CENG 215 Veri Yapıları Laboratuvarı
Ödev 1: Bağlı Liste İşlemleri

Şevket Umut ÇAKIR

3 Kasım 2023

Verilen çift bağlı liste sınıfını kullanan, iş ilanlarının listesini saklayan `IlanListesi` sınıfının metotlarının yazılması istenmektedir.

1 Ödev Senaryosu

Bir programcı, serbest çalışan(freelance) bilgisayar işlerinin yayınlandığı bir platform oluşturmak istemektedir ve bunun için Java dilini kullanmayı tercih etmiştir. İşlerin listesini saklamak için derste öğrendiği çift bağlı listeler için kullanılan `DoublyLinkedList` sınıfını kullanmıştır. İş ilanlarının iş tanımı, kategorisi, ücreti, işin süresi(gün) ve aktif özellikleri bulunmaktadır. İş ilanlarını saklamak için kullanılan `IsIlani` sınıfının yapısı Listing 1'de verilmiştir. İş ilanlarının özellikleri aşağıdaki gibi tanımlanmıştır:

- **isTanimi(String):** İşin tanımını belirtir. İçinde satır sonu karakterleri bulunabilir.
- **kategori(String):** İşin kategorisini belirtir. Birkaç kelimedir.
- **ucret(double):** İşin ücretini belirtir.
- **isSuresi(int):** İşin, çalışan tarafından ne kadar sürede bitirilmesi gerektiğini belirtir.
- **aktif(boolean):** İşin aktif olup olmadığını belirtir. `false` ise pasiftir.

`IsIlani` sınıfının özelliklerinin değiştirilmemesi istendiği için sadece getter metotlar bulunmaktadır.

İş ilanları gerçek hayatta var olan ve `https://www.kaggle.com/datasets/ahmedmyalo/upwork-freelance-jobs-60k` adresinde bulunan veri kümesinin ilk 300 satırı kullanılarak oluşturulmuştur.

Derste öğretilen `DoublyLinkedList` sınıfı ufak değişikliklerle Listing 2'de verilmiştir. Her ne kadar sınıf içinde silme metotları bulunsa da listedeki elemanları silmemeli ve sırasını değiştirmemelisiniz. Listenin yapısını değiştirmek testlerin başarısız olmasına neden olacaktır.

İş ilanlarının listesi otomatik olarak rastgele değerlerle oluşturulmaktadır ve sistemdeki her değerlendirmede istenilen 4 metot için aynı liste kullanılmaktadır. Listenin yapısının bir metotta bozulması diğer metot testlerinin de başarısız olmasına neden olacaktır.

`IlanListesi` sınıfında ilanların listesi `DoublyLinkedList` sınıfı kullanılarak saklanır ve sizlerin göremediği kısımda otomatik olarak rastgele sayıda oluşturulurlar. Sizden istenen, `IlanListesi` sınıfında içi dolu olan ve `ilanlar` değişkeninde bulunan listeyi kullanarak ilgili metotları yazmanızdır.

2 Yazılması İstenen Metotlar

Bu bölümde yazılması istenen metotlar hakkında bilgi verilmektedir.

2.1 enBuyukUcretliAktifIlan

```
public IsIlani enBuyukUcretliAktifIlan()
```

Pasif olan ilanların `aktif` özelliği `false` olmaktadır. Liste içinde bulunan aktif ilanların içinde `ucret` alanı en büyük olan ilan nesnesinin geri döndürülmesi istenmektedir.

2.2 isTanimindaSozcukGecenIlanSayisi

```
public int isTanimindaSozcukGecenIlanSayisi(String arananSozcuk)
```

İş tanımları metinsel bir veridir. Parametre olarak verilen sözcüğün(`arananSozcuk`) kaç ilanın iş tanımında geçtiğini bulam metodu yazmanız istenmektedir.

2.3 isSuresiEnAzKinciIsinSuresi

`public int isSuresiEnAzKinciIsinSuresi(int k)`

Oluşturulan işlerin süreleri eşsiz olacak şekilde ayarlanmıştır. İlan listesindeki işler sürelerine göre küçükten büyüğe sıralandığında en küçük k. sıradaki işin süresini veren metodu yazmanız istenmektedir.

2.4 ucretToplamiEnBuyukKategori

`public String ucretToplamiEnBuyukKategori()`

Bu metotta her bir kategorideki ilanların ücretleri toplanacaktır ve ücret toplamı en büyük olan kategorinin ismi geri döndürülecektir.

