به نام خداوند بخشنده و مهربان

توضیح پروژه درس ساختمان داده

استاد درس : جناب آقای دکتر دهقان تخت فولادی

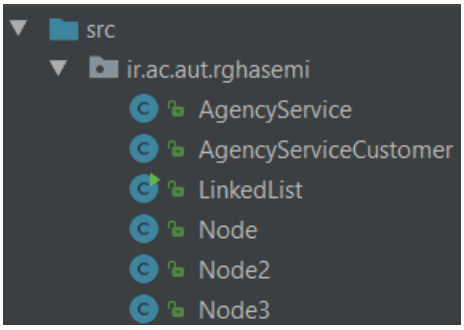
تدریس یاران : جناب آقای مهندس احمدپناه و جناب آقای مهندس اسدی

موضوع پروژه : سامان‌دهی نمایندگی‌های خودروسازی

دانشجو : روزبه قاسمی ۹۵۳۱۴۲۴

پاییز ۱۳۹۶

توضیح پروژه :



در این پروژه از ۶ کلاس مجزا استفاده شده که به شرح کامل مفصل و مجزا می‌پردازیم :

کلاس LinkedList (کلاس اصلی پروژه)

از آنجایی که استفاده کردن از کتابخانه های آماده مجاز نبود ، پس مجبور بودیم به صورت دستی LinkedList را پیاده کردیم . یعنی با استفاده از Node آن را ایجاد میکنیم.

```
/**
 * This is the constructor of this class.
 * Make own LinkedList with out use the library and make it with nodes.
 */
public LinkedList() {
    head = new Node( d: "0",  t: "m");
    listCount = 0;
}
```

نکته : پیچیدگی زمانی الگوریتم استفاده شده در LinkedList بالا برابر با n است.

این یک کانستراکتور یا سازنده کلاس LinkedList است که ابتدا شمارشگر list از صفر شروع شده و موقعیت head آن مشخص می‌شود.

اولین Method که استفاده می‌شود نقش نمایش دادن را دارد و به همین علت نام آن show است.

```
/**
 * @param current This node pass the current node that is using in this method.
 *                this method will show the data of the nodes.
 */
public void show(Node current) {

    while (current != null) {
        System.out.print(current.data + "    ");


        if (current.SubNode != null) {
            System.out.print(" (");
            show(current.SubNode);
            System.out.print(" )");

        }
        current = current.next;
    }

}
```

نکته : پیچیدگی زمانی الگوریتم استفاده شده در الگوریتم بالا برابر با n است.

دومین Method که استفاده می‌شود نقش اضافه کردن را دارد و به همین علت نام آن add است.

```
/**
 * @param d this parameter gives the String from main,it means the name of tasks.
 */
public boolean add(String d) {
    Node end = new Node(d, t: "m");
    Node current = head;

    while (current.next != null) {
        current = current.next;
    }
    current.next = end;
    listCount++;
    System.out.println(d + " appended to tail!");
    return true;
}

/**
```

نکته : پیچیدگی زمانی الگوریتم استفاده شده در الگوریتم بالا برابر با n است.

سومین Method که استفاده می‌شود نقش پیداکردن سرویس را دارد و به همین علت نام آن findservices است.

```
/**
 * @param d       this string takes the string of the name of Services and will search.
 * @param current takes the current node.
 * @return this part is return the subnode, if the next node was null so it return null.
 */
public Node findService(String d, Node current) {


    while (current.next != null && !current.data.equals(d)) {
        current = current.next;
    }
    if (current.next != null) {
        return current.SubNode;
    } else
        return null;
}
```

نکته : پیچیدگی زمانی الگوریتم استفاده شده در الگوریتم بالا برابر با n است.

چهارمین Method که استفاده می‌شود نقش اضافه کردن زیرسرویس را دارد و به همین علت نام آن addsubservice است.

```
/**
 * @param d        this string takes the string of the name of subservices.
 * @param d1       this string takes the string of the name of services.
 * @param current this parameter takes the Current node.
 * @return this parameter return boolean values.
 */
public boolean addSubService(String d, String d1, Node current) {


    while (current.next != null && !current.data.equals(d)) {
        current = current.next;
    }
    if (current.next != null) {
        Node end = new Node(d1, t: "s");
        if (current.SubNode != null) {
            current = current.SubNode;
            while (current.next != null) {
                current = current.next;
            }
            current.next = end;
        } else
            current.SubNode = end;
    }

    return true;
}
```

پنجمین Method که استفاده می‌شود نقش حذف کردن با آدرس دریافتی را دارد و به همین علت نام آن deleteNodeWithAddress است.

```
/**
 * @param d this parameter takes the node that would be delete.
 * @return this parameter return false is the node can't be delete or not found the Specified node.
 */
public boolean deleteNodeWithAddress(Node d) {
    Node current = head.next, y = head;
    while (current != null) {
        if (current == d) {

            listCount--;
            y.next = current.next;
            break;

        }
        y = current;
        current = current.next;
    }
    System.out.println("Delete Failed: No node found with given data!");
    return false;
}
```

نکته : پیچیدگی زمانی الگوریتم استفاده شده در الگوریتم بالا برابر با n است.

