Opdracht JSF32, week 17

In deze opdracht experimenteren we met:

* Java JNA 🡪 Gebruik de JNA versie op de portal.

**Opdracht 1**

Maak een Java project aan waarmee een Windows system call wordt aangeroepen om de systeemtijd op te vragen. De te ondernemen stappen zijn:

* Zoek een Windows system call om de systeemtijd op te vragen (je kan hiervoor de functie GetSystemTime gebruiken, maar een andere mag ook).
* Maak deze system call beschikbaar in Java door middel van JNA.
* Meet hiermee de verwerkingsduur van een bepaald langdurig statement, bv  
  for(int i=0;i<1000000000;i++);
* Meet hetzelfde statement ook met de Java methode System.nanoTime().
* Geef alle resultaten in milliseconden weer.
* Vergelijk de resultaten.

**getSystemTime laat meestal een tijd rond 300ms zien maar soms komt er een tijd uit rond de -600ms.**

**System.nanoTime() geeft altijd een cijfer weer dat rond de 290ms ligt.**

**Opdracht 2**

Bepaal nu de hoeveelheid vrije ruimte op disk C:.

Je kan hiervoor de Windows functie GetDiskFreeSpace gebruiken, maar houdt wel rekening met de volgende kanttekeningen:

* In deze functie heeft de eerste parameter het type LPCTSTR, hetgeen een C string voorstelt (met deze parameter geeft de root aan van de schijf waarvan we de vrije ruimte willen bepalen, dus in dit geval moet dat C:\ zijn).  
  In de Windows API wordt echter onderscheid gemaakt tussen strings met ASCII codes (1 byte per karakter) en strings met UniCodes (2 bytes per karakter). Het laatste soort strings wordt ook wel “wide string” genoemd.   
  ASCII strings worden in Java afgebeeld op het gewone type String, terwijl Wide strings worden afgebeeld op het type char[] (array van char’s).  
  Omdat er 2 soorten strings zijn, zijn er ook 2 varianten van system calls die strings gebruiken: de ene eindigt op A en gebruikt ASCII strings, de andere eindigt op W en gebruikt Wide strings.   
  In dit geval betekent dit dat we kunnen kiezen tussen GetDiskFreeSpaceA (in dat geval moet de eerste parameter in Java een String zijn) en GetDiskFreeSpaceW (in dat geval moet de eerste parameter in Java een char[] zijn).
* De andere parameters zijn van het type LPDWORD, hetgeen een pointer naar een int voorstelt. In Java vervangen we dat door het type IntByReference.
* De functie retourneert niet zonder meer de hoeveelheid vrije ruimte op de schijf, maar een aantal andere waardes. Met behulp van deze waardes kunnen we de hoeveelheid vrije ruimte wel berekenen.  
  Een complicatie is dat de hoeveelheid vrije ruimte op je schijf waarschijnlijk groter is dan wat we in een int kunnen opslaan (de maximale waarde in een int is ongeveer 2G).   
  We kunnen dit oplossen door de berekening met het type long te doen in plaats van int, of door de vrije ruimte in kBytes te geven in plaats van Bytes.

**Opdracht 3 (optioneel)**

Als opdracht 1, maar nu voor Linux.