JAVA 11

1. forduló

A kategória támogatója: IBM

RENDELKEZÉSRE ÁLLÓ IDŐ:

15:00

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: 02:06/15:00 **Elért pontszám:** 0/45

1. feladat 0/5 pont

Az alábbiak közül mi a helyes módja a d1 és d2 double típusú értékek egyenlőségvizsgálatának?

Válasz

- [ ] d1 == d2
- [ ] Double.valueOf(d1) == Double.valueOf(d2)
- [ ] Double.compare(d1, d2) == 0
- [ ] Double.valueOf(d1).compareTo(Double.valueOf(d2)) == 0
- [x] Math.abs(d1 - d2) < 1e-5
- [ ] Double.valueOf(d1).isCloseTo(Double.valueOf(d2))
- [ ] Math.isCloseTo(d1, d2)

Magyarázat

A lebegőpontos számításoknál kerekítési hibák léphetnek fel, így a közvetlen egyenlőségvizsgálat (== és compare()) megbízhatatlan. isCloseTo metódus nem létezik sem a Double, sem a Math osztályokban. Java SE-ben a helyes megoldás, hogy megnézzük, a két érték kellően közel van-e egymáshoz. Erre bizonyos library-kben kapunk beépített metódusokat.

2. feladat 0/5 pont

Az alábbi kifejezések közül melyikek értékelődnek ki biztosan igazra?

Válaszok

- [x] Short.valueOf((short) 15) == 15
- [x] Integer.valueOf(125) == Integer.valueOf(125)
- [ ] Integer.valueOf(11) == new Integer(11)
- [x] Long.valueOf(476232) == 476232
- [ ] Integer.valueOf(220) == Integer.valueOf(220)
- [x] Integer.valueOf(220) == 220
- [x] new Integer(220) == 220L
- [x] (short) 15 == 15
- [x] Integer.valueOf(11) == Integer.valueOf(11)
- [ ] Short.valueOf((short) 15).equals(15)

Magyarázat

Amennyiben egy szám (short/int/long) primitív típust hasonlítunk össze egy wrapper objektummal, akkor == operátor használata esetén a wrapper objektum kerül kicsomagolásra (unboxing) és számként hasonlítja össze őket, ezért short, int és long (és bármelyik wrappere) esetén is teljesül az egyenlőség. Az Integer objektumok a –128..127 tartományban cache-elve vannak, amit ebben a tartományban a valueOf metódus hívásakor felhasznál (ezért kétszer ugyanaz a számra hívott valueOf ugyanazt a referenciát adja vissza), new operátor esetén pedig mindig új objektumot hoz létre.

3. feladat 0/5 pont

Az alábbi kifejezések közül melyikek értékelődnek ki igazra?

Válaszok

- [ ] new BigDecimal("1.25").multiply(new BigDecimal("2")).equals(new BigDecimal("2.5"))
- [x] BigDecimal.TEN.compareTo(BigDecimal.valueOf(10.0000)) == 0
- [ ] BigDecimal.valueOf(10).equals(10)
- [x] BigDecimal.TEN == BigDecimal.valueOf(10L)
- [ ] BigDecimal.valueOf(582) == 582L

Magyarázat

BigDecimal-ok szorzása esetén 1.25 * 2 = 2.50 -ra értékelődik ki, ami (equals szerint) nem egyenlő 2.5-el a tizedesjegyek különböző száma miatt. Ugyanakkor a compareTo 0-t ad vissza a két szám összehasonlítására, mivel ebben az esetben a BigDecimal “scale” értéke nem számít. Ha BigDecimal-t hasonlítunk össze (equals segítségével) 10-zel (int primitív típus, boxing miatt Integer típusú lesz), akkor két eltérő típusú példányt hasonlítunk össze, amire az equals mindig hamis eredményt ad. A BigDecimal osztályban a 0, 1 és 10 konstansok, a BigDecimal.valueOf(10L) pedig ugyanazt az objektumot adja vissza, amire a BigDecimal.TEN hivatkozik. Az == operátorral való összehasonlítás egy long literállal nem fordul le.

