Java progamozás

I. Nagy ZH - programozás Java nyelven

A zh írására 13:15 és 16:30 között nyílik lehetőség. Fontos, hogy időhosszabbításra nincs lehetőség.
Figyelem, 100 pont elérése számít 100%-os eredménynek!
A feladatokat figyelmesen olvasd el, tervezd meg a kódot és készítsd el. A programnak futnia kell és a kódnak a kivételeket kezelnie kell. A ZH írása során kizárólag a **csatolt hálózati meghajtón** található anyagok illetve. Meg nem engedett segédeszköz használata a ZH sikertelen teljesítését vonja maga után.

Egyetlen projektet készíts! Az egyes feladatoknak külön csomagja legyen: `hu.ppke.itk.java.DIGITUSAZONOSÍTÓ.zh1.feladatN`, ahol a DIGITUSAZONOSÍTÓ a digitus azonosítód, a feladatN-ből az N pedig a feladat sorszáma.

1. Dinamikus vektor osztály (40 pont)

Adott egy absztrakt vektor osztály (`Vector`), amely egyszerű dinamikus tömb funkciókat lát el plusz képes az elemeit rendezni. A rendezés implementációját a belőle örököltetett osztályokra bízza. A tömb csak olyan elemeket képes tárolni, amelyek a `VectorElement` osztályból származnak.

A feladatod a `Vector` osztályból örököltetett `VectorBubble` és `VectorSelection` osztályok `sort()` függvényének megvalósítása buborék és maximum kiválasztásos rendezéssel. Ezután hozz létre egy `Autó` és egy `Ember` osztályt amelyek a `VectorElement` osztályból öröklődnek. Az `Ember` 3 mezővel rendelkezzen: név, kor, magasság. Az autó mezői a típus, kor, ár és rendszám legyenek.

Definiáld felül a `greater()` függvényeket úgy, hogy az `Ember` osztályban életkorokat, az `Autó` osztályban árakat hasonlítson össze. Ez ahhoz szükséges, hogy a `Vector` osztályok rendezni tudjak az elemeket. A létrehozott osztályok teszteléséhez az emberek.txt és az autok.txt tartalmát töltsd be egy-egy vektorba, majd rendezés után írd ki őket a képernyőre.

A rendezések eredményét tetszőleges formátumban mentsd le a rendezett_emberek.txt és rendezett_autok.txt fájlokba is. Ne felejtsd el felüldefiniálni a két osztály `toString()` metódusát sem. Enélkül nem látszódna a rendezéseid eredménye.

2. Naplózó program (60 pont)

A logger objektumok feladata, hogy üzeneteket tároljanak egy rendszerről vagy alkalmazásról egy előre megadott csatornára. (Ez azért nagyon hasznos, mert ez alapján percre pontosan vissza lehet követni, hogy mi történt, ami például hiba esetén nagy segítség.)

Logger esetén az üzeneteket a feladatban három szintre osztjuk:

- debug: üzenetek szintje, amelyek hibakereséshez használhatók
- warn: üzenetek szintje, amelyek tájékoztatást adnak olyan eseményekről, melyek lehet, hogy hibák, vagy károsak lehetnek a program szempontjából
- error: üzenetek szintje, melyek tájékoztatást adnak bekövetkezett hibákról

Egy logger objektumnak van egy level tulajdonsága is, mely megadja, hogy jelenleg milyen szintű üzeneteket naplóz. Ha a logger debug level-en van, akkor kiírja a debug, warn, error üzeneteket egyaránt (logolja a megadott szintű üzeneteket és minden más üzenetet, melyek magasabb réteghez tartoznak). Ha warn level-en van, akkor már csak a warn és error szinthez tartozó üzeneteket logolja.

Példa egy logger működéséről:

2011.03.16. 7:53:18 +0100 logging.Main-DEBUG: A program elindult

2011.03.16. 7:53:18 -0200 logging.Main-WARNING: Nem találom a megadott fájlt.

Ez a logger debug szinten van. Ha magasabb szintre lenne állítva, a debug üzeneteket nem látnánk.

A feladat egy egyszerűsített MySimpleLogger osztály írása, mely a következő funkciókkal rendelkezik:

- Tud logolni konzolra vagy fájlba
- Megvalósítja a SimpleLogger interfészt, amit meg kell írnod, így rendelkezik a következő függvényekkel:
  - `debug(String msg)`: debug szintű üzenetet ír ki
  - `warning(String msg):` warning szintű üzenetet ír ki
  - `warning(String msg, Exception e)`: warning szintű üzenetet ír ki, a hozzá tartozó kivétellel együtt
  - `error(String msg)`: error szintű üzenetet ír ki
  - `error(String msg, Exception e)`: error szintű üzenetet ír ki, a hozzá tartozó kivétellel együtt
  - `setLevel(Level level)`: beállítja a logger level-jét, ahol Level egy enum típus.
  - `close()`: lezárja a loggert. Figyelni kell arra, hogy ezután már ne lehessen használni a Loggert.

A Logger a következő formátumot használja a kiírásra: (Időformátumhoz a feladat végén van egy kis segítség.)

```
[yyyy.MM.dd. HH:mm:ss Z] [A logger neve]-[LEVEL]: [üzenet]
```

Megjegyzések a megvalósításhoz:

- A logger egyparaméteres konstruktora (`String` típusú) létrehoz egy loggert a paraméterben megadott névvel, ami a konzolt használja kimenetként.
- A logger kétparaméteres (két `String` típus) konstruktora hozzon létre egy loggert, ami a második paraméterként megadott nevű fájlba logol, a név itt is az első paraméter.
- A logger háromparaméteres konstruktora (két `String` típusú és egy logikai típusú paraméter) segítségével tudjunk létrehozni olyan loggert ami a megadott nevű fájlba logol, ahol a harmadik paraméter az append módot befolyásolja. (Ez lesz a fájlmegnyitás `append` paraméterének értéke.)
- A default level az a DEBUG.
- A kivételek logolása ne csak a `getMessage()` függvény hívásával történjen, hanem jelenjen meg a teljes stacktrace a log üzenetben.
- Legyen a `MySimpleLogger` osztálynak egy `setDateFormat(String fmt)` függvénye, amelyben a felhasználó beállíthatja a használni kívánt dátumformátumot (a helyessége legyen ellenőrizve).
- Legyen a `MySimpleLogger` osztálynak egy `config()` függvénye, mely következő reguláris kifejezésre illeszkedő `String`-et vár: "`TIME.*NAME.*LEVEL.*MESSAGE`". Ezzel legyen a felhasználónak lehetősége beállítania a kiírás formátumát. Tehát például meg lehessen adni a következő Stringet: "TIME - NAME : LEVEL - MESSAGE" (A program a TIME helyére az időt helyettesíti, stb.)

Segítség az időpont formázott megjelenítéséhez

- Idő lekérdezése: `Date date = new Date();`
- Alapértelmezett dátumformázó létrehozása: `DateFormat dateFormat = new SimpleDateFormat();`
- Dátumformázó létrehozása formátumsztringből: `DateFormat dateFormat = new SimpleDateFormat(formátumsztring);`
- A formátumsztring szintaxisa megtalálható a Java API dokumentációban. (Fentebb van egy példa.)
- A formázó használata egy adott időpontra (`String`-et ad vissza): `dateFormat.format(date)`