بسمه تعالی

گروه ۲۴ : علیرضا قلی زاده – امیرمحمد حبیب زاده

موضوع پروژه : سیستم نوبت گیری بیماران در بیمارستان زبان: جاوا

در این پروژه ما نیاز به ا ستفاده از دو لیست هستیم. یکی برای دکتر ها که مقدار اول آن شامل نام دکتر و مقدار دوم آن شامل تخصص دکتر می شود. این لیست اطاعات دکتر ها را برای ما نگه میدارد. لیست دوم برای بیماران تولید و ا ستفاده میشود بطوری که نوبت های گرفته شده توسط بیماران در آن ذخیره می شوند. مقدار اول این لیست شامل نام بیمار و مقدار دوم آن شامل نام دکتر (که بیمار برای آن دکتر نوبت گرفته است) می باشد.

ساختار هر دو لیست ما شامل دو String می باشد پس از یک الگو استفاده میکنیم. و در کلاس LinkedList قالب هر عضو از آرایه را بصورت دو String مینویسیم.

ساختار یا الگو اعضای لیست:

```
public static class Pattern{
    String firstData;
    String secondData;
    Pattern link;
}

Pattern head = new Pattern();
Pattern member;
```

الگو یا node که ما در این برنامه با نام Pattern تعریف کردیم بصورت زیر تعریف شده است. همچنین اولین عضو لیست را هم head تعریف کردیم و عضوی بنام member هم ساختیم تا در برخی توابع به ما کمک کند. (این قسمت از کد در کلاس LinkedList نوشته شده است.)

توابع کلاس LinkedList:

```
public void addEnd(String data1, String data2){
    Pattern newMember = new Pattern();

    member = head;
    if (head.firstData == null){
        head.firstData = data1;
        head.secondData = data2;

    }else {
        while (member.link != null){
            member = member.link;
        }
        member.link = newMember;
        newMember.firstData = data1;
        newMember.secondData = data2;
    }
}
```

تابع add: این تابع اطلاعاتی که کاربر برای برنامه ارسال کند را دریافت و به لیست اضافه میکند. ما در این پروژه تنها به انتهای لیست عضو اضافه میکنیم. در این تابع ابتدا عضو جدیدی بنام newMember میسازیم. سپس مقدار head را در member ذخیره میکنیم تا در عملیات های تابع مقدار head تغییر نکند. سپس شرط خالی بودن لیست را بررسی میکنیم که در این صورت باید اطلاعات ورودی کاربر مستقیما بعنوان اولین عضو لیست قرار گیرد. در غیر اینصورت حلقه while را تا زمانی ادامه میدهیم که به عضو آخر لیست برسیم. در آن صورت

از حلقه خارج شده و link آخرین عضو لیست که خالی است را با عضو جدیدی که ابتدای لیست ساختیم جایگذاری میکنیم. و در پایان به عضو جدیدمان مقدار دهی میکنیم.

تابع edit: این تابع شامل ۴ آرگومان ورودی است که دو آرگومان به معنای اطلاعات فعلی عضو لیست است که قصد تغییر آنرا داریم و دو آرگومان دوم اطلاعات جدیدی که باید جایگزین اطلاعات قبلی شوند. در ابتدا مانند بقیه توابع مقدار head را در متغیری بنام member ذخیره میکنیم. سپس خالی نبودن تابع را بررسی میکنیم. اگر لیست خالی باشد چیزی برای ویرایش وجود ندارد. اگر لیست خالی نباشد وارد حلقه while میشویم که تا عضو یکی مانده به آخر جلو میرود. و در هر مرحله از حلقه این شرط را بررسی میکند که اگر مقدار اول و دوم ارسالی توسط کاربر، عضو کنونی لیست باشد با اطلاعات جدید جایگزین میشود. در انتهای حلقه هم یک عضو به جلو میرویم. در پایان حلقه یکبار دیگر شرط برابری اطلاعات ارسالی با عضو لیست را بررسی میکنیم زیرا حلقه تا عضو یکی مانده به آخر جلو میرود و عملا شرط پایانی تابع تنها برای عضو آخر تابع است.

```java
public void edit(String data1, String data2, String newData1, String newData2){
    member = head;
    if (head.firstData == null) {
        System.out.println("List is empty!");
        return;
    }

    while (member.link != null){
        if (member.firstData.equals(data1) && member.secondData.equals(data2)){
            member.firstData = newData1;
            member.secondData = newData2;
        }
        member = member.link;
    }
    if (member.firstData.equals(data1) && member.secondData.equals(data2)){
        member.firstData = newData1;
        member.secondData = newData2;
    }
}
```

تابع search: این تابع تنها یک آرگومان میگیرد که درواقع هر عبارتی است که در لیست موجود باشد. سپس متغیر های مورد نیاز مانند flag برای بررسی اینکه تابع جوابی پرینت کرده یا نه و همینطور counter برای ردیف کردن حواب های تابع میباشد. مانند توابع قبلی مقدار head را حفظ و سپس خالی نبودن لیست را بررسی میکنیم. اگر لیست خالی نبا شد وارد حلقه شده و شرط **وجود** هر رشته ای که شامل عبارت ار سالی کاربر با شد را برای مقدار اول و دوم عضو لیستمان بر رسی میکنیم. اگر وجود دا شت counter اضافه میشود و flag هم مقدار صحیح میگیرد که یعنی حداقل یکبار جوابی پرینت داده شده. بعد از شرط هم یک عضو به جلو حرکت میکنیم. شرطی که بعد از اتمام حلقه نیز آمده برای بررسی عضو آخر لیست است.

```java
public void search(String data1){
    boolean flag = false;
    int counter = 0;
    member = head;
    if (head.firstData == null){
        System.out.println("List is empty!");

    }else {
        while (member.link != null){
            if (member.firstData.contains(data1) || member.secondData.contains(data1)){
                counter++;
                System.out.println("\t" + counter + ". " + member.firstData + " (" + member.secondData + ")");
                flag = true;
            }

            member = member.link;
        }
        counter++;
        if (member.firstData.contains(data1) || member.secondData.contains(data1)){
            System.out.println("\t" + counter + ". " + member.firstData + " (" + member.secondData + ")");
            flag = true;
        }

        if (!flag){
            System.out.println("Not found!");
        }
    }
}
```

