3P-1 Házi feladat

Felügyeletre tervezés

FIGYELEM: A házi feladat megoldása előtt olvassa el a tárgy weblapján lévő HF tudnivalókat! A házi feladat leadása előtt nézze végig a HF tudnivalóknál szereplő ellenőrző listát!

Kerettörténet

Cégünk be akar törni az okos háztartási eszközök piacára, és a tévéket, hűtőszekrényeket „intelligens" funkciókkal akarja kiegészíteni. Az új szolgáltatások lelkét egy kis beépített webszerver adná, mely az eszközzel kapcsolatos adatok és grafikus felület elérését biztosítaná. Mivel az ilyen eszközök valamilyen egyszerű célfelülettel rendelkeznek csak, ezért fontos, hogy a beállítások kiolvasását és az esetleges beavatkozásokat távolról, szabványos felületen meg tudjuk tenni.

Az első prototípus elkészítésekor magára a Java nyelven megvalósított webszerverre koncentrálunk, ennek a távoli felügyelhetőségi lehetőségeit kell demonstrálnunk. A távfelügyelet megoldására a legalkalmasabb megközelítés az alkalmazás ***JMX* instrumentációval** történő ellátása.

A program használata

Az alkalmazás egy többszálú webszerver, mely elindítása után a 8080-as porton szolgál ki HTTP-kéréseket. A szerver az alkalmazást futtató JRE `/lib` könyvtárában elhelyezett `www-server.properties` fájl segítségével konfigurálható, melyben az alábbi jellemzők adhatók meg:

- *root*: a szerver gyökérkönyvtára (alapértelmezett: az aktuális munkakönyvtár)
- *log*: annak a fájlnak az elérési útja, ahova a szervernek naplóznia kell (alapértelmezett: stdout).
- *workers*: kiszolgáló szálak száma (alapértelmezett: 5)
- *timeout*: timeout intervallum milliszekundumban (alapértelmezett: 5000)

Amennyiben a konfigurációs fájl nem létezik, a fenti jellemzők alapértelmezett értékkel inicializálódnak.

A program az alábbi paranccsal indítható:

```
java -jar WebServer.jar
```

A szerverhez ezután egy böngészővel lehet csatlakozni a http://localhost:8080/ címen.

1. feladat: Ismerkedés az alkalmazással

Hogy jobban megértsük az alkalmazás működését, a forráskód tanulmányozásával készítsünk egy UML modellt (tipikusan komponens vagy osztálydiagramokkal), ami ábrázolja a rendszer főbb részeit, külső függőségeit és azok kapcsolatait. A modell tartalmazza, hogy melyik komponens milyen adatot tárol, hogyan kommunikálnak

egymással a komponenseink, tipikus esetben melyikből hány darab fut stb. A modell elemeit, és így az alkalmazás felépítését és működését, egy rövid szöveges leírásban is ismertessük.

Figyelem: nem egy reverse engineering eszközzel előállított részletes osztálydiagramot kérünk, hanem egy saját modellt, ami csak az alkalmazás fontosabb részeit ábrázolja!

2. feladat: Felügyeleti modell elkészítése

Az MBean-alapú instrumentációt előkészítendő és dokumentálandó, UML segítségével adjuk meg az alkalmazás felügyeleti modelljét! Jelen feladatban a felügyeleti modell minimálisan egy statikus struktúra diagram, mely megadja a felügyeletet lehetővé tevő MBean(ek) menedzsment-interfészét és az azokat megvalósító osztályok kapcsolatát az alkalmazáslogikáért felelős osztályokkal.

- A felügyeleti modellezés során figyeljünk arra, hogy a menedzsment-tevékenységeket alapvetően ne a funkcionalitást megvalósító osztályok végezzék! (Lásd: Rendszermonitorozás előadás.)
- Ebben a feladatban koncentráljunk a rendszer állapotinformációjára és beállításaira (azaz ne teljesítmény jellegű attribútumokat definiáljunk).
- Az egyes, külső felügyelet számára elérhetővé tett attribútumokat/metódusokat dokumentáljuk úgy, hogy abból a megvalósítást nem ismerő szereplők számára is egyértelmű legyen azok jelentése!

Röviden ismertessük, hogy a felügyelet megvalósításához milyen változtatások szükségesek a már adott osztályokon, illetve alkalmazás-logikán!

3. feladat: Felügyeleti működés megvalósítása JMX segítségével

Valósítsuk meg a fent megtervezett JMX alapú távfelügyeleti műveletek közül **ötöt**, és dokumentáljuk a megvalósításhoz kapcsolódó főbb tervezési döntéseket!

A dokumentáció tartalmazza a távoli felügyelethez szükséges indítási paraméterezést, valamint környezet-konfigurációt. A tesztelés dokumentálása során mutassuk be a megvalósítás távolról vezérlését (mindegyik megvalósított attribútumot vagy metódust külön-külön)!

További követelmények

- Az alkalmazás új publikus elemeihez kötelező Javadoc típusú kommenteket készíteni.
- A dokumentáció mellett a teljes forráskód, JAR állományként a lefordított kód és egy indítószkript is leadandó.
- A megoldás során lehet, hogy módosítani kell az eredeti forráskódot. A forrásfájlok új verzióján kívül a leadott csomagba rakjunk be *diff* fájlokat is (unified diff formátumban[1]), minden módosított fájlhoz egy-egy külön diff fájl tartozzon. Diff fájlokat Linux alatt a `diff -u eredeti_fajl modositott_fajl` paranccsal tudunk létrehozni, Windows alatt pl. a GnuWin[2] tartalmazza a diff parancsot.

[1] Wikipedia. Diff, Unified format, http://en.wikipedia.org/wiki/Diff#Unified_format
[2] GnuWin, http://gnuwin32.sourceforge.net/packages.html