حال به Main میرسیم که بیشترین کارهارا انجام میدهد و برنامه از آن جا شروع می‌شود.

خود Main از چند بخش مهم تشکیل می‌شود که نیاز است آن ها را توضیح دهیم :

```
public static void main(String args[]) {

    LinkedList L = new LinkedList();
    LinkedList A = new LinkedList();
    AgencyService AS = new AgencyService();
    AgencyServiceCustomer ASC = new AgencyServiceCustomer();
    int[] MaxHeap = new int[100];
    int[] AgencyLevel = new int[100];
    String[] Customer = new String[100];
    String[] services = new String[100];
    int sizeMaxHeap = -1;
    Scanner scan = new Scanner(System.in);
```

در این قسمت از LinkedList که در ابتدا ذکر کردیم ۲ آبجکت مختلف،یکی برای لیست سرویس‌ها و زیرسرویس‌ها و دیگری برای نمایندگی های خدماتی است. سپس از دو کلاس مخصوص نمایندگی ها ۲ آبجکت

مختلف می سازیم. سپس برای قسمت سفارش و Agencylevel ها یک آرایه به اندازه ۱۰۰ تعریف می‌کنیم که بعدا استفاده آن‌ها را خواهیم دید.

برای مشتری ها و سرویس ها نیز دو آرایه مجزا از جنس String به اندازه ۱۰۰ در نظر می‌گیریم. همچنین سایز MaxHeap را برابر با 1- در نظر میگیریم.

حال برنامه وقتی اجرا می‌شود با استفاده از دستوراتی که از کاربر می‌گیرد هر کدام از دستور هایی که در ادامه

آمده است را انجام می‌دهد.

```
Scanner scan = new Scanner(System.in);
String a = "";
while (!a.contains("exit")) {
    a = scan.nextLine();
    String[] parts = a.split( regex: " ");
    if (a.contains("add service")) {
        L.add(parts[2]);
        //System.out.println(parts[2]);

    } else if (a.contains("add subservice")) {
        L.addSubService(parts[4], parts[2], L.head);
        //System.out.println(parts[2]);

    } else if (a.contains("list services from")) {
        L.show(L.findService(parts[3], L.head));

    } else if (a.contains("list services")) {
        L.show(L.head);

    } else if (a.contains("add agency")) {
        A.add(parts[2]);

    } else if (a.contains("list agencies")) {
        A.show(A.head);

    } else if (a.contains("add offer")) {
        Node x, y;
        x = L.findService(parts[2], L.head);
        y = A.findService(parts[4], A.head);
        AS.add(x, y);
```

با استفاده از دستور scan دستور را از کاربر میگیرد و با دستورهای موجود تطابق می‌دهد،در صورت وجود وارد آن شرط می‌شود.هر دستور مطابق آن چیزی است که در تعریف پروژه آمده است و با دستور دادن به آنها میتوان صحت آنها را چک کرد.

ادامه دستورات در پروژه موجود است و برای جلوگیری از طولانی شدن توضیح پروژه از آن صرف نظر می‌کنیم!

کلاس AgencyService (کلاس مخصوص سرویس‌های نمایندگی‌ها)

در این کلاس از یک سازنده استفاده شده که در دستور Main کلاس اصلی یعنی LinkedList صدا می‌شود.

```
public class AgencyService {

    public Node2 head;
    public int listCount;

    /**
     * This is the constructor of this class.
     */
    public AgencyService() {
        head = new Node2( a: null,  b: null);
        listCount = 0;
    }
```

در اینجا منظور از Node2 یک کلاس است که از آن آبجکت ساخته ایم و این کلاس مخصوص کلاس AgencyService هست.

```
    /**
     * The Node2 is specific for Agency Services.
     *
     * @param a this parameter use for the node that would be the position of nodes.
     * @param b this parameter use for the node that would be add after node a.
     */
    public boolean add(Node a, Node b) {
        Node2 end = new Node2(a, b);
        Node2 current = head;

        while (current.next != null) {
            current = current.next;
        }
        current.next = end;
        listCount++;

        return true;
    }
```

