Magasszintű programozás 2 – elméleti rész lehetséges kérdései

1. számonkérés

**01001. Java kód fordításának lépései**

source code –> javac MyProgram.java -> bytecode -> java MyProgram.class-> futás

01002. JVM: mi az, miért fontos?

JVM: java virtual machine: ez teszi lehetővé a cross platformos futtatást

01003. Oldja fel és magyarázza, mire lehet használni: JRE, JDK, IDE

JRE: java runtime environment

JDK: java development kit

IDE: integrated development environment …

01004. main metódus: hogy néz ki, miért fontos, mi a szerepe?

psvm

a program futtatásának kezdőpontja

01005. Sorolja fel és jellemezze a primitív adattípusokat és a wrapper osztályaikat!

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

primitiv: do not share state with other primitive values

01006. String osztály

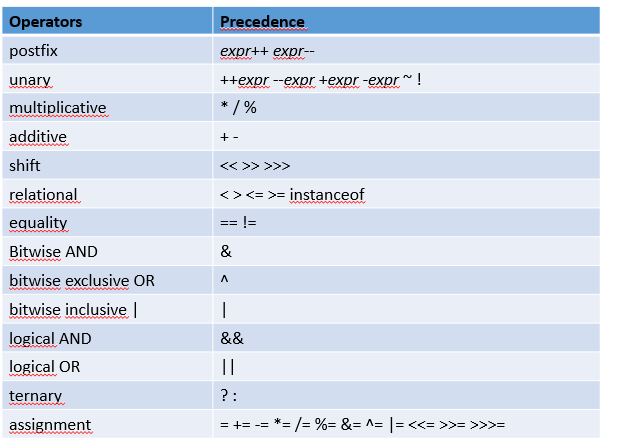
like a primitive data type, gives ability to store non mutable character strings

01007. Tömb (array) deklarációja, használata

int[] a = new int[5];

index …

01008. Precedenciatáblázat ☹



02009. Általánosan az objektum és osztály definíciója

…

02010. Absztrakció fogalma

a modellezésnél az alkalmazás számára fontos adattagokat, viselkedést leírjuk, míg a nem fontosakat elhagyjuk

**02011. Mit jelent az egységbezárás**

a kulvilás számára nem látható a metodusok implementációja

adatelemekt csak methodokon keresztul erhetjuk el

**02012. Mi a különbség az objektum állapota és viselkedése között?**

állapot: attributes

viselkedés: methods

**02013. Mi az osztályattribútum és osztálymetódus?**

osztály attrib == static attribute

osztalymetodus == static method

ezek az osztályhoz tartoznak és nem az egyedhez

**02014. Mi az a getter és setter?**

adatelrejtés segítik (accessor, mutatos is)

**02015. Mi az öröklés (általánosan)?**

két osztály közötti kapcsolat

egy gyerek osztály számára lehetőveé teszi hogy egy szülő osztálytól örökölje az adattagok tulajdonsáit és metodusait

**02016. Mi az aggregáció?**

rész-egész kapcsolat két osztály között-> egy osztály tartalmaz egy másik osztály objektumot (kurzus – hallgatok , mert a hallgato létezhet a kurzusnélkül)

**02017. Mi a kompozíció?**

aggregáció egy típusa, ahol az objektumok közötti kapcsolat erősebb

pl név – személy (mert a név nem létezhet a személy nélkül)

02018. Mi az asszociáció?

ebbe a halmazba tartozik a komp. és aggreg. is

két osztály között van kapcsolat

**02019. Mi az absztrakt osztály?**

ennek az osztálynak a metodusainak nincsenek implementácciójai

nem peldanyosithato

használatához egy alosztályt kell csinálni, amely implem. az absztrakt osztályt

**02020. Mit csinál a final kulcsszó?**

osztály: leaf osztály, nem lehet gyereke

adattag: nem változtatható

**02021. Soroljon fel 4 népszerű objektumorientált programozási nyelvet!**

java c# python c++

03022. Írjon egy Java osztályt, amely tartalmaz adattagokat és metódusokat ügyelve az elvárt névkonvenciókra.

…

**03023. Milyen névkonvenciókat kell használni a Java osztály, adattag, metódus és paraméterek definiálásánál?**

osztály: PascalCase

adattag, metodus, paraméter: camelCase

03024. Mi az a konstruktor? Mi történik, ha egy osztályhoz nem adunk meg konstruktort Javaban?

**03025. Hogyan példányosítunk Javaban egy osztályt?**

Person newPerson = new Person();

**03026. Java Garbage Collector mit csinál? Mit kell róla tudni?**

A java futtató környezet törli azokat az objektumokat, amelyeket úgy érzékeli már nem fogjuk használni (pl hatáskörön kívülre kerül)

automatikusan és időszakosan működik

**03027. Osztály tagjainak és metódusainak láthatósági módosítói Javaban**

public: mindenhonnan látható (class – package – subclass – world)

protected: (class – package – subclass)

modosito nelkuli: package szintű (class - package)

private: csak osztályon belül látható ( class )

**03028. Javaban a static kulcsszó használata**

nem a peldanyhoz hanem a osztályhoz tartozik a method **vagy adatag**

**03029. Javaban hogyan deklarálunk konstanst? Névkonvenció is kell.**

final double PI = 3.12512312;

**04030. Mi az a Java csomag? Hogyan adunk neki nevet?**

egy névtér amely rendszerezi az összetartozó classokat és interface-eket

fordított internet domain : com.example.mypackage

**04031. A Java csomag elemeinek milyen láthatósági módosítót lehet adni? Melyik mit jelent?**

public : midnenhonnan látható

no modifier: package-private (csak package-en belül)

**04032. Hogyan lehet használni (meghívni) a Java csomag elemeit?**

teljes név alapján

import the package member (import com.example.Person)

import the whole package (import com.example.\*)

04033. Soroljon fel 4-et a Java API beépített csomagjai közül!

java.io

java.

