

JAVA PROGRAMOZÁS

4. forduló

cl'ck

A kategória támogatója: Click Clock by BCS

Ismertető a feladatlaphoz

Közeleg az 5. forduló, figyelj az időpontokra!

Használd a naptárat:

KATEGÓRIÁIM

Összesen 10 kategóriára jelentkezted



Versenynaptár letöltése



Vagy figyeld kategóriánként az időpontokat (íme egy MINTA, hol találod):




● 3. FORDULÓ

A lezárt fordulókban eddig megszerzett pontok:

0/105 pont

**BOSCH**
Invented for life

Fordulók

Forduló	Pontok, időtartam	Feladat megoldható	Státusz
7. forduló	23 pont 25:00	 2023.11.28. 20:00-tól 2023.11.28. 20:35-ig	Feladatlap
6. forduló	23 pont 30:00	 2023.11.21. 20:00-tól 2023.11.21. 20:40-ig	Feladatlap
5. forduló	28 pont 25:00	 2023.11.14. 20:00-tól 2023.11.14. 20:35-ig	Feladatlap

Amennyiben olyan kategóriában játszol, ahol van csatolmány, de hibába ütközel a letöltésnél, ott valószínűleg a vírusirtó korlátoz, annak ideiglenes kikapcsolása megoldhatja a problémát. (Körülbelül minden 3000. letöltésnél fordul ez elő.)

Jó versenyzést kívánunk!

Ismerjük a nyelvet, de vajon az azt körülölelő eszközöket is?

A fordulóban elhangzó kérdések mindegyike Java 17 alapokon értelmezendő. Ha valamelyik kérdés JDK-ban használt eszközre kérdez rá, ott mindig az OpenJDK-t vegyük alapul.

Az említett verzió a következő linken tölthető le: <https://jdk.java.net/java-se-ri/17>

1. feladat 2 pont

Kérdés: Melyik Java verziótól érvényes/működőképes a következő futtatási mód?

```
public class HelloWorld {  
    public static void main(String[] args){  
        System.out.println("Hello OITM!");  
    }  
}
```

```
}  
}
```

```
java HelloWorld.java
```

Válasz

- ☐ Java 7
- ☐ Java 8
- ☐ Java 9
- ☐ Java 10
- ☐ Java 11

2. feladat 2 pont

Szerettük volna az alkalmazásunkat úgy elindítani, hogy minden egyes teljes GC ciklusban egy dump is készüljön. Hogyan tudjuk ezt utólag megtenni, ha tudjuk, hogy az alkalmazás PID (folyamat azonosítója) 3476?

Válasz

- ☐ jcmd 3476 +HeapDumpAfterFullGC
- ☐ jinfo -flag +HeapDumpAfterFullGC 3476
- ☐ jstat -t +HeapDumpAfterFullGC 3476
- ☐ jshell -C +HeapDumpAfterFullGC 3476

3. feladat 2 pont

Futó alkalmazásunkról szeretnénk egy heap dump-ot készíteni, viszont azt szeretnénk, hogy csak az "élő" objektumokról készítse ezt el egy heap.hprof nevű fájlba (természetesen hprof bináris formában). Hogyan tudjuk ezt megtenni, ha tudjuk, hogy az alkalmazás PID (folyamat azonosítója) 5634 ?

Válaszok

- ☐ Ezt csak fejlesztői eszköz segítségével lehet megcsinálni, csak azok képesek rá könnyedén.
- ☐ jmap -dump:format=b,live,file=heap.hprof 5634

- ☐ jmap -dump:format=b,live,output=heap.hprof 5634
- ☐ jmap -dump:format=b,alive,file=heap.hprof 5634
- ☐ jcmd 5634 GC.heap_dump heap.hprof
- ☐ jcmd 5634 GC.heap_dump filename=heap.hprof

4. feladat 2 pont

Kollégád segítséget kér tőled, azt állítja, hogy az egyik alkalmazásokban úgy néz ki, hogy egy deadlock alakult ki. Melyik parancs segítségével tudjátok bizonyítani a feltételezését, ha a Java folyamat azonosítója 13341 ?

Válaszok

- ☐ Nem lehet bizonyítani, ezt extra loggolás segítségével tudjátok kideríteni a feltételezett területen.
- ☐ jcmd 13341 Thread.print
- ☐ jcmd 13341 Thread.dump
- ☐ jstack -thread-dump 13341
- ☐ jstack 13341

5. feladat 2 pont

Hogyan indíthatunk el egy JFR (Java Flight Recorder) rögzítési folyamatot?