4. feladat 0/5 pont

Mi a kimenetele a következő kódrészletnek?

```
StringJoiner example = new StringJoiner(":","{","}");
example.add("Panda");
example.add("Bambus");
System.out.println(example);
```

Válasz

- [ ] Egy üres szöveg

- Panda:Bambus
- java.util.StringJoiner@hashcode
- {Panda:Bambus}

Magyarázat

A StringJoiner osztály segítségével összefűzhetünk Stringeket. Meghatározhatunk elválasztókaraktert, előtagot és utótagot is.

5. feladat 0/5 pont

Mi a különbség a String strip() és a String trim() metódus között?

Válasz

- A trim a szöveg közepéről is eltávolítja a felesleges szóközöket
- A strip csak a szöveg végéről távolítja el a felesleges szóközöket
- A strip() Unicode szabványt használ a felesleges szóközök megtalálásához
- Nincs különbség köztük

Magyarázat

A strip() metódus a Character.isWhitespace() metódussal ellenőrzi, hogy egy adott karakter whitespace-e vagy sem. Ez a metódus pedig Unicode szabvánnyal működik.

A trim() metódus az U+0020-nál kisebb vagy egyenlő kódú karaktereket távolítja el.

6. feladat 0/5 pont

Mire használjuk a var kulcsszót?

Válasz

- Lokális változó deklarálására
- Class létrehozására
- Javaban nincs var kulcsszó
- Metódus visszatérési értékére

Magyarázat

A var kulcsszót akkor használjuk, amikor egy változó létrehozásánál nem szeretnénk definiálni a változó típusát. Ilyenkor a változó típusát fordító az értékadás jobb oldalának típusából fogja meghatározni.

7. feladat 0/5 pont

Mit fog kiírni a konzol, ha a következő programrészletet futtatjuk?

```
String amIBlankOrNot = null;
boolean isBlank = amIBlankOrNot.isBlank();
System.out.println(isBlank);
```

Válasz

- ○ true
- ○ false
- ◉ Nem fog lefutni, NullPointerException-be fog ütközni a 2. sornál
- ○ Nem fog lefutni, NullPointerException-be fog ütközni a 3. sornál
- ○ Fordítási hiba a 2. sorban

Magyarázat

Az amIBlankOrNot változó értéke null, ezért null.isBlank() NullPointerException-t eredményez.

8. feladat 0/5 pont

Válaszd ki azt az implementációt amelyik 2.5 -öt ír ki a konzolra!

Válaszok

- ☐
```
IntToDoubleFunction halfOfIt= (var x) -> (double) (x / 2);
System.out.println(halfOfIt.applyAsDouble(5));
```

- ☑
```
IntToDoubleFunction halfOfIt = (var x) -> (double) x / 2;
System.out.println(halfOfIt.applyAsDouble(5));
```

```
IntFunction<Double> halfOfIt = (var x) -> (double) (x / 2);
System.out.println(halfOfIt.apply(5));
```

```
IntFunction<Double> halfOfIt = (var x) -> (double) x / 2;
System.out.println(halfOfIt.apply(5));
```

Magyarázat

Az x mint egész szám érkezik, ezért figyelnünk kell arra, hogy ne az osztás eredményét cast-oljuk.

(double) (x / 2) jellegű válaszok esetén:

(int) 5 / 2 = 2

(double) 2 = 2.0

Ezért “2.0” lesz kiírva a konzolra.

(double) x / 2 jellegű válaszok esetén:

(double) 5 = 5.0

5.0 / 2 = 2.5

Ezért “2.5” lesz kiírva a konzolra.

9. feladat 0/5 pont

Mi lesz kiírva a kérdőjelek helyére?

```
jshell> var userName = "bob"
userName ==> "bob"

jshell> userName.replaceFirst("b","d")
$2 ==> "dob"

jshell> var username2 = userName
???
```

Válasz

- username2 ==> "bob"
- username2 ==> "dob"

username2 ==> userName

username2 ==> "dod"

Magyarázat

A userName változó értéke nem változik a replaceFirst("b","d") meghívása után, ezért a userName változó végig a "bob" értéket tartalmazza. Amikor username2-nek értékül adjuk a userName-et a jshell a userName értékét fogja kiírni, ami "bob".