تابع delete: این تابع نیز دو آرگومان از کاربر میگیرد که همان اطلاعتی عضوی هستند که باید حذف شود. سپس عضو جدیدی میسازیم. مانند توابع قبل مقدار head را حفظ میکنیم و شرط خالی بودن تابع را بررسی می کنیم. سپس تو سط حلقه به عضو های بعدی میرویم. شرطی که در حلقه بررسی میشود این است که آیا عضوی که با آن وارد حلقه شدیم همان عضوی است که باید حذف شود. نکته مهم اینجا ست که در این شرط باید شرط دیگری نیز بیاوریم که این نکته را برر سی میکند که

```java
public void deleteByData(String data1, String data2){
    Pattern newMember = new Pattern();

    member = head;
    if (head.firstData == null) {
        System.out.println("List is empty!");
        return;
    }

    while (member.link != null){
        if (member.firstData.equals(data1) && member.secondData.equals(data2)){
            if (member == head){
                head = head.link;
            }else {
                member = member.link;
                newMember.link = member;
            }
            break;
        }
        newMember = member;
        member = member.link;
    }
    if (member.firstData.equals(data1) && member.secondData.equals(data2)){
        member.firstData = null;
        member.secondData = null;
        newMember.link = null;
    }

}
```

آیا عضو اول حذف میشود (که در این صورت head یک عنصر جلو میرود) یا اینکه عضوی غیر از اولی حذف میشود که ما باید اطلاعات عضو قبلی را در انتهای حلقه به newMember بدهیم تا اینجا عضو قبلی به عضو بعدی اشاره کند. مثلا اگر عضو سوم باید حذف شود ما اشاره گر عضو دوم را به عضو چهارم متصل میکنیم. شرطی که بعد از اتمام حلقه نیز آمده برای بررسی عنصر آخر لیست است که اشاره گر عنصر یکی مانده به آخر لیست را خالی میکند.

تابع print: این تابع ساده نیز در ابتدا counter برای ایجاد ردیف تولید میکند. سپس مقدار head را حفظ میکند و شرط خالی بودن لیست را بررسی میکند. با ورود به حلقه while اعضا را تک به تک پرینت میگیرد. آخرین خط پرینت نیز برای عضو آخر است که خارج از حلقه آمده.

```
public void printList(){
    int counter = 0;
    member = head;

    if (head.firstData == null) {
        System.out.println("\tList is Empty!");
        return;
    }

    while (member.link != null){
        counter++;
        System.out.println("\t" + counter + ". " + member.firstData + " (" + member.secondData + ")");
        member = member.link;
    }
    counter++;
    System.out.println("\t" + counter + ". " + member.firstData + " (" + member.secondData + ")");

}
```

تابع *print: این تابع نیز دقیقا مشابه تابع بالا ست با این تفاوت که برای پرینت شرطی گذاشته و آن مقداری است که کاربر وارد میکند. برای مثال در قسمت بیماران، بیمار تنها میتواند نوبت هایی که خودش گرفته را مشاهده کند.

```
public void printList(String data){
    int counter = 0;
    member = head;

    if (head.firstData == null) {
        System.out.println("\tList is Empty!");
        return;
    }

    while (member.link != null){
        if (member.firstData.equals(data)) {
            counter++;
            System.out.println("\t" + counter + ". " + member.firstData + " (" + member.secondData + ")");
            member = member.link;
        }
        member = member.link;
    }
    if (member.firstData.equals(data)) {
        counter++;
        System.out.println("\t" + counter + ". " + member.firstData + " (" + member.secondData + ")");
    }

}
```

»» برنامه بطور کل شامل دو لیست می باشد.

```
public static LinkedList doctorList = new LinkedList();
public static LinkedList patientList = new LinkedList();
```

منوی اولیه:

```
System.out.println("\tHospital Project ===>");
System.out.println("\t----------------------");
System.out.println("\t\t1. Special access");
System.out.println("\t\t2. Patient access");
System.out.println("\t\t3. Exit");
System.out.print("\tChoose one: ");
```

منوی بخش ویژه:

```
System.out.println("\tSpecial Access ===>");
System.out.println("\t----------------------");
System.out.println("\t\t1. Show Doctor List");
System.out.println("\t\t2. Add Doctor");
System.out.println("\t\t3. Edit Doctor Data");
System.out.println("\t\t4. Search for Doctor");
System.out.println("\t\t5. Delete Doctor");
System.out.println("\t\t6. Show all tickets");
System.out.println("\t\t7. Back");
System.out.print("\tChoose one: ");
```

منوی بخش بیماران:

```
System.out.println("\tPatient Access ===>");
System.out.println("\t----------------------");
System.out.println("\t\t1. Show Doctor List");
System.out.println("\t\t2. Get ticket");
System.out.println("\t\t3. Delete ticket");
System.out.println("\t\t4. Show my ticket");
System.out.println("\t\t5. back");
System.out.print("\tChoose one: ");
```