Operativni Sistemi

Drugi domaći zadatak

Java FAT16 simulacija

# Opis zadatka

Napisati Java aplikaciju koja simulira rad FAT16 diska, uz grafički prikaz skladištenja podataka u simulirane klastere fajl sistema i sektora na disku.

Studentima je obezbeđen kostur aplikacije kroz sledeće:

Disk - interfejs koji opisuje disk uređaj sa sektorima fiksne veličine.

SimpleDisk - klasa koja implementira opisani interfejs.

FAT16 - interfejs koji opisuje FAT16 sistem datoteka.

FAT16Test - skup jediničnih testova za proveru implementacije FAT16 sistema datoteka.

Directory - interfejs koji opisuje direktorijumski servis za zapisivanje i čitanje datoteka.

AbstractDirectory - apstraktna klasa koja referencira FAT16 i Disk.

DirectoryTest - skup jediničnih testova za proveru implementacije direktorijuma.

Od studenata se očekuje da implementiraju FAT16 interfejs, kao i da naprave implementaciju Direktorijuma koja nasleđuje AbstractDirectory, tako da priloženi jedinični testovi prolaze sa tim implementacijama. Trenutno testovi koriste “mock” implementaciju za FAT16 i Directory, i padaju na svim proverama.

# Opis datih klasa i interfejsa

## Interfejs Disk i klasa SimpleDisk

Predstavlja implementaciju diska. Pri konstruisanju diska se navodi veličina za sektor na disku, kao i ukupan broj sektora.

Metode writeSector() i writeSectors() mogu da se koriste za pisanje sektora na disk.

Metode readSector() i readSectors() mogu da se koriste za čitanje sektora.

Sve metode za čitanje i pisanje rade sa nizovima bajtova kao podacima i int-ovima kao rednim brojevima sektora. Numerisanje sektora počinje od 0, i završava se sa sectorCount-1.

Metode getSectorSize(), getSectorCount() i diskSize() pružaju informacije o veličini sektora, ukupnom broju sektora, i ukupnoj veličini diska, respektivno.

Validna implementacija ovog interfejsa je takođe priložena, pod nazivom SimpleDisk, i nju treba koristiti pri izradi domaćeg zadatka.

## Interfejs FAT16

Predstavlja implementaciju FAT16 sistema datoteka. Prvi deo domaćeg zadatka se odnosi na implementaciju ovog interfejsa.

FAT16 je tabela dvobajtnih vrednosti - klastera, gde se svaki klaster zapisuje kao fiksan broj sektora na disku (veličinu klastera u sektorima ćemo nazivati širinom klastera). Pošto su vrednosti za klastere 0 i 1 rezervisane, numerisanje klastera ne počinje od 0, već od 2, završava se sa 0xFFEF (poslednjih nekoliko vrednosti su takođe rezervisane). Bitne rezervisane vrednosti su:

* 0 - slobodan klaster
* 0xFFF8 - kraj lanca

Za potrebe domaćeg je neophodno implementirati sledeće FAT16 operacije:

* Konstruktor za klasu koja implementira ovaj interfejs, koji uzima broj klastera u FAT tabeli i širinu klastera, u sektorima. Napomena: broj klastera u tabeli ne bi trebalo da bude parametar, već bi trebalo da bude konstanta kod FAT16. Maksimalna validna vrednost za klaster sa podacima je 0xFFEF, a numerisanje klastera počinje od 2, tako da je broj mogućih klastera kod FAT16 fiksan: 0xFFEF - 2 = 0xFFED = 65517. Ipak, za potrebe domaćeg zadatka, pretpostavićemo da ova vrednost nije konstanta, i dozvoliti korisniku sistema da ručno navede broj klastera u našem sistemu (dok god je taj broj manji od 0xFFED), kako bismo lakše testirali sistem. Pogledajte jedinične testove za FAT za primer testa koji se oslanja na malu FAT tabelu.
* int getEndOfChain() - vraća konstantu koja predstavlja vrednost u FAT tabeli koja naznačava kraj lanca. Za FAT16, ovo je 0xFFF8.
* int getClusterCount() - vraća broj klastera ove FAT tabele.
* int getClusterWidth() - vraća širinu klastera, u sektorima.
* int readCluster(int clusterID) - čita vrednost u tabeli za zadati redni broj klastera. Ovde je bitno napomenuti da se klasteri numerišu od 2 do (brojKlastera+2), a ne od 0 do brojKlastera. Ako korisnik zada redni broj koji nije validan, treba baciti FATException sa odgovarajućom porukom.
* int writeCluster(int clusterID, int valueToWrite) - upisuje vrednost u zadatu poziciju u tabeli. Opet, numeracija ide od 2 do (brojKlastera+2). Ako korisnik zada redni broj koji nije validan, treba baciti FATException sa odgovarajućom porukom.
* String getString() - vraća stringovnu prezentaciju tabele, koja je pogodna za proveru da li je stanje u tabeli validno. Za tabelu sa četiri stavke, gde je prva stavka 1, druga 2, treća 0 i četiri 0, treba vratiti: “[1|2|0|0]”

