prop fields eventlar yoziladi, meros olib bo’ladi, interfacelarni qo’llaydi, reference type

1. **object**

Hayotdagi hamma narsa obyektdir class qoliblarda obyektlar olinadi, hayotdagi hamma malumotlarni o’z ichiga oladi, barcha turdagi malumot turlarini o’z ichiga saqlaydi, obyektlar sinfning nusxasi.

**8. abstraction**

Mavhumlic, abstract classlarning asosiy maqsadi koddagi murakkablikni yashirish, faqat kerakli malumotlar ko’rsatiladi murakkablik yashiriladi, faqat zarur xususiyatlar ko’rsatish jarayonidir. Abstrac classlar ichida 3 xil methodlar yozish mumkin 100% abstractk, virtulat (xohlasa override qilinadigan) va konkret yashirilmagan method.

**9. inheritence**

Meros olish, bu kodni qayta ishlatishni oldini olish va kodni soddalashtirsh tushunishga osonlashrish uchun ishladiladi, ota class bola classga inheritance qilinadi bola class ota classdan meros oladi, shunda ota classda hamma narsalar bola classda ishliydi, class bola : ota (meros olishi), faqat bitta ota meros bo’lishi mumkin faqat interfacelardan ko’plab meros bo’lishi mumkin

**10. encapsulation**

Malumotni kapsulalash yashirish izalatsiyalash himoya qilish, Malum bir tartib ruxsat bergan usul bilan aloq qilishi mumkin, bu asosan public protected privete va proplar orqali amalga oshiriladi

**11. OOP**

Obyektga yo’naltirlgan dasturlash, dastut yozishni ikki yo’lidan bir bo’lib dastur oop fundamentida ko’tariladi, obyeklar orqali dasturlash usulidir, dasturlarni modulli qilib boshqarishni osonlashtiradi, Tamoyillari – Encapsulation inheritance abstaraction polymorphism

**12. Interface**

100% abstarac shartnoma (contract ), kodlarni murakkabligini yashirih uchun ishlatiladi bir necha interfacelar bola classga inhert bo’lishi mumkin, interface ichidagi hamma methodlar public methodlar va bola classda implatsiya qilinishi shart

**13. value type**

Qiymat turi, stackning qisqa muddatli xotirada saqlandi va shuning uchun tez, nusxa ko’chirilganda nusxa mustaqil qiymatga ega bo’ladi. oddiy, stuct, enum. Null bo’lmaydi, nusxa ko’chiradi. Ini, float, bool, struct

**14. ref type**

Heapda saqlanadi uzoq muddatli xotirada), havola orqali qiymat oladi, null bo’la oladi, class string array obj. qiymatning o’zi emas havolasi

**15. Casting types**

**16. Repository & service pattern**

**17. Dto**

**18. Constructor**

**19. object**

**20. Interface vs Abstract class**

**21. Abstract class**

**22. Has a, is a**

**23. Generic**

**24. Struct**

**25. Partial class**

**26. Exception**

**27. try, catch, finally**

**28. Extension methods**