اولین Method این کلاس متود اضافه کردن است که نقش آن اضافه کردن نمایندگی است

دومین Method که میخواهیم به آن بپردازیم نقش جستجو و پیدا کردن موقعیت هر گره از یک نمایندگی را دارد.

```
/**
 * The Node2 is specific for Agency Services.
 * @param a this parameter use for the node that would be the position of nodes.
 * @param b this parameter use for the node that would be find after node a.
 */
public Node2 find(Node a, Node b) {

    Node2 current = head;

    while (current != null && current.service != a && current.agency != b) {
        current = current.next;
    }

    return current;
}
```

سومین و آخرین Method که در این کلاس هست نقش حذف کردن یک سرویس مشخص از یک نمایندگی مشخص را دارد.

```
/**
 * The Node2 is specific for Agency Services.
 * @param a this parameter use for the node that would be the position of nodes.
 * @param b this parameter use for the node that would be delete after node a.
 */
public Node2 delete(Node a, Node b) {

    Node2 current = head;

    while (current != null && current.service != a && current.agency != b) {
        current = current.next;
    }

    if (current != null) {
        current.next = current.next.next;
        listCount--;
    }
    current = head;
    while (current != null && current.service != a) {
        current = current.next;
    }


    return current;
}
}
```

کلاس AgencyServiceCustomer (کلاس مخصوص ارتباط نمایندگی و مشتری)

این کلاس یک سازنده یا کانستراکتور دارد و تنها از یک Method تشکیل شده است که نقش اضافه کردن سفارشات مشتری ها دارد.

```
public class AgencyServiceCustomer {
    public Node3 head;
    public int listCount;

    /**
     * This is the constructor of this class.
     */
    public AgencyServiceCustomer() {
        head = new Node3( ag: null,  cn: "0",  l: 0);
        listCount = 0;
    }
```

در اینجا مقصود از Node 3 یک کلاس مخصوص است که نقش Handle کردن گره های مخصوص سفارشات مشتری‌ها را دارد.

```
    /**
     * The Node3 is specific for Agency Service Customers.
     * @param ag the node of Agency names and details.
     * @param cn the string neme of Customer name.
     * @param l
     * @return
     */
    public Node3 add(Node2 ag, String cn, int l) {
        Node3 end = new Node3(ag, cn, l);
        Node3 current = head;

        while (current.next != null) {
            current = current.next;
        }
        current.next = end;
        listCount++;

        return end;
    }
}
```

نکته اینجاست که از کلاس Node2 که مخصوص Handle کردن نمایندگی هاست در این Method استفاده شده است.

کلاس Node (کلاس مخصوص Handle کردن سرویس‌ها و زیرسرویس‌ها)

```
public class Node {
    Node next = null;
    String data;
    Node SubNode = null;
    String type;

    public Node(String d, String t) {
        data = d;
        type = t;
    }
}
```

کلاس Node2 (کلاس مخصوص Handle کردن‌نمایندگی و سرویس‌های ارائه‌شده در آن نمایندگی)

```
public class Node2 {
    Node2 next = null;
    Node agency;
    Node service = null;

    /**
     * This is the constructor of this class.
     */
    public Node2(Node a, Node b) {
        agency = b;
        service = a;
    }

}
```

کلاس Node3 (کلاس مخصوص Handle کردن سفارشات هر نمایندگی)

```
public class Node3 {
    Node3 next = null;
    Node2 agencyService;
    String Customer_Name;
    int level;

    /**
     * This is the constructor of this class.
     */
    public Node3(Node2 ag, String cn, int l) {
        agencyService = ag;
        Customer_Name = cn;
        level = l;
    }

}
```

نحوه‌ی اجرای برنامه

برای اجرای پروژه کافی است همانند دستوراتی که در تعریف پروژه آمده ، زمانی که برنامه Run می‌شود به آن بدهیم.

از آنجایی که اجرای تمام دستورات باعث طولانی شدن توضیح پروژه می‌شود ، فقط چند دستور ابتدایی داده و خروجی آن هارا نشان می‌دهیم :

```
add service Cleaning
Cleaning appended to tail!
add subservice Car to Cleaning
add agency Tehran
Tehran appended to tail!
list agencies
0   Tehran
add offer Cleaning to Tehran
offer is added
order Cleaning to Tehran by Mr.karimi with 3
list orders Tehran
Mr.karimi 3 Cleaning
```

نکته مهم اینجاست که هزینه زمانی MaxHeap استفاده شده برابر با logn است.

پایان