Primer korišćenja FAT tabele sa četiri klastera, širina klastera je jedan sektor:

@Test

public void simpleWriteReadTest() {

//4 clusters, each one sector width

FAT16 fat = new MockFAT(1, 4);

assertEquals(1, fat.getClusterWidth());

assertEquals(4, fat.getClusterCount());

assertEquals(0xFFF8, fat.getEndOfChain());

fat.writeCluster(2, 3);

fat.writeCluster(5, 4);

assertEquals(3, fat.readCluster(2));

assertEquals(4, fat.readCluster(5));

assertEquals("[3|0|0|4]", fat.getString());

}

## 

## Interfejs Directory

Ovaj interfejs opisuje direktorijum koji na zadatom disku zapisuje datoteke pomoću zadatog FAT. Nudi korisniku mogućnost da zapisuje nizove bajtova kao datoteke pod proizvoljnim imenom, da naknadno čita te datoteke, da briše datoteke, lista ih, kao i da prijavi korisniku spisak svih datoteka koje su trenutno upisane, veličinu diska i količinu slobodnog prostora na disku.

Najmanja adresibilna jedinica što se tiče direktorijuma je klaster, pošto direktorijum mora da pomoću FAT prati u kojim sektorima na disku je datoteka zapravo zapisana. Dakle, ako imamo disk sa sektorima veličine 512 bajtova, i FAT kod kojeg je širina klastera 2 sektora, datoteka će sigurno zauzeti bar 1024 bajta na disku, čak i ako je manja od toga.

Studentima je dozvoljeno da naprave internu strukturu / klasu, koja će predstavljati datoteku na disku.

Pošto je domaći zadatak simulacija pravog sistema datoteka, nema potrebe da se “zapisane” datoteke stvarno čuvaju na disku. Traži se da “zapisane” datoteke mogu da se pročitaju dok je Java program aktivan. Nakon što se Java program završi “zapisane” datoteke mogu da nestanu.

* boolean writeFile(String name, byte[] data) - zapisuje niz bajtova kao datoteku sa zadatim imenom. Vraća false ako nema dovoljno mesta na disku da se zapiše ova datoteka.
* byte[] readFile(String name) - čita datoteku sa zadatim imenom sa diska. Vraćeni niz bajtova treba da bude identičan kao što je bio u trenutku pisanja (uključujući dužinu niza). Ova metoda baca DirectoryException ako je zadato ime datoteke koja ne postoji.
* void deleteFile(String name) - briše datoteku sa zadatim imenom sa diska. Ova metoda baca DirectoryException ako je zadato ime datoteke koja ne postoji.
* String[] listFiles() - daje niz naziva imena datoteka koje se trenutno nalaze na disku.
* int getUsableTotalSpace() - vraća veličinu ukupnog prostora na disku koji može da se koristi, bez obzira na već zauzet prostor. Ova metoda treba da vrati manju od dve potencijalne vrednosti - fizičku veličinu diska, ili količinu prostora koji može da se adresia sa FAT-om koji koristimo. Veliki disk nam ništa ne znači ako FAT tabela ne može da ga pokrije, kao što velika FAT tabela ništa ne znači ako ne postoji fizički prostor da se upiše toliko podataka.
* int getUsableFreeSpace() - vraća količinu raspoloživog prostora na disku koji može da se koristi.

