**1. Primitív adattípusok, hosszuk byte-ban, ábrázolható értéktartományuk, melyiket mikor használjuk.**

byte – 1 bájt Byte.MIN\_VALUE, Byte.MAX\_VAUE  
short – 2 bájt Short.MIN\_VALUE…  
int – 4 bájt…  
long – 8 bájt  
bool – 1 bájt  
char – 2 bájt  
float – 4 bájt  
double – 8 bájt

**2. Java tömbök**

int[] numbers = {1, 2, 3, 4, 5};

int[] numbers;  
numbers = new int[] {1, 2, 3, 4, 5};

int[] numbers = new int[5]; // 5 elemű tömb deklaráció  
numbers[0] = 1;  
numbers[1] = 2;  
numbers[2] = 3;  
numbers[3] = 4;  
numbers[4] = 5;

**3. A referencia változó fogalma, szerepe, használata**

Amikor objektumot adunk meg például paraméterként egy függvénynek, akkor nem készül róla másolat, mint a primitív adattípusokról, hanem egy nem változtatható pointer adódik át, ami az objektum memóriacímére mutat. Ez különösen akkor hasznos, ha nagy, összetett objektumok másolását spóroljuk így meg. A referenciát nem látjuk a szintaxisban, a háttérben működik.

**4. Objektum, osztály (class) és a referencia kapcsolata**

A java minden objektumot implicit referenciaként ad át. Az osztály példányosításakor pedig objektumok jönnek létre. A referenciáknak is van típusa, olyan típusúak, amilyen osztályra mutatnak. Vagyis két nem azonos osztályra mutató referenciát nem lehet egyenlővé tenni.

**5. A Java osztály deklarálása: szintaktikai szabályok**

class MyClass{} – az osztály neve mindig nagy kezdőbetű, az adattagok kerülnek előre, aztán a konstruktor(ok), végül a függvények

**6. Java program fordítása és futtatása**

javac MyClass.java paranccsal lefordítjuk a java compilerrel a java fájlunkat, így készítünk egy class fájlt, majd java MyClass paranccsal futtatjuk a class fájlt a virtuális gépen.

**7. A Java virtuális gép szerepe a Java programok futtatásában és a platform függetlenség biztosításában, főbb funkciói**

A virtuális gép egy elméleti processzor, ami lefuttatja a java által generált java bájtkódot. Ezáltal a virtuális gép teszi lehetővé a java platformfüggetlenségét.

**8. A Java konstruktor deklarálási szabályai, default konstruktor jelentése és szerepe, objektumok inicializálása**

MyClass(int parameter1, double parameter2){  
this.parameter1 = parameter1;  
this.parameter2 = parameter2;  
this.parameter3 = 0;  
}

a default konstruktort nem szükséges megírni, a compiler automatikusan készít egy paraméter nélküli konstruktort, ha nincs már konstruktor az osztályban megírva.

MyClass object = new MyClass();

**9. Memória kezelés megvalósítása Java-ban, ezzel kapcsolatos programozói feladatok**

A memória lefoglalása a new kulcsszóval történik, a memóriakezelést a virtuális gép garbage collectora végzi (detektálja, majd felszabadítja a használaton kívüli memóriát), a programozónak nincs több feladata

**10. Metódusok helye a Java osztályban, deklarálásuk**

…

**11. Változók és metódusok láthatósági kategóriái, melyiket mikor használjuk?**

private – csak a saját osztály tagjai láthatják

protected – csak a saját osztály és a leszármazott osztály tagjai láthatják

public – minden osztály láthatja

package private – csak a package osztályai láthatják -ez az alapértelmezett láthatóság javában

**12. Vezérlési szerkezetek: ciklusok, feltételes végrehajtási szerkezetek**

ciklusok: while, do-while, for, for[each]  
vezérlési szerkezetek: if-else, switch

**13. Végrehajtási stack jelentése, szerepe a program végrehajtása és annak ellenőrzése során**

A végrehajtási stack az a verem típusú tároló, ami a függvények végrehajtási sorrendjét tárolja. Ha a program exception-t dob, a végrehajtási stack megjeleníthető és a hiba forrása ezáltal detektálható.

**14. Az Exception kezelés alapelvei, a try-catch blokk használata, az Error és Exception közötti különbség**

Az exception kezelés alapelve, hogy azt a kódrészletet, aminek a futása során valamilyen hiba léphet fel, és exceptiont dobhat, try blockba tesszük, majd ezt a hibát a catch blockban elkapjuk, és lekezeljük (kiírjuk a hiba okát, vagy megjelenítjük a végrehajtási stacket. A catch blockot sosem hagyjuk üresen.

Exception és error között az a különbség, hogy Exception esetén még tovább futhat a program, az error már egy súlyosabb hiba, ilyenkor nem fut tovább a program.

**15. A "static" kulcsszó jelentése, szerepe a programokban**

A static kulcsszó statikussá tesz egy adattagot vagy függvényt, ami azt jelenti, hogy annak tartalmát vagy működését nem befolyásolja az osztályból létrehozott objektumok különbözősége. Ezeket nem az objektumokból, hanem az osztályokból tudjuk létrehozni.

**16. Osztályok közötti relációk (kapcsolatok), ezek ismertetése, jellemzése, példák a használatukra**

aggregáció – Az A osztály ismeri a B osztályt, B osztály objektuma A osztály egy adattagja. De A B nélkül is létezik (az autó utasai)

származtatás – A a B őse, B pedig A utódja (az autó egy jármű)

kompozíció (tartalmazás) - Az A osztályt nem lehet létrehozni a B osztály nélkül, a konstruktorban van benne (az autó motorja)

**17. Az absztrakt osztály jelentése, szerepe a programfejlesztés során, példák a használatukra**

Olyan osztály, melyet nem példányosítunk, mert van egy tisztán virtuális függvénye, amit majd a származtatott osztályok fognak felülírni, és az azokból létrehozott példányok használni. Például ha van egy Alakzatok osztályunk, ahol van egy területszámoló tisztán virtuális függvény, az abból származtatott Kör és Négyzet osztályok felül tudják írni. De arra nincs szükség, hogy egy Alakzat objektum területét számoljuk, az maradhat virtuális függvény, és az osztály ezáltal absztrakt. Ilyenkor az abstract szót ki kell tenni az osztály neve elé.

**18. Az interface szerepe a Java programokban, szintaxis, deklarálás, használatuk, stb.**

interface-ben nem lehetnek attribútumok, csak metódusok

class ClassName implements InterfaceName

származni csak egy osztályból lehet, de több interface is implementálható. Az interface-ből nem hozható létre objektum, csak az implementáló osztályból. Az interface-ben csak absztrakt metódusok vannak, azokat ki kell fejteni az implementáló osztályban