04034. (Java) Mi az az annotáció? Milyen formátuma van?

04035. Mit csinál az @Override annotáció Javaban??

05036. Javaban mit csinál a this kulcsszó? Hol használjuk, mire?

05037. Javaban mit csinál a super kulcsszó? Hol használjuk, mire?

05038. Miért fontos a Java Object osztálya? Mit kell róla tudni?

05039. (Java) Mi a baj a String osztállyal? Mit és hogyan használunk helyette?

05040. (Java) Mit kell tudni a numerikus típusok közötti konverzióról?

05041. (Java) Hogyan konvertálunk számot Stringgé és vissza?

05042. (Java) paraméterátadás módjáról mit kell tudni?

05043. Hogyan lehet tetszőleges számú paramétert átadni egy Java metódusnak?

05044. Mit csinál Javaban a return utasítás?

05045. Inicializáló mező és blokk Javaban. Mire valók, hogyan használjuk őket?

05046. Java Enum típus

06047. Java osztályok közötti öröklés

06048. Java osztályok közötti öröklés esetén a konstruktorok hogyan öröklődnek?

06049. Java osztályok közötti öröklés esetén mit jelent a metódus felülírása?

06050. Java szuperosztály és alosztály is definiál egy ugyanolyan nevű statikus metódust. Hívható-e és ha igen hogyan a szuperosztály metódusa? Ha csak a szuperosztály definiál statikus metódust, akkor az alosztállyal tudjuk-e hívni?

06051. Java szuperosztály és alosztály is definiál egy ugyanolyan nevű, de más típusú adattagot. Használhatom-e és ha igen, hogyan az alosztályból a szuperosztály adattagját?

06052. Mit jelent a konstruktorok lánca (chain of constructor)

0653. Java öröklésnél, metódus felülíráskor a láthatósági módosító változhat-e, és ha igen, hogyan?

06054. Mit jelent Javaban a polymorfizmus?

06055. Java absztrakt osztály, absztrakt metódus

07056. Az alkalmazásfejlesztés életciklusának lépései (felsorolás elég)

07057. Mi az az UML?

07058. Hogy néz ki az UML osztálydiagram?

07059. Az UML hogyan jelöli az osztályok és interfészek közötti öröklést? Mi örökölhet mitől és hány szülő lehet?

07060. Az UML hogyan jelöli az absztakt osztály és az interfészt?

08061. Mi az a Java interfész? Mit tartalmazhat (csak felsorolás)?

08062. Mire szolgál a Java interfész default metódusa?

08063. Java interfészek közötti öröklés

08064. Java interfész és osztály közötti öröklés

08065. Hogyan lehet használni egy Java interfészt?

08066. Java interfészekben definiált default metódusok ha az öröklés során konfliktusba kerülenk, annak mi lesz a feloldása?

09067. Mi az a Java generikus? Milyen szintaktikával lehet generikust definiálni? Hogyan lehet meghívni generikussal definiált elemet?

09068. Java generikus metódus, statikus metódus, és hívásuk

09069. Java generikus, bounded type

09070. Java generikus, wildcards

09071. Java generikus, type erasure

09072. Mi a kollekció (Javaban)? A Java API milyen eszközöket biztosít a kollekciókhoz?

09073. Rajzolja le a Java kollekció interfészeinek az öröklési rendszerét (a mappel együtt)!

09074. A kollekció elemeinek a rendezését hogyan lehet megvalósítani?

09075. Comparable interfész

09076. Comparator interfész

09077. Collections osztály

10078. Java beágyazott osztály

10079. Statikus beágyazott osztály

10080. Java local class

10081. Java anonymous class

11082. Java Funkcionális interfész

11083. Soroljon fel 5 beépített Java funkcionális interfészt

11084. Java lambda kifejezés

11085. Java metódus referencia

12086. Javadoc mit csinál, mire való, miért használjuk?

12087. Soroljon fel 5 Javadoc tag-et, és mutassa be, hogy mit csinálnak.

13088. Mi az a kivétel?

13089. Hogyan lehet kezelni a kivételt?

13090. A kivételeknek mi a 3 alapvető kategóriája? Melyikről mit kell tudni?

13091. Hogy néz ki a try-catch-final utasítás, és melyik része mit csinál?

13092. Mit csinál a try-with-resource utasítás?

13093. Hogyan lehet kivételt dobni?

13094. Hogyan hozhatunk létre Java kivétel osztályokat?

14095. Mi az az I/O Stream?

14096. Mi az a byte stream?

14096. Mi az a character stream?

14096. Mi az a buffered stream?

14097. Scanning and Formatting

14098. Mi a 3 standard stream a Java-ban? Melyik mire való?

14099. Mi a data stream?

14100. Mi az object stream?