JAVA 11

6. forduló

A kategória támogatója: IBM

RENDELKEZÉSRE ÁLLÓ IDŐ:

15:00

Ismertető a feladathoz

Válaszolj az alábbi kérdésekre!

Felhasznált idő: 02:06/15:00 **Elért pontszám:** 0/35

1. feladat 0/5 pont

Meg szeretnénk keresni az összes "log"-al kezdődő, "txt" kiterjesztésű fájlt egy mappában. Adott a következő kódváz:

```
Set<Path> searchForLogFiles(String logsDirectory) {
  try (Stream<Path> paths = /* 1 */) {
    return paths
      .filter(path -> Files.isRegularFile(path) && /* 2 */)
      .collect(Collectors.toSet());
  } catch (IOException e) {
    e.printStackTrace();
  }
  return null;
}
```

Mit kell írni a /* 1 */ helyére?

Válasz

- `Files.lines(Path.of(logsDirectory))`

- [ ] `Files.list(Path.of(logsDirectory))`
- [ ] `Paths.lines(Path.of(logsDirectory))`
- [ ] `Paths.list(Path.of(logsDirectory))`

Magyarázat

Files.list(Path dir) visszaadja a mappa tartalmát, a Files.lines(Path file) -t fájlolvasásra használjuk. A Paths osztálynak nincsenek ilyen metódusai.

2. feladat 0/5 pont

Meg szeretnénk keresni az összes "log"-al kezdődő, "txt" kiterjesztésű fájlt egy mappában. Adott a következő kódváz:

```
Set<Path> searchForLogFiles(String logsDirectory) {
  try (Stream<Path> paths = /* 1 */) {
    return paths
      .filter(path -> Files.isRegularFile(path) && /* 2 */)
      .collect(Collectors.toSet());
  } catch (IOException e) {
    e.printStackTrace();
  }
  return null;
}
```

Mit kell írni a /* 2 */ helyére?

Válaszok

- [x] `path.getFileName().toString().startsWith("log") && path.getFileName().toString().endsWith(".txt")`
- [x] `path.getFileName().toString().matches("log.*\.txt")`

- [ ] `path.startsWith("log") && path.endsWith(".txt")`
- [ ] `path.matches("log.*\.txt")`

Magyarázat

A path ebben a kontextusban java.nio.file.Path típusú, így előbb Stringgá kell konvertálni a getFileName() metódussal. Ezután össze tudjuk hasonlítani a kért Stringgel.

A feladatért akkor is jár a teljes pontszám, ha csak az egyik helyes válaszlehetőséget jelölted meg, de nem jelöltél helytelent, ugyanis maradt egy véletlen hiba a kódrészletben (nem volt escapelve a stringben a backslash (\\ kellett volna), így mindkét matches válasz helyből rossz volt, ha valaki ezt észreveszi).

3. feladat 0/5 pont

Minden log fájlban maximum MAXIMUM_LOG_CAPACITY sort tárolhatunk. A következő kódot írtuk:

```
boolean maximumLinesCapacityNotExceed(Path logFile) throws IOException {
  long numberOfLines = /* 1 */;
  return numberOfLines < MAXIMUM_LOG_CAPACITY;
}
```

Mi kerüljön a /* 1 */ helyére?

Válaszok

- [x] `Files.lines(logFile, StandardCharsets.UTF_8).count()`
- [ ] `Files.walk(logFile).count()`
- [ ] `Files.size(logFile)`
- [x] `Files.readAllLines(logFile).size()`

Magyarázat

A Files.walk() egy könyvtárat jár be, nem fájlt nyit meg.

A Files.size() a fájl méretét adja vissza bájtokban.

A többi rendben van.

