Szerializálás

1. Transient módosítójú attribútumok nem lehetnek primitív típusúak.
2. Szerializálható osztály private adattagja minden esetben szerializálódik.
3. Az abstract módosítójú osztályok is megvalósíthatják a Serializable interfészt.
4. Szerializálható osztály static attribútuma nem szerializálódik
5. A standard java.lang csomagban nincsenek olyan osztályok, amelyek példányai nem szerializálhatók.
6. A standard java.io csomagban nincsenek olyan osztályok, amelyek példányai nem szerializálhatók.
7. Szerializálás körkörös hivatkozású adatszerkezeten (pl. gyűrű) végtelen ciklusba kerül.
8. Szerializálás körkörös hivatkozású adatszerkezeten (pl. gyűrű) kivételt dob.
9. Az abstract módosítójú osztályok is megvalósíthatják a Serializable interfészt.
10. Egy final osztály csak akkor valósíthatja meg a Serializable interfészt, ha implementálja a void write(OutputStream os) metódust.
11. A void write(OutputStream os) metódus implementálása nélkül is meg lehet valósítani a Serializable interfészt.
12. Egy final osztály csak akkor valósíthatja meg a Serializable interfészt, ha implementálja a void read(InputStream is) metódust.
13. Szerializálható osztály protected adattagja minden esetben szerializálódik.
14. A standard java.lang csomagban van olyan osztály, amely példányai nem szerializálhatók.
15. A void read(InputStream is) metódus implementálása nélkül is meg lehet valósítani a Serializable interfészt.
16. Szerializálás körkörös hivatkozású adatszerkezeten (pl. gyűrű) nem dob kivételt.
17. Szerializálható osztályban definiált private módosítójú példányattribútum sosem szerializálódik
18. Szerializálás körkörös hivatkozású adatszerkezeten (pl. gyűrű) nem kerül végtelen ciklusba.
19. Az abstract módosítójú osztályok nem valósíthatják meg a Serializable interfészt.
20. Ha egy FileReadert egy BufferedReaderbe csomagolunk, akkor a BufferedReader bezárása után (close metódus meghívása) a FileReadert is be kell zárnunk.
21. Szerializálható osztály final módosítójú példányattribútuma sosem szerializálódik
22. Az X osztály nem szerializálható. Az Y osztály X leszármazottja és megvalósítja a Serializable interfészt. Ha Y-t szerializáljuk, akkor az X-től örökölt attribútumok is szerializálódnak.

23. Az X osztály nem szerializálható. Az Y osztály X leszármazottja és megvalósítja a Serializable interfészt. Ha Y-t szerializáljuk, akkor az X-től örökölt attribútumok nem szerializálódnak.
24. A standard java.io csomagban van olyan osztály, amely példányai nem szerializálhatók.
25. Szerializálható osztályban definiált protected módosítójú pédányattribútum sosem szerializálódik
26. A transient módosítójú attribútumok mindig private-ok is egyben.
27. Ha egy FileReadert egy BufferedReaderbe csomagolunk, akkor a BufferedReader bezárása után (close metódus meghívása) a FileReader automatikusan bezáródik.
28. Szerializálható osztály public adattagja minden esetben szerializálódik.

Megoldás

1. h
2. h
3. i
4. i
5. h
6. h
7. h
8. h
9. i
10. h
11. i
12. h
13. h
14. i
15. i
16. i
17. h
18. i
19. h
20. h
21. h
22. h
23. i
24. i
25. h
26. h
27. i
28. h

Generikus osztályok és kollekciók

1. A for (Y y : d) típusú (foreach) ciklusban a d változóval referált objektumnak nem kell megvalósítania a java.util.Iterator interfészt.
2. A Java generikus osztályok példányosításakor sosem lehet primitív típus a template-paraméter.
3. A for (Y y : r) típusú (foreach) ciklusban az y változóval referált objektumnak meg kell valósítania a java.util.Collection interfészt.
4. Foreach ciklusban (pl. for (A a : x){...}) nem szabad az iterált kollekcióhoz (a példában x) új elemet hozzáadni, mert exception kapunk.
5. A for (Z z : s) típusú (foreach) ciklusban a z változóval referált objektumnak nem kell megvalósítania a java.util.Comparator interfészt.
6. Csak primitív típusokon értelmezett a "természetes rendezés" (natural ordering).
7. Primitív típusok csomagoló osztályain (wrapper class, pl. Integer vagy Double) nem értelmezett a természetes rendezés (natural ordering).
8. A for (E e : w) típusú (foreach) ciklusban a w változóval referált objektumnak nem kell megvalósítania a java.util.Comparator interfészt.
9. Generikus osztály példányosításakor lehet másik generikus osztály a paraméter.
10. A for (S x : z) fejlécű for ciklusban a z referencia csak tömbre vagy a JDK-val szállított gyári kollekciók példányaira referálhat.
11. A for (R r : t) típusú (foreach) ciklusban az r változóval referált objektumnak nem kell megvalósítania a java.util.Iterator interfészt.
12. A for (F f : g) típusú (foreach) ciklusban a g változóval referált objektum meg kell valósítsa a java.util.Comparator interfészt.
13. Primitív típusok csomagoló osztályain (wrapper class, pl. Integer vagy Double) értelmezett a természetes rendezés (natural ordering).
14. Egy kollekció elemeit iterátorral érjük el. A kollekciót nem lehet úgy módosítani, hogy utána az iterátor a hasNext() metódushívásra ne dobjon kivételt.
15. Generikus osztály példányosításakor nem lehet másik generikus osztály a sablon paramétere.
16. A for (Z z : s) típusú (foreach) ciklusban az s változóval referált objektumnak meg kell valósítania a java.util.Collection interfészt.
17. Primitív típus lehet generikus osztály template-paramétere.
18. A for (Q q : c) típusú (foreach) ciklusban az q változóval referált objektum meg kell valósítsa a java.util.Iterator interfészt.
19. A for (G g : h) típusú (foreach) ciklusban a h változóval referált objektum meg kell valósítsa a java.util.Iterator interfészt.
20. Egy kollekció elemeit iterátorral érjük el. A kollekciót lehet úgy módosítani, hogy utána az iterátor a hasNext() metódushívásra ne dobjon kivételt.

