PROGRAMMING 2

Hospital system

المعيدة: ميساء العبودي

الطلاب:

حمزة محمد خلدون سيراجي

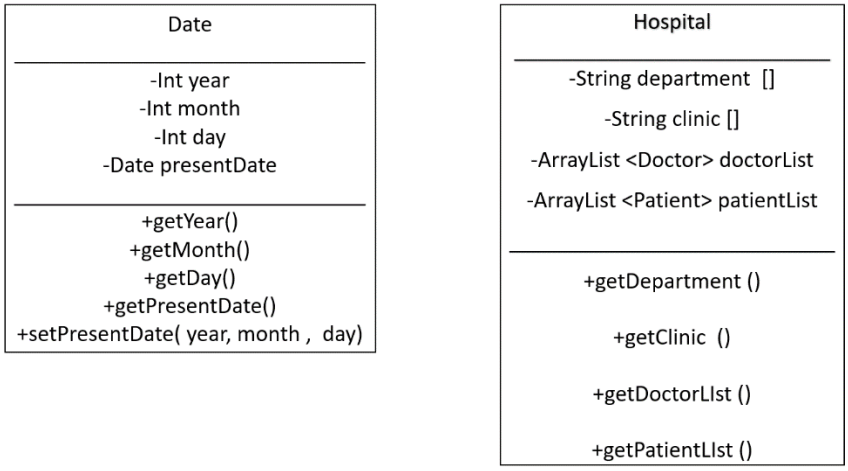
محمد نبيل بسام محمح

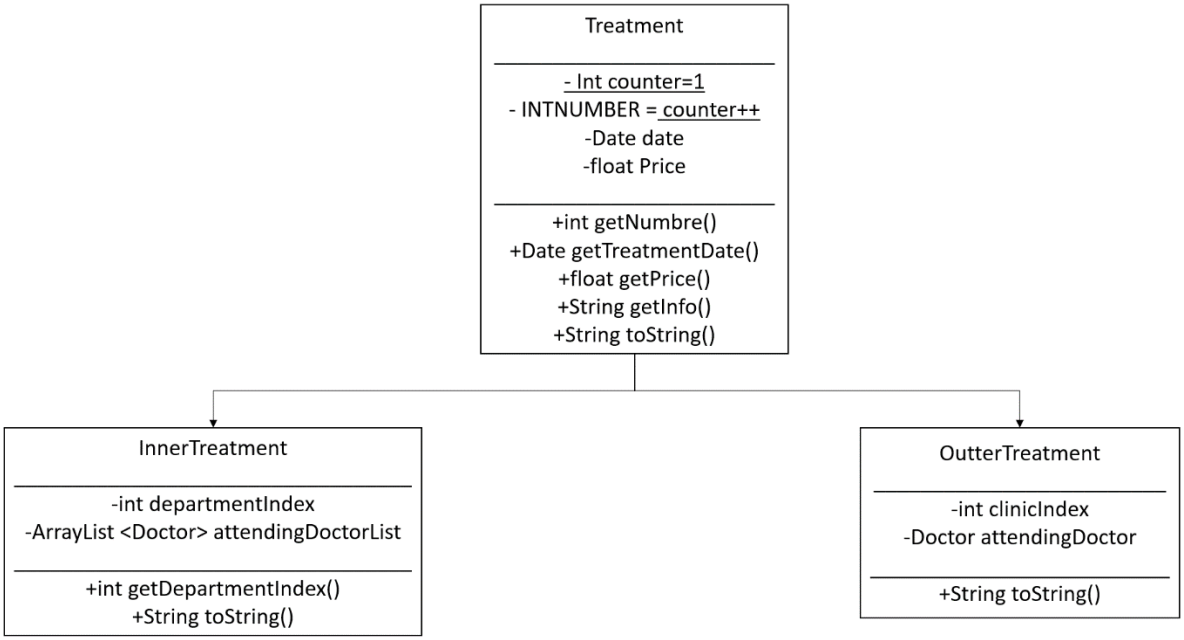
محمد ناصر المصطفى

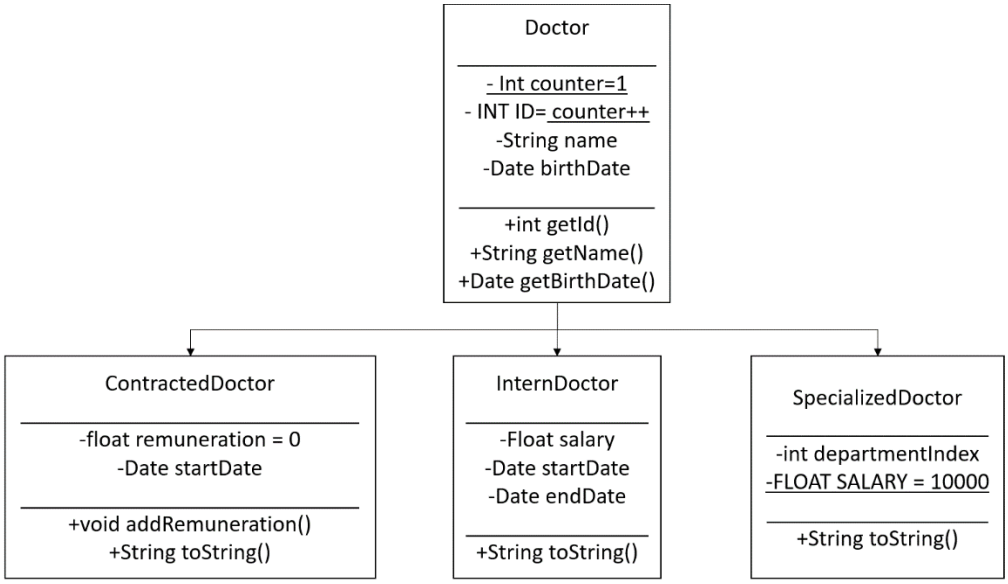
الدكتور:

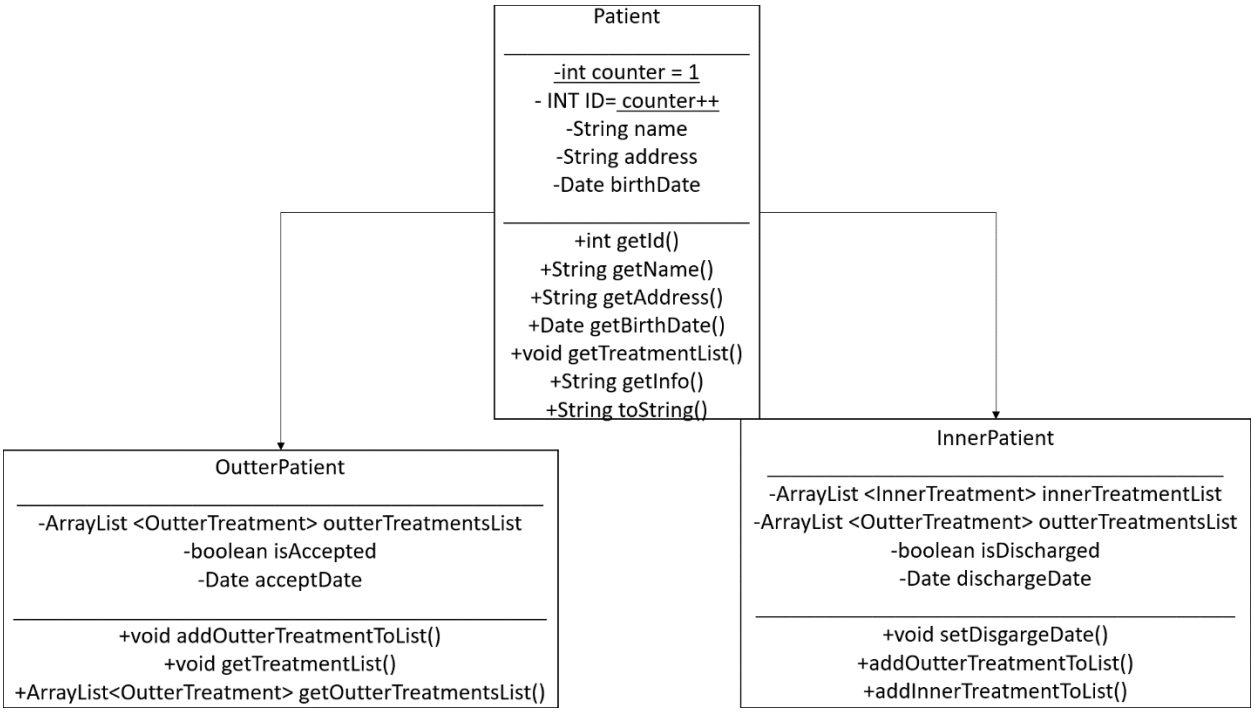
ماهر الصارم

مخطط الصفوف


Date
-Int year
-Int month
-Int day
-Date presentDate
+getYear()
+getMonth()
+getDay()
+getPresentDate()
+setPresentDate(year, month , day)
Hospital
-String department []
-String clinic []
-ArrayList <Doctor> doctorList
-ArrayList <Patient> patientList
+getDepartment ()
+getClinic ()
+getDoctorLIst ()
+getPatientLIst ()


Treatment
- Int counter=1
- INTNUMBER = counter++
-Date date
-float Price
+int getNumbre()
+Date getTreatmentDate()
+float getPrice()
+String getInfo()
+String toString()
InnerTreatment
-int departmentIndex
-ArrayList <Doctor> attendingDoctorList
+int getDepartmentIndex()
+String toString()
OutterTreatment
-int clinicIndex
-Doctor attendingDoctor
+String toString()


Doctor
- Int counter=1
- INT ID= counter++
-String name
-Date birthDate
+int getId()
+String getName()
+Date getBirthDate()
ContractedDoctor
-float remuneration = 0
-Date startDate
+void addRemuneration()
+String toString()
InternDoctor
-Float salary
-Date startDate
-Date endDate
+String toString()
SpecializedDoctor
-int departmentIndex
-FLOAT SALARY = 10000
+String toString()


Patient
-int counter = 1
- INT ID= counter++
-String name
-String address
-Date birthDate
+int getId()
+String getName()
+String getAddress()
+Date getBirthDate()
+void getTreatmentList()
+String getInfo()
+String toString()
OutterPatient
-ArrayList <OutterTreatment> outterTreatmentsList
-boolean isAccepted
-Date acceptDate
+void addOutterTreatmentToList()
+void getTreatmentList()
+ArrayList<OutterTreatment> getOutterTreatmentsList()
InnerPatient
-ArrayList <InnerTreatment> innerTreatmentList
-ArrayList <OutterTreatment> outterTreatmentsList
-boolean isDischarged
-Date dischargeDate
+void setDisgargeDate()
+addOutterTreatmentToList()