4. feladat 0/5 pont

A log fájlban az első sor a legrégebbi, az utolsó a legújabb bejegyzés. Ezért amikor elérjük a maximum megengedett sorok számát, az elsőt ki kell törölni. Eddig jutottunk:

```
void deleteFirstRow(Path file) throws IOException {
  try (/* 1 */) {
    StringBuilder stringBuilder = new StringBuilder();

    int lineNumber = 1;
    String line;

    while (/* 2 */) {
      if (lineNumber != 1) {
        stringBuilder.append(line).append("\n");
      }
      /* 3 */ ;
    }
    /* 4 */ ;
    fileWriter.write(stringBuilder.toString());
    br.close();
    fileWriter.close();
  }
}
```

Milyen sorrendben helyettesítenéd be a következő programrészeket, hogy a program működjön?

```
/* A */ lineNumber++
/* B */ FileWriter fileWriter = new FileWriter(file.toString())
/* C */ (line = br.readLine()) != null
/* D */ BufferedReader br = new BufferedReader(new FileReader(file.toFile()))
```

Válasz

- A B C D
- D A C B
- B C A D

- [x] DCAB

Magyarázat

Először a BufferedReader-t kell létrehozni a FileReader-rel (D).

Addig olvasunk a fájlból, ameddig még van benne adat (C).

Azután növeljük lineNumber-t, hogy a következő sort már hozzáfűztük a StringBuilder-hez (A),

Végül a fileWriter-t hozzuk létre (B).

5. feladat 0/5 pont

Adott egy fájl a következő tartalommal:

```
2021-07-16T11:01:15.710734800 Sok sikert
2021-07-16T11:01:20.902988500 minden kedves versenyzőnek!
```

Mi történik, ha meghívjuk rá a következő metódust?

```
void deleteFirstRow(Path file) throws IOException {
  String linesString = Files.readAllLines(file, StandardCharsets.UTF_8) // 1
      .stream()
      .skip(1)
      .collect(Collectors.joining("\n"));
  Files.writeString(file, linesString + "\n", StandardCharsets.US_ASCII); // 2
}
```

Válasz

- [ ] A program probléma nélkül lefut és letörli az első sort
- [ ] A program probléma nélkül lefut, letörli az első sort, de az "ő" karakter helyére "&0337" kerül
- [ ] A program java.nio.charset.UnmappableCharacterException kivételt dob a // 1 sornál
- [x] A program java.nio.charset.UnmappableCharacterException kivételt dob a // 2 sornál

Magyarázat

A program beolvassa gond nélkül a fájl tartalmát, de amikor a második sor tartalmát próbálja írni olyan karakterbe ütközik, ami nincs benne az US_ASCII karakterkészletben.

6. feladat 0/5 pont

Adott a LOGS_DIRECTORY-ban és annak közvetlen alkönyvtáraiban található néhány log.txt nevű fájl. Az alkönyvtáraknak nincsenek további alkönyvtárai. Szeretnénk megtalálni az összes log.txt fájlt. Eddig jutottunk a kóddal:

```
Path directory = Path.of(LOGS_DIRECTORY);
String searchedFileName = "oldLog.txt";
try (/* 1 */) {
  files.filter(file -> file.getFileName().toString().equals(searchedFileName))
       .forEach(System.out::println);
}
```

Válaszd ki az összes lehetőséget, ami az /* 1 */ helyére kerülhet!

Válaszok

- [ ] `Stream<File> files = Files.walk(directory, 1)`
- [x] `Stream<Path> files = Files.walk(directory)`
- [x] `Stream<Path> files = Files.walk(directory, 2, FileVisitOption.FOLLOW_LINKS)`
- [ ] `Stream<File> files = Files.walk(directory)`

Magyarázat

A Files.walk Stream<Path>-et ad vissza, nem Stream<File>-t.

A Files.walk opcionális maxDepth paraméterével állíthatjuk be milyen mélységig keressen a fájlrendszerben. Ha nincs felülírva, teljesen bejárja a struktúrát.

FileVisitOptions.FOLLOW_LINKS beállítást akkor végezzük el ha a symbolic linkeket követve is szeretnénk keresni, de ez számunkra most irreleváns.

7. feladat 0/5 pont

AsynchronousFileChannel legfőbb használati módja a következő:

Válasz

- [ ] Fájlokat tudunk küldeni aszinkron módon
- [x] Olvasni, írni, és változtatni tudunk fájlokat
- [ ] Eseményeket regisztrálhatunk amik jelelzhetik ha a fájl vagy mappa változott
- [ ] Másolhatunk, áthelyezhetünk fájlokat amik nagyobbak, mint 4GB

Magyarázat

Aszinkron fájl írásra és olvasásra használjuk.