21. Csak a Comparable interfészt megvalósító osztályokon értelmezett a természetes rendezés (natural ordering).
22. Egy kollekció elemeit iterátorral érjük el. A kollekciót semmilyen módon nem lehet úgy módosítani, hogy utána az iterátor a hasNext() metódushívásra ne dobjon kivételt.
23. Csak a Comparator interfészt megvalósító osztályokon értelmezett a természetes rendezés (natural ordering).
24. A for (W w : d) típusú (foreach) ciklusban a w változóval referált objektum meg kell valósítsa a java.util.Comparator interfészt.
25. A for (W w: c) típusú (foreach) ciklusban a c változóval referált objektum meg kell valósítsa a java.util.Collection interfészt.
26. A for (Q q : k) típusú (foreach) ciklusban a k változóval referált objektumnak nem kell megvalósítania a java.util.Collection interfészt.
27. A for (X x : h) típusú (foreach) ciklusban az x változóval referált objektumnak nem kell megvalósítania a java.util.Collection interfészt.

Megoldás

1. i
2. i
3. h
4. i
5. i
6. h
7. h
8. i
9. i
10. h
11. i
12. h
13. i
14. h
15. h
16. h
17. h
18. h
19. h
20. i
21. i
22. h
23. h
24. h
25. h

26. i
27. i

Szálkezelés

1. Nem fordulhat elő, hogy két szál (T1 és T2) ugyanazon objektum ugyanazon synchronized metódusát futtatva T1 T2 sorrendben lép be, de T2 T1 sorrendben lép ki.
2. wait metódust csak azon az objektumon lehet hívni, aminek a hívást végrehajtó szál a monitorában tartózkodik.
3. synchronized módosítójú metódusban nem lehet wait() metódust hívni
4. Egy objektum notifyAll() metódusának meghívásakor a végrehajtó szál kilép az objektum monitorából.
5. Ha egy szál véget ért, nem lehet újraindítani.
6. Egy szál egyszerre csak egy objektum monitorában tartózkodhat.
7. A wait() függvény csak olyan objektumon hívható, amelyre rászinkronizáltunk.
8. Ha van két szálunk, akkor a join metódusaikat csak az elindításuk sorrendjében szabad meghívni.
9. Egy objektum notify() metódusának meghívásakor a végrehajtó szálnak nem kell az objektum monitorában tartózkodnia.
10. Egy objektum wait() metódusának meghívásakor a végrehajtó szálnak nem kell az objektum monitorában tartózkodnia.
11. synchronized blokkok nem ágyazhatók egymásba.
12. Szálat a run() metódus meghívásával indíthatunk.
13. Amikor egy szál véget ér, a szálat reprezentáló objektum start metódusának meghívása kivételdobást erdményez.
14. Szálak nem képesek saját magukat közvetlenül waiting állapotból notify-jal felébreszteni.
15. Minden objektumnak van wait() metódusa.
16. Egy szálat csak a szál start() függvényével szabad elindítani, és csak a stop() függvényével szabad leállítani.
17. Nincs olyan várakozó szál, amelyik egyből RUNNABLE állapotú lesz a notifyAll() hatására.
18. Amikor egy szál véget ér, a szálat reprezentáló objektum bármely metódusának meghívása kivételdobást erdményez.
19. Egy objektum notify() metódusának meghívásakor a végrehajtó szálnak az objektum monitorában kell tartózkodnia.
20. Egy szál egy időben több objektum monitorában is tartózkodhat.
21. A t szál belépett x objektum szinkronizált foo metódusába. Az x foo metódusát semmilyen körülmények között nem hajthatja végre másik szál, amíg t vissza nem tért a metódusból.
22. Egy objektum wait() metódusának meghívásakor a végrehajtó szálnak az objektum monitorában kell tartózkodnia.

23. Előfordulhat, hogy két szál (T1 és T2) ugyanazon objektum ugyanazon synchronized metódusát futtatva T1 T2 sorrendben lép be, de T2 T1 sorrendben lép ki.
24. synchronized blokkok egymásba ágyazhatók.

Megoldás

1. h
2. i
3. h
4. h
5. i
6. h
7. i
8. h
9. h
10. h
11. h
12. h
13. i
14. i
15. i
16. h
17. i
18. h
19. i
20. i
21. h
22. i
23. i
24. i