JAVA 11

2. forduló

A kategória támogatója: IBM

RENDELKEZÉSRE ÁLLÓ IDŐ:

10:00

Ismertető a feladathoz

Fontos információk

Ha kifutsz az adott feladatlap kitöltésére rendelkezésre álló időből, a felület **automatikusan megpróbálja beküldeni** az addig megadott válaszokat

A kérdésekre **mindig van helyes válasz**! Ha csak egy helyes válasz van az adott kérdésre, radio button-os választási lehetőségeket fogsz látni.

Olyan kérdés viszont nincs, amelyre az összes válasz helyes!

Egyéb információkat a versenyszabályzatban találsz!

Felhasznált idő: 02:06/10:00 **Elért pontszám:** 0/40

1. feladat 0/5 pont

Válaszd ki azokat a kollekciókat amelyek megvalósítják a Collection interfacet!

Válaszok

- [x] ArrayDeque
- [x] Vector
- [x] LinkedHashSet
- [ ] TreeMap

Magyarázat

A Map-ek nem valósítják meg a Collection interface-t.

2. feladat 0/5 pont

Mit kapunk a Collections.singletonList() meghívásával?

Válasz

- Többelemű megváltoztathatatlan listát
- Egyelemű megváltoztatható listát
- Egyelemű megváltoztathatatlan listát
- Többelemű megváltoztatható listát

Magyarázat

A Collections.singletonList-nek egy paramétere van. Ez a paraméter lesz a lista egyetlen eleme, mindezen felül a lista megváltoztathatatlan lesz.

3. feladat 0/5 pont

Hogyan határozza meg a Stream.toArray() a létrehozandó tömb méretét?

Válasz

- A tömb elemeinek számából
- Nem tudja
- A tömb hosszából
- A stream hosszából

Magyarázat

Mivel a tömb még nem létezik, ezért a Stream.toArray() függvény a terminális művelet végrehajtása után, a Stream hosszából találja ki a létrehozandó tömb méretét.

4. feladat 0/5 pont

Válaszd ki a szálbiztos kollekciókat az alábbiak közül!

Válaszok

- [x] ConcurrentHashMap
- [ ] HashMap
- [x] Stack
- [ ] ArrayList

Magyarázat

A ConcurrentHashMap szálbiztos anélkül, hogy szinkronizálná az egész Map-et, nincs lezárás objektum szinten. A Stacknél Objektum szintű szinkronizálás van.

5. feladat 0/5 pont

Hogyan csinálhatunk Listát amely tartalmazza az 1 és 2 számokat?

Válaszok

- [ ]
```
Lists.of(1,2);
```
- [x]
```
new ArrayList<>() {{
  add(1);
  add(2);
}};
```
- [x]
```
Stream.of(1,2).collect(Collectors.toList());
```
- [x]
```
Arrays.asList(1,2);
```

Magyarázat

Nincs Lists osztály. Egyes számban a List.of() viszont helyes lenne.

6. feladat 0/5 pont

Mi jellemző a java.util.TreeSet-re?

Válaszok

- [x] Egy piros-fekete fát reprezentál
- [ ] Mindig a legjobb teljesítményt érhetjük el vele a halmazok közül
- [x] A TreeSet egy kollekció
- [x] Az elemeken rendezetten tudunk végigiterálni

Magyarázat

A legjobb teljesítményt általában a HashSet-tel érhetjük el. A TreeSet akkor lehet hatékonyabb, ha elrontottuk a tartalmazott osztályok hash függvényét.

7. feladat 0/5 pont

Hogyan rendezhetjük a következő listát List<Integer> example = List.of(3, 2, 5, 6)?

Válasz

- ( ) example.sort();
- ( ) Lists.sort(example);
- (x) example = example.stream().sorted().collect(Collectors.toList());
- ( ) Collections.sort(example);

Magyarázat

A List.sort() metódus egy Comparator-t vár paraméterként. Nem létezik Lists osztály Java-ban. A Collections.sort() létezik, de a List.of() immutable listával tér vissza, így ott kivételt kapunk.

8. feladat 0/5 pont

Milyen igényekre ad implementációt a Collection API?

Válaszok

- [x] Megváltoztathatatlanság (immutability)
- [x] Szálbiztosság (thread-safety)
- [x] Teljesítmény
- [ ] JSON szerializáció

Magyarázat

A Collection API-nak nem célja semmilyen szerializációs megoldást nyújtani.

A teljesség igénye nélkül néhány példa, amivel a Collection API a többi igényt támogatja:

Immutability

Decorator metódusok:

- Collections.unmodifiableCollection()
- Collections.unmodifiableList()
- Collections.unmodifiableSet()
- Collections.unmodifiableMap()

Factory metódusok:

- List.of()
- Set.of()
- Map.of()

Szálbiztosság

Decorator metódusok:

- Collections.synchronizedCollection()
- Collections.synchronizedList()
- Collections.synchronizedSet()
- Collections.synchronizedMap()

Szálbiztos implementációk:

- Stack
- CopyOnWriteArrayList, CopyOnWriteArraySet
- ConcurrentHashMap

Bár szigorúan véve a fentiek a Stack kivételével nem a Collection API része, mert a java.util.concurrent csomagban vannak.

Teljesítmény

Adott feltételek melletti teljesítményre optimalizált implementációk:

- ArrayList - sok index szerinti hozzáférés esetén
- LinkedList - sok beszúrás/törlés esetén
- HashSet, HashMap - feltéve, hogy jól implementáltuk a tárolt objektumok hashCode() metódusát
- LinkedHashSet, LinkedHashMap - majdnem olyan gyors, mint az előző kettő, de iterálásnál megtartják a beszúrási sorrendet
- EnumSet, EnumMap - enum-ok hatékony tárolására