**jelmagyarázat:**

* **zöld=comment**
* **kék=kezeld extrán fentartásal de ez amugyis az egészre érvényes**

**……………………………………………………………………………………………………………………**

**mi a JVM?**

java virtual machine

**milyen primitiv tipusok vannak?**

byte, short, int, long, float, double, char, boolean

**Access level modifiers, láthatósági szintek?**

<https://docs.oracle.com/javase/tutorial/java/javaOO/accesscontrol.html> főleg táblázatok

no modfier vagyis ha nem irsz oda semmit az automatikusan package private lesz de megszívhatod névütközésel ekkor. ezért ált érdemes odairni valami láthatósági szintet

**object metodusai?**

* getclass: visszadja az osztályát
* hashcode: visszadja a hashcodeját, ha equals felülirod ezt is felül kell
* equals: visszadja hogy a két objektum egyenlő e.
* clone: másolatot készít az objektumról
* notify: felébreszt egy szálat
* notifyall: ua. csak az összes szálra
* wait: várakoztat egy szálat
* finalize: lefut ha a garbge collector úgy látja nincs már referenia az objektumra. részletesebben A mi a finalize metodus kérdésnél

**oop paradigmak?**

* Enkapszuláció
* Kompozició
* öröklödés
* Delegáció
* Polimorfizmus

**mit jelent a ? geneikusnál?**

wildcard

**Mit használhatunk trywithresorcesben?**

Ami megvalósítja Autoclosable interface-t.

**Mi az osztály (class)?**

blueprint ami alapján objektumokat létrehozuk. itt definiáljuk a objektum változoit és metodusait.

**Mi az objektum?**

egy példánya egy adott osztálynak.

**Mi az interface?**

**Mi az absztract osztály?**

**Mi a különbség az interface és az absztract osztály között?**

**Statikus és dinamikus típus jelentése?**

**Primitív és referencia típusok jelentése?**

**Singleton pattern elvi működése?**

Lényege, hogy az adott típusból biztos csak egy jöhessen létre. Singleton pattern két megvalósítási módja:

1. privát konstruktorunk van és egy statikus létrehozó metodusunk. A statikus metodus meghívja a privát konstruktort ha még nincs példányosítva az adott típusból. Ha már van példányosítva akkor viszont visszaadja az egyetlen létező példányt.
2. Enum segítségével mert abbol alapból csak egy jöhet létre.

**Factory pattern elvi müködése?**

**Mi a különbség a ++i és az i++ között?**

**Mit okoz a +op és a –op unáris operátorok használata?**

**Mi a belső és a beágyazott osztály?**

**Mit okoznzak a break, continue és a return utasitások?**

**Milyen vezérlési szerkezetek vannak?**

* feltételes elágazás:
  + if(feltétel)  
     {  
     utasitások;  
     }  
     else  
     {  
     utasitasok;  
     }
  + switch(amialapján)  
    case 1:{utasitás;break;} case 2:{ utasitás;break;}de
* ciklusok:
  + elöltesztelő
  + hátultesztelő
  + for
  + foreach

**Mi Garbage Collector?**

Nem használt erőforásokat felszabadítja ha észleli hogy valamire már nem mutat referencia.

**Mi a finalize metodus?**

Object class metodusa ami lefut ha a garbge collector úgy látja nincs már referenia az objektumra. Ha valamilyen resourcet le kell zárni azt itt érdemes.

**Mikor és miért kell String buildert használni?**

String az immutable vagyis nem lehet változtatni rajta. ezért ha szövegen akarunk változtatást végezni pl hozzáfüzés stb akkor stringbuildert kell használnunk.

**Bubble sort elvi müködése?**

**Quick sort elvi müködése?**

**Mit jelent a TDD?**

test driven development: Elöször tesztet írunk a specifikáció alapján. A tesztet futatjuk, elbukik. Megírjuk a kódot amire a teszt már nem bukik el. Csak a publikus metódusokat teszteljük.

**Mi az Enum?**

Felsorolás típus. Kötött konstans értékek tárolására használjuk.

tartalmazhat: metodusokat adattagokat.

implementálhat interfészt.

nem lehet publikus konstruktora és leszármazootja.

szerintem itt ez a lényeg de kicsit részletesseben java se2 diában van róla. a diába még volt ilyen de ezt nem teljesen értem: object metodusok, összehasonlíthatóak szerializákhatóak.

**Mi a különbség az error és exeption között?**

**Milyen exeptionok vannak?**

* checked
* unchecked

**Mi a különbség a checked és unchecked exeption között melyiket mikor érdemes használni?**

checkedet kötelező lekezelni uncheckedet nem.

asszem itt volt még valami futás meg fordítási ídővel kapcsolatban.

**Mik a láncolt kivételek?**

**Mivel léphetünk ki egy metodusból?**

* return
* exeption dobás

**Builder pattern mire jó, elvi müködése?**

**Mik az annotációk?**

**Mik a meta annotációk?**

olyan annotáció ami annotációra vonatkozik.

**Mi a reflection?**

**Mi a lambda kifejezés?**

**Mi a stream api?**

**Mit tutsz a szálkezelésről?**

**Alternatív switch-case(wiki.codigmentor,xyz oldalon a design patternes résznél) néhány mondatban?**

**Lazy initialization(wiki.codigmentor,xyz oldalon a design patternes résznél) néhány mondatba?**

**Strategy pattern (wiki.codigmentor,xyz oldalon a design patternes résznél) néhány mondatba?**

**Miért használunk generikus kódokat?**

* jobb tipusellenörzés fordítási időben
* castolás elkerülhető vele sok esetben
* kevesebb kódot kell írni mert nem kell megírnunk ugyanazt különböző tipusokra.

**Ezen kívűl át kell nézni gites anyagot mert lehet elöveszik megint néhyány ismétlő kérdés erejéig + a fontosabb java szintaktikákat.**