Válaszok

- ☐ Nem tudunk, mert még fejlesztési fázisban van és nem érhető el a végfelhasználók számára.
- ☐ Java alkalmazás indítása során a **java** parancson belül a **-XX:StartFlightRecording** kapcsolóval.
- ☐ Java alkalmazás indítása során a **java** parancson belül a **-XX:EnableFlightRecording** kapcsolóval.
- ☐ Futó alkalmazás esetén, ha ismerjük a folyamat azonosítóját (PID), akkor a **jcmd <pid-azonosító> JFR.start** paranccsal.
- ☐ Futó alkalmazás esetén, ha ismerjük a folyamat azonosítóját (PID), akkor a **jcmd <pid-azonosító> [JFR.run](#)** paranccsal.
- ☐ Futó alkalmazás esetén, ha ismerjük a folyamat azonosítóját (PID), akkor a **jfr <pid-azonosító> start** paranccsal.

- ☐ Futó alkalmazás esetén, ha ismerjük a folyamat azonosítóját (PID), akkor a **jfr <pid-azonosító> run** paranccsal.

6. feladat 2 pont

Melyik opcióval tudjuk bekapcsolni a GC (Garbage Collector) loggolását, amelyet egy gc.log nevű fájlba szeretnénk írni a java eszköz használata esetében?

Válasz

- ☐ -XX:+PrintGCDetails:file=gc.log
- ☐ -Xlog:gc=info:file=gc.log
- ☐ -XX:GCLog=gc.log
- ☐ -XX:VerboseGC=gc.log

7. feladat 2 pont

Szeretnénk beállítani az alkalmazásunk futtatása során, hogy hány "minor" GC ciklus után kerüljön egy objektum az úgynevezett fiatal (young) generációból az idős (old) generációba, melyik kapcsolóval tudjuk ezt megtenni?

Válasz

- ☐ -XX:MinMinorCycle
- ☐ -XX:YoungGenerationCycle
- ☐ -XX:MaxTenuringThreshold
- ☐ -XX:OldGenerationLimit

8. feladat 2 pont

Szeretnénk egy generációs GC-vel futtatott alkalmazásban követni a fiatal generációban lévő objektumok életkorát. Melyik java kapcsolóval tudjuk ezt megtenni?

Válasz

- ☐ java -XX:+PrintTenuringDistribution

- ☐ java -Xlog:gc+age*=trace
- ☐ java -XX:+PrintYoungAge
- ☐ java -Xlog:gc,age=trace

9. feladat 5 pont

Adott a következő bájtkód, mely kódrészletet tükrözi azt?

```
1 class hu.megmerettetes.bytecode.Feladat {
2     hu.megmerettetes.bytecode.Feladat();
3     Code:
4         0: aload_0
5         1: invokespecial #1           // Method java/lang/Object."<init>":
        ()V
6         4: return
7
8     public static void main(java.lang.String[]);
9     Code:
10        0: invokestatic #7           // Method doJob:()V
11        3: return
12
13    public static void doJob();
14    Code:
15        0: iconst_0
16        1: istore_0
17        2: iconst_0
18        3: istore_1
19        4: iload_1
20        5: bipush        20
21        7: if_icmpge      23
22       10: iload_0
23       11: iload_1
24       12: iadd
25       13: istore_0
26       14: iinc          0, -1
27       17: iinc          1, 1
28       20: goto          4
29       23: getstatic     #12           // Field
        java/lang/System.out:Ljava/io/PrintStream;
30       26: iload_0
31       27: invokevirtual #18           // Method
        java/io/PrintStream.println:(I)V
32       30: return
33 }
34
```

Válasz

- ☐ A.



```
1 package hu.megmerettetes.bytecode;
2
3 class Feladat {
4
5     public static void main(String[] args) {
6         doJob();
7     }
8
9     public static void doJob() {
10         int sum = 0;
11         for (int i = 0; i < 10; i++) {
12             sum += i;
13         }
14         System.out.println(sum);
15     }
16 }
17
```

☐ B.



```
1 package hu.megmerettetes.bytecode;
2
3 public class Feladat {
4
5     public static void main(String[] args) {
6         doJob();
7     }
8
9     public static void doJob() {
10         int result = 0;
11         for (int i = 0; i < 20; i += 2) {
12             result += i;
13         }
14         System.out.println(result);
15     }
16 }
17
```

☐ c.



```
1 package hu.megmerettetes.bytecode;
2
3 class Feladat {
4
5     public static void main(String[] args) {
6         doJob();
7     }
8
9     public static void doJob() {
10         int result = 1;
11         for (int i = 1; i < 20; i++) {
12             result *= i;
13             result--;
14         }
15         System.out.println(result);
16     }
17 }
18
```

☐ D.



```
1 package hu.megmerettetes.bytecode;
2
3 class Feladat {
4
5     public static void main(String[] args) {
6         doJob();
7     }
8
9     public static void doJob() {
10         int result = 0;
11         for (int i = 0; i < 20; i++) {
12             result += i;
13             result--;
14         }
15         System.out.println(result);
16     }
17
18
19 }
20
```

Megoldások beküldése