HATÉKONY JAVA PROGRAMOZÁS

1. forduló

MSCI

A kategória támogatója: MSCI

RENDELKEZÉSRE ÁLLÓ IDŐ:

10:00

Ismertető a feladathoz

Fontos információk:

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

Kérjük, hogy a feladatok szövegeit **ne másold** és a böngésződ fejlesztő eszközét/ konzolját se nyisd meg feladatmegoldás közben! Mindkettő kizárást vonhat maga után.

Minden forduló után a **megoldások csütörtök reggel 8 órakor** lesznek elérhetőek.

A megoldásokkal kapcsolatos esetleges **észrevételeket a megoldások megjelenését követő kedd éjfélig** várjuk.

A több válaszlehetőségű feleletválasztós kérdéseknél járnak **részpontszámok, ha egyik rossz választ sem jelölöd be**.

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

Minden feladatsornak van egy **becsült minimum megoldási ideje** (legalább a feladat elolvasási ideje). Aki ennél rövidebb idő alatt oldja meg, kizárható a versenyből.

Az első kategória után, amelynek a feladatlapját megoldod a fordulóban, kapni fogsz egy 2-3 perc alatt kitölthető **kérdőívet**. Az ezekből összeállított piackutatás legfontosabb eredményeit a díjátadót követően Veled is megosztjuk majd. Formáljuk közösen a piacot!

Felhasznált idő: 02:07/10:00 **Elért pontszám:** 0/5

1. feladat 0/1 pont

Adott az alábbi Ticker osztály. Melyik kódrészlet ad hatékony megvalósítást a hashcode és equals metódusokra?

```
class Ticker {
        String symbol;
        BigDecimal tradedPrice;
}
```

Válasz

```
@Override
public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Ticker ticker = (Ticker) o;
        return Objects.equals(symbol, ticker.symbol);
}

@Override
public int hashCode() {
        return Objects.hash(symbol, tradedPrice);
}
```

A

```
@Override
public boolean equals(Object o) {
        return (this == o);
}

@Override
public int hashCode() {
        return 1;
}
```

B

```
@Override
public boolean equals(Object o) {
        return true;
}

@Override
public int hashCode() {
        return super.hashCode();
}
```

C

```
@Override
public boolean equals(Object o) {
        if (this == o) return true;
```

```
        if (o == null || getClass() != o.getClass()) return false;
        Ticker ticker = (Ticker) o;
        return Objects.equals(symbol, ticker.symbol) &&
                Objects.equals(tradedPrice, ticker.tradedPrice);
    }

    @Override
    public int hashCode() {
        return super.hashCode();
    }
```

D

```
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        Ticker ticker = (Ticker) o;

        if (symbol != null ? !symbol.equals(ticker.symbol) : ticker.symbol != null) return false;
        return tradedPrice != null ? tradedPrice.equals(ticker.tradedPrice) : ticker.tradedPrice
    }

    @Override
    public int hashCode() {
        int result = symbol != null ? symbol.hashCode() : 0;
        result = 31 * result + (tradedPrice != null ? tradedPrice.hashCode() : 0);
        return result;
    }
```

E

Magyarázat

A. válasz hibás, mert egyenlőség esetén is lehet különböző hashcode

B. válasz hibás, mert mindig hash ütközést okoz

C. válasz hibás, mert egyenlőség esetén is lehet különböző hashcode

D. válasz hibás, mert egyenlőség esetén is lehet különböző hashcode

E. válasz megfelel a Java nyelv által támasztott követelményeknek (hashcode-equals contract)

2. feladat 0/1 pont

Melyik algoritmus állítja elő az eredményt a leghatékonyabban?

Válasz

```
String result = "";
for (String item : list) {
        result += (item + ",");
}
if (result.length() > 0) {
        result = result.substring(0, result.length() - 1) + ";";
}
System.out.println(result);
```

```
String result = "";
list.forEach(
                item -> result += (item + ",")
);
if (result.length() > 0) {
        result = result.substring(0, result.length() - 1) + ";";
}
```

```
Iterator<String> iterator = list.iterator();
StringBuilder builder = new StringBuilder();
while (iterator.hasNext()) {
        builder.append(iterator.next());
        if (iterator.hasNext()) {
                builder.append(",");
        } else {
                builder.append(";");
        }
}
String result = builder.toString();
```

```
String result = "";
list.parallelStream().forEach(
                item -> result += (item + ",")
);
if (result.length() > 0) {
        result = result.substring(0, result.length() - 1) + ";";
}
```

Magyarázat

A helyes válasz az egyedüli, amelyik hatékony String összefűzést használ (StringBuilder). A parallelStream alapú megoldás véletlenszerűen hibás eredményt adhat.

3. feladat 0/1 pont

Melyik állítások igazak a try-with-resource konstrukcióra?

Válaszok

- [ ] ritkán használjuk, mert nem hatékony
- [ ] csak egy erőforrás felszabadítására képes
- [x] több erőforrás felszabadítására is képes
- [x] általunk megírt erőforrások felszabadítására is alkalmas
- [x] catch blokkal együtt is használható
- [x] finally blokkal együtt is használható
- [ ] nem mindig használható, mert nem szálbiztos
- [ ] csak teljesítménykritikus környezetben használatos, mert a Java nyelv esetében az automatikus szemétgyűjtés ezt kiváltja

Magyarázat

A Java nyelv specifikációja egyértelműen megfogalmazza a konstrukció működését.

4. feladat 0/1 pont

Válaszd ki azokat az állításokat, amelyek egyéb körülményektől függetlenül biztosan igaz értékkel térnek vissza Java 11-ben!

Válaszok

- [ ] Integer.valueOf(255) == Integer.valueOf(255)
- [x] Integer.valueOf(8) == Integer.valueOf(8)
- [ ] new Integer(127) == new Integer(127)
- [ ] new Long(103) == new Long(103)
- [x] Byte.valueOf((byte)97) == Byte.valueOf((byte)97)
- [x] Byte.valueOf("2") == Byte.valueOf("1") + 1
- [x] Integer.valueOf(200) == Integer.valueOf(199) + 1

- [ ] "200" == new String("2") + "0" + "0"
- [ ] new Object().equals(new Object())

Magyarázat

Futtassuk le a kódrészleteket.

5. feladat 0/1 pont

Mit ír ki az alábbi kódrészlet?

```
System.out.println(Math.min(Double.MIN_VALUE, 0.0d));
```

Válasz

- ◯ 0.0
- ◯ -4.9E-324
- ◯ -1.7976931348623157E308
- ◯ NaN

Magyarázat

Érdemes lefuttatni a kódrészletet, a Double.MIN_VALUE a legkisebb ábrázolható pozitív szám.