HALİME NUR İZMİRLİ – 1206210091 – FİLE ORGANİZATİONS LAB UYGULAMA

İÇİNDEKİLER

SINIFLAR VE DOSYALAR

Programda TreeFile ve Node olmak üzere iki adet sınıf vardır. Verileri tutan iki adet dosya vardır. “veriler.txt” dosyası eklenecek verilerin yollarını içeren dosyadır. Veriler.txt dosyasındaki yollar ağaç şeklinde “veriler_sirali.txt” dosyasına alfabetik olarak yazdırılır.

veriler.txt dosyasının içeriği:

```
Media_Root
Desktop
Media_Root/Videos
Media_Root/Music/Pop/track2.mp3
Media_Root/Photos
Media_Root/Music/Pop/track1.mp3
Media_Root/Photos/Events/photo4.jpg
Media_Root/Music/Jazz/track4.mp3
Media_Root/Music/Jazz/track3.mp3
Media_Root/Photos/Vacation/photo1.jpg
Media_Root/Videos/Movies/movie2.mp4
Media_Root/Photos/Vacation/photo2.jpg
Media_Root/Photos/Events/photo3.jpg
Media_Root/Videos/Movies/movie1.mp4
Media_Root/Videos/Documantaries/doc2.mp4
Media_Root/Videos/Documantaries/doc1.mp4
Desktop/belgeler
```

Veriler_sirali.txt dosyası:

```
Media_Root
  Music
    Jazz
      track3.mp3
      track4.mp3
    Pop
      track1.mp3
      track2.mp3
  Photos
    Events
      photo3.jpg
      photo4.jpg
    Vacation
      photo1.jpg
      photo2.jpg
  Videos
    Documantaries
      doc1.mp4
      doc2.mp4
    Movies
      movie1.mp4
      movie2.mp4
Desktop
  belgeler
```

TreeFile Sınıfı

Main metodunu içeren sınıftır. 3 adet metod ve bir liste içerir.

```
static List<Node> roots = new ArrayList<Node>();
```

liste, oluşturulan kök düğümleri tutar.

```
public static Node singletion(String name)
```

singletion metodu yeni node oluşturma sırasında eğer hazırda bir node varsa bunu döndürür. Yoksa yeni node oluşturur ve yeni node’u döndürür.

```
static void readFile()
```

Ağaç olarak sıralanıp yazılmış “veriler_sirali.txt” dosyasını okur ve ekrana bastırır.

Node sınıfı

Ağacın düğümlerini ifade eden sınıftır. String tipinde name değişkeni, ve çocuk düğümleri tutmak için liste içerir. Constructure içerisinde atama yapılır. 8 adet fonksiyon ve 1 adet inner class içerir.

```
String name = null;
List<Node> children;
```

`static void addChildren(String childName, Node node)`

Parametre olarak verilen childName yolunu "/" karakterine göre böler ve string tipinde `path[]` dizisine kaydeder. Her alt düğümü kontrol eder. Eğer düğüm yoksa yeni düğüm ekler. (`current.children.add(child);`)

Yeni düğümü listeye ekledikten sonra, çocuk düğümleri, kendi arasında, isimlerine göre alfabetik olarak `Collections.sort(current.children, new ComparebyName());` komutu ile sıralar. Bu işlem için bir inner class oluşturdum:

```
static class ComparebyName implements Comparator<Node> {
	public int compare(Node n1, Node n2) {
	return n1.name.compareTo(n2.name);
	}
}
```

ComparebyName sınıfı iki düğümün isimlerini karşılaştırır.

`static void writeToFile(int level, Node node, BufferedWriter writer)`

Verilen kökteki bütün çocukları "veriler_sirali.txt" dosyasına `write(node.name.indent(level * 2))` komutu ile yazdırır. Eğer ağacın birden fazla kökü varsa (örneğin "Media_Root" ve "Desktop" gibi) main fonksiyonu içerisinde bir for döngüsü ile bu kökler döndürülmelidir. Bu fonksiyon sadece verilen kökün çocuklarını yazdırır. Main içerisindeki for döngüsü aşağıdadır:

```
BufferedWriter writer = new BufferedWriter(new FileWriter("veriler_sirali.txt"));
	for (Node n : roots) {
		Node.writeToFile(0, n, writer);
	}
```

Bu kod sayesinde `List<Node> roots` içerisinde tuttuğumuz bütün kökler yeni dosyaya yazdırılacak.

`public static void deleteNode(String path, Node node)`

Node silmek için kullanıcı silmek istediği düğümün tam yolunu girmelidir. Eğer bu yol doğruysa, eşleşen düğüm ve bütün çocuk düğümleri silinir. Eğer böyle bir düğüm bulunamazsa uyarı mesajı verilir. Kodun içerisine yorum satırları ile açıklamalar eklenmiştir.

`public static Boolean searchNode(String name, Node node)`

Kullanıcının girdiği dosya ismini arar. Eğer böyle bir dosya veya klasör bulunmuşsa true döndürür. Bütün kök düğümleri aratmak için main içerisinde for döngüsüne ihityaç vardır.

```
public static String findPath(Node node, String fileName, String currentPath)
```

İsmi verilen dosyanın veya klasörün tam yolunu bulur. Eğer kök düğümün ismi girilmişse kök düğümün ismini döndürür. Kök düğüm değilse özyineli(recursive) olarak çocukları arar. Her çocuk düğümüne geçişte araya '/' karakteri koyar. Yolu string olarak döndürür. Bu fonksiyon eğer searchNode sonucu true ise çağırılacağından "verilen isimde bir düğüm bulunamadı." Şeklinde bir uyarı bastırmaya ihtiyaç yoktur.

Main içerisinde searchNode ve findPath fonksiyonlarının kullanımı:

```
for (Node n : roots) {
	if (Node.searchNode(name, n)) {
	System.out.println("Node found. The path is: " + Node.findPath(n,name,""));
		found = true;
		break;
	}
}
	if (!found) {
		System.out.println("Node not found");
	}
```

Traversal Metodları

Main metodu içerisinde aşağıdaki gibi çağırılırlar:

```
System.out.println("Select traversal method:");
System.out.println("1. In-Order");
System.out.println("2. Pre-Order");
System.out.println("3. Post-Order");
	int traversalType = scanner.nextInt();
	scanner.nextLine();
		switch (traversalType) {
			case 1:
				System.out.println("In-order traversal:");
				Node.inOrderTraversal(root);
				break;
			case 2:
				System.out.println("Pre-order traversal:");
				Node.preOrderTraversal(root);
				break;
			case 3:
				System.out.println("Post-order traversal:");
				Node.postOrderTraversal(root);
				break;
			default:
				System.out.println("Invalid traversal type.");
				break;
			}
```

```
public static void inOrderTraversal(Node node)
```

Ağaç BST olmadığından inorder traversal mümkün değildir. Sağ ve sol terimleri bu ağaç için geçerli değildir.

```
public static void preOrderTraversal(Node node)
```

Önce kök düğümü ardından çöcok düğümleri bastırır.

```
public static void postOrderTraversal(Node node)
```

Önce çocuk düğümleri ardından kök düğümü bastırır.

MAİN METODU VE EKRAN ÇIKTILARI

Main metodu içerisindeki kodlar ve açıklamaları aşağıda verilmiştir.

“veriler.txt” dosyasından veri okuma

```
while ((line = reader.readLine()) != null) {
                    String[] path = line.split("/");
                    root = singletion(path[0]);

                    Node.addChildren(line, root);
                }
```

İlk olarak "veriler.txt" dosyasındaki yollar okunur ve ‘/’ karakterine göre bölünür. Okunan veriler tek tek kontrol edilir ve eğer böyle bir düğüm yoksa yenisi singletion metodu ile oluşturulur. Kök ve çocuk düğümler böylece oluşturulmuş olur.

“veriler_sirali.txt” dosyasına veri yazdırma

```
BufferedWriter writer = new BufferedWriter(new
FileWriter("veriler_sirali.txt"));
     for (Node n : roots) {
          Node.writeToFile(0, n, writer);
     }
```

writeToFile metodu ile yeni dosyaya veriler ağaç halinde yazdırılır.

Ağaç halindeki verileri ekrana bastırma

```
readFile();
```

Bu metod, "veriler_sirali.txt" dosyasından satır satır verileri okur ve ekrana bastırır.

Ekran çıktısı yandaki gibidir.

```
Media_Root
  Music
    Jazz
      track3.mp3
      track4.mp3
    Pop
      track1.mp3
      track2.mp3
  Photos
    Events
      photo3.jpg
      photo4.jpg
    Vacation
      photo1.jpg
      photo2.jpg
  Videos
    Documantaries
      doc1.mp4
      doc2.mp4
    Movies
      movie1.mp4
      movie2.mp4
Desktop
  belgeler
```

Silme

```
System.out.println("Enter the path to deleting file: ");
     String path = scanner.nextLine();
          for (Node root1 : roots) {
               Node.deleteNode(path, root1);
          }
```

deleteNode() metodu ile yolu girilen dosyayı siler. Ekran çıktısı aşağıdadır.

```
Enter the path to deleting file:
Media_Root/Music/Jazz/track3.mp3
Deleting is successful
```

Arama işlemi

```
System.out.println("Enter the name to searching: ");
      String name = scanner.nextLine();
      boolean found = false;
            for (Node root2 : roots) {
                  if (Node.searchNode(name, root2)) {
                        System.out.println("Node found. The path is: " +
Node.findPath(root2,name,"" ));
                        found = true;
                        break;
                  }
            }
                  if (!found) {
                        System.out.println("Node not found");
                  }
```

Eğer searchNode() metodu true değer döndürürse ismi girilen dosyanın findPath() metodu ile tam yolu bulunur ve ekrana bastırılır. Ekran çıktısı aşağıdadır.

```
Enter the name to searching:
photo2.jpg
Node found. The path is: Media_Root/Photos/Vacation/photo2.jpg
```

Gezinme işlemi

```
      System.out.println("Select traversal method:");
      System.out.println("1. In-Order");
      System.out.println("2. Pre-Order");
      System.out.println("3. Post-Order");
            int traversalType = scanner.nextInt();
            scanner.nextLine();
switch (traversalType) {
      case 1:
            System.out.println("In-order traversal:");
            Node.inOrderTraversal(root3);
            break;
      case 2:
            System.out.println("Pre-order traversal:");
            Node.preOrderTraversal(root3);
            break;
      case 3:
            System.out.println("Post-order traversal:");
            Node.postOrderTraversal(root3);
            break;
```

Traversal metodları çağırılarak işlemler gerçekleşir. Ekran çıktısı yandadır.

```
Select traversal method:
1. In-Order
2. Pre-Order
3. Post-Order
3
Post-order traversal:
track4.mp3
Jazz
track1.mp3
track2.mp3
Pop
Music
photo3.jpg
photo4.jpg
Events
photo1.jpg
photo2.jpg
Vacation
Photos
doc1.mp4
doc2.mp4
Documantaries
movie1.mp4
movie2.mp4
Movies
Videos
Media_Root
Post-order traversal:
belgeler
Desktop
```