3 Ödev Açıklamaları

- Ödevler `https://bilmoodle.pau.edu.tr` adresindeki arayüzlerden cevaplanacaktır. Ödevinizdeki kod dosyasını kişisel bilgisayarınıza indirip, ödevi çözüp tekrar sisteme geri yükleyebilirsiniz fakat sınıf, dosya ve paket yapısının korunduğundan emin olmalısınız.
- Ödevler bireysel olarak cevaplanmalıdır, grup çalışması yapmak yasaktır
- Ödevleri yapay zeka sohbet robotlarına cevaplatmak yasaktır
- Ödevlerde kopya kontrolü yapılacaktır ve benzerlik oranı belirli bir yüzdenin üzerinde olan ödevler kopya olarak değerlendirilecektir. Değişken isimlerini değiştirmek, kodların sırasını değiştirmek anlaşılabilen yöntemlerdir, lütfen bu yola başvurmayınız.
- Ödevin son teslim tarihi **12.11.2023 saat 23:59**'dur. Belirtilen saatte sistem kapanacaktır, lütfen son ana kadar beklemeden ödevinizi gönderin.

4 Ödev Sınıfları

Listing 1: IsIlani.java

```java
/**
 * IsIlani sınıfı iş ilanlarının bilgisini saklamak için kullanılır
 */
public class IsIlani {
    // İş açıklaması
    private String isTanimi;
    public String getIsTanimi() {
        return isTanimi;
    }
    // İş kategorisi
    private String kategori;
    public String getKategori() {
        return kategori;
    }
    // İşin ücreti
    private double ucret;
    public double getUcret() {
        return ucret;
    }
    // İşin kaç gün süreceği
    private int isSuresi;
    public int getIsSuresi() {
        return isSuresi;
    }
    // İşin aktiflik durumu, false ise iş yayında değil
    private boolean aktif;
    public boolean isAktif() {
        return aktif;
```

```
    }
    // Constructor
    public IsIlani(String isTanimi, String kategori, double ucret, int isSuresi, boolean
    ↪  aktif) {
        this.isTanimi = isTanimi;
        this.kategori = kategori;
        this.ucret = ucret;
        this.isSuresi = isSuresi;
        this.aktif = aktif;
    }
    // Alttaki metotlar testlerin çalışması ve yazdırma işlemleri için kullanılmaktadır,
    // kullanmanız ya da öğrenmeniz zorunlu değildir.
    @Override
    public String toString() {
        return "IsIlani [isTanimi=" + isTanimi + ", kategori=" + kategori + ", ucret=" +
        ↪  ucret + ", isSuresi="
                + isSuresi + ", aktif=" + aktif + "]";
    }
    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + ((isTanimi == null) ? 0 : isTanimi.hashCode());
        result = prime * result + ((kategori == null) ? 0 : kategori.hashCode());
        long temp;
        temp = Double.doubleToLongBits(ucret);
        result = prime * result + (int) (temp ^ (temp >>> 32));
        result = prime * result + isSuresi;
        result = prime * result + (aktif ? 1231 : 1237);
        return result;
    }
    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        IsIlani other = (IsIlani) obj;
        if (isTanimi == null) {
            if (other.isTanimi != null)
                return false;
        } else if (!isTanimi.equals(other.isTanimi))
            return false;
        if (kategori == null) {
            if (other.kategori != null)
                return false;
        } else if (!kategori.equals(other.kategori))
            return false;
        if (Double.doubleToLongBits(ucret) != Double.doubleToLongBits(other.ucret))
            return false;
        if (isSuresi != other.isSuresi)
            return false;
        if (aktif != other.aktif)
            return false;
        return true;
    }
}
```