Primer korišćenja Directory interfejsa:

@Test

public void simpleWriteReadTest() {

//4 clusters, each one sector width

FAT16 fat = new MockFAT(1, 4);

//sectors are 40 bytes, 10 of them on disk

Disk disk = new SimpleDisk(40, 10);

Directory dir = new MockDirectory(fat, disk);

//50 bytes of data, should take up two clusters, which are two sectors

byte[] data = new byte[50];

for(int i = 0; i < 50; i++) {

data[i] = (byte)(i\*2);

}

//160 allocatable bytes - FAT is smaller than actual disk

assertEquals(160, dir.getUsableTotalSpace());

assertEquals(160, dir.getUsableFreeSpace());

if (dir.writeFile("Even", data)) {

byte[] readData = dir.readFile("Even");

assertEquals(50, readData.length);

for (int i = 0; i < 50; i++) {

assertEquals((byte)(i\*2), readData[i]);

}

} else {

fail("Could not write file");

}

assertEquals("[3|65528|0|0]", fat.getString());

assertEquals(160, dir.getUsableTotalSpace());

assertEquals(80, dir.getUsableFreeSpace());

}

## Jedinični testovi

Kratki Java programi koji su napisani tako da proveravaju validnost implementacije. Izvršavaju se “Run As -> JUnit Test”. Svaka assert\* linija može da izazove da test padne. Ako se vrednost u pozivu assert\* metode ne poklapa sa očekivanom, razvojno okruženje će prijaviti da je test pao, koja je bila očekivana vrednost i koja je vrednost koju je dala trenutna implementacija. Bacanje izuzetaka se proverava u testovima koji imaju dodatni “expects” atribut u okviru opisa metode za testiranje.

Isprva svi priloženi testovi padaju, zato što su implementacije za FAT16 i Directory prazne. Sa pravilnom implementacijom ovih interfejsa svi testovi treba da prolaze.

# Grafički prikaz (računarske nauke)

Pored implementacije direktorijuma nad FAT16 diskom, treba napraviti i GUI prikaz rada ovog sistema. GUI treba da sadrži:

* Prozor koji prikazuje sadržaj FAT16 tabele, gde je vrednost svakog klastera zapisana u zasebnom kvadratnom polju, uz mogućnost da se stavke koje se odnose na jednu zadatu datoteku markiraju posebnom bojom. Pretpostaviti da će FAT tabela imati dovoljno mali broj stavki, tako da ceo prikaz može da se smesti na jedan ekran.
* Prozor koji prikazuje sadržaj diska, gde je svaki sektor predstavljen zasebnim kvadratnim poljem, i obojen crno ako je zauzet, ili belo ako je slobodan. U slučaju da je sektor zauzet, klik mišem na njegovo polje treba da izvrši prikaz sadržaja tog sektora u zasebnoj labeli. Pretpostaviti da će disk imati dovoljno mali broj sektora, tako da ceo prikaz može da se smesti na jedan ekran.
* Nema potrebe implementirati dodatne kontrole koje bi omogućile ostale komande nad direktorijumom, kao što su upisivanje datoteke, čitanje datoteke, brisanje, itd. Dovoljno je da se napravi test gde se konstruiše direktorijum, u njega se smeste neke datoteke, i pokrene se prikaz ova dva prozora nad tim direktorijumom.

# Bodovanje

* Računarske nauke:
  + Implementacija zadatih interfejsa: 8 poena
    - FAT16 - 3 poena
    - AbstractDirectory - 5 poena
  + Grafički prikaz: 7 poena
* Računarsko inzenjerstvo:
  + Implementacija zadatih interfejsa: 15 poena
    - FAT16 - 5 poena
    - AbstractDirectory - 10 poena

Napomene:

* Domaći se radi isključivo individualno - studenti koji rade grupno će dobiti 0 poena

# Rok za predaju zadatka

Zadatak se predaje putem mail-a na [bmilojkovic@raf.rs](mailto:bmilojkovic@raf.rs) ili [mveniger@raf.rs](mailto:mveniger@raf.rs).

Subject mail-a mora da bude u obliku: “[OS] D2 ime\_prezime\_ind”

* Npr. “[OS] D2 branislav\_milojkovic\_3807”

Naziv arhive mora da bude u obliku: “os\_d2\_ime\_prezime\_ind.ext”

* Npr. “os\_d2\_branislav\_milojkovic\_3807.rar”

Rok za predaju je:

* Ponoć, utorak-sreda, 1-2 maj

Neće se pregledati zadaci **(tj. biće dodeljeno 0 poena)** ako:

* Subject mail-a nije po navedenom obliku.
* Naziv arhive nije po navedenom obliku.
* Predaja se desi nakon navedenog roka.

Odbrana domaćih zadataka je obavezna. Imaćemo jedan poseban termin van časova vežbi koji će služiti samo za odbranu, i koji ćemo dogovoriti i objaviti nakon isteka roka domaćeg zadatka.