Listing 2: DoublyLinkedList.java

```java
import java.util.Iterator;

public class DoublyLinkedList<T> implements Iterable<T> {
    /**
     * Node sınıfı, inner
     * @param <T> Generic değer
     */
    public static class Node<T> {
        public T value;
        public Node<T> next;
        public Node<T> previous;

        public Node(T value, Node<T> next, Node<T> previous) {
            this.value = value;
            this.next = next;
            this.previous = previous;
        }
    }
    private Node<T> head;
    private Node<T> tail;
    private int size;

    public DoublyLinkedList() {
        size=0;
    }
    public int size(){return size;}
    public boolean isEmpty(){return size==0;}
    public void addFirst(T value){
        Node<T> node=new Node<>(value, head, null);
        if (head!=null)
            head.previous=node;
        head=node;
        if (tail==null)
            tail=node;
        size++;
    }
    public void addLast(T value){
        Node<T> node=new Node<>(value, null, tail);
        if (tail!=null)
            tail.next=node;
        tail=node;
        if(head==null)
            head=node;
        size++;
    }
    public T removeFirst(){
        Node<T> node=head;
        head=head.next;
        if(head!=null)
            head.previous=null;
        size--;
        return node.value;
    }
    public T removeLast(){
        Node<T> node=tail;
        tail=tail.previous;
        if(tail!=null)
```

```
            tail.next=null;
        size--;
        return node.value;
    }
    public void print(){
        Node<T> node=head;
        while(node!=null) {
            System.out.print(node.value + " ");
            node=node.next;
        }
        System.out.println();
    }
    public void reversePrint(){
        Node<T> node=tail;
        while(node!=null) {
            System.out.print(node.value + " ");
            node=node.previous;
        }
        System.out.println();
    }

    public Node<T> getHead() {
        return head;
    }

    public Node<T> getTail() {
        return tail;
    }

    @Override
    public Iterator<T> iterator() {
        //Interface gerçekleştiren nesne oluşturma
        Iterator<T> iterator=new Iterator<T>() {
            private Node<T> node=head;
            @Override
            public boolean hasNext() { //Sonraki değer var mı
                return node!=null;
            }
            @Override
            public T next() { //Sonraki değeri döndür
                T rval=node.value;
                node=node.next; //iterator'u ilerlet
                return rval;
            }
        };
        return iterator;
    }

    @Override
    public String toString() {
        if (size==0)
            return "";
        String r="";
        for(T val:this)
            r+=val+", ";
        r=r.substring(0,r.length()-2);
        return r;
    }

    public void setSize(int size) {//size değerini değiştirmek için...
```

```
        this.size = size;
    }
    // Alttaki metotlar testlerin çalışması ve yazdırma işlemleri için kullanılmaktadır,
    // kullanmanız ya da öğrenmeniz zorunlu değildir.
    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 17;
        result = prime * result + ((head == null) ? 0 : head.hashCode());
        result = prime * result + ((tail == null) ? 0 : tail.hashCode());
        result = prime * result + size;
        return result;
    }
    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        DoublyLinkedList<T> other = (DoublyLinkedList<T>) obj;
        if (size() != other.size())
            return false;
        Iterator<T> iterator = iterator();
        Iterator<T> otherIterator = other.iterator();
        while (iterator.hasNext()) {
            if(!iterator.next().equals(otherIterator.next()))
                return false;
        }
        return true;
    }

}
```

Listing 3: IlanListesi.java

```
// İlan listesi sınıfı. Sadece belirtilen satırdan sonra değişiklik yapınız
// İstisna olarak üste import ifadeleri eklenebilir.
public class IlanListesi {
    // İlan listesini tutan değişken, dışarda oluşturulup constructor ile atanır.
    private DoublyLinkedList<IsIlani> ilanlar;
    public DoublyLinkedList<IsIlani> getIlanlar() {
        return ilanlar;
    }
    public IlanListesi(DoublyLinkedList<IsIlani> ilanlar) {
        this.ilanlar = ilanlar;
    }
    // Bu satırdan itibaren değişiklik yapabilirsiniz, metod imzalarını değiştirmeyiniz.
    /**
     * Aktif ilanlar arasında ücreti en büyük ilanı döndüren metot
     * Çok düşük bir ihtimalle aynı ücrete sahip olan birden fazla ilan bulunabilir. Bu
↪   durumda testi tekrar
     * çalıştırın lütfen.
     *
     * @return Ücreti en büyük aktif ilan
     */
    public IsIlani enBuyukUcretliAktifIlan() {
```

```
21          return null;
22      }
23      /**
24       * İş tanımı içinde aranan sözcüğün kaç ilanda geçtiğini veren metot
25       *
26       * @param arananSozcuk Aranan sözcük
27       * @return Aranan sözcüğün geçtiği ilan sayısı
28       */
29      public int isTanimindaSozcukGecenIlanSayisi(String arananSozcuk) {
30          return -1;
31      }
32      /**
33       * İş süreleri küçükten büyüğe sıralandığında k. sırada olan işin süresini veren
    ↪  metot,
34       * başka bir deyişle k. en küçük iş süresi
35       * Örneğin iş süreleri 6,3,11,2,14,8 ve k=3 olsun, geri dönüş değeri 6 olacaktır.
36       *
37       * @param k En küçük kaçıncı değerin istediğini belirten parametre
38       * @return İş süreleri küçükten büyüğe sıralandığında k. sırada olan işin süresi
39       */
40      public int isSuresiEnAzKinciIsinSuresi(int k) {
41          return -1;
42      }
43
44      /**
45       * Her bir kategorideki ilanlardaki ücretler toplanarak, toplam ücreti en çok olan
    ↪  kategori ismi döndürülmelidir
46       * Problemin kolay çözümü için Map arayüzünü ve HashMap sınıfının kullanımını
    ↪  inceleyebilirsiniz
47       *
48       * @return İçindeki ilanların ücret toplamı en çok olan kategori
49       */
50      public String ucretToplamiEnBuyukKategori() {
51          return null;
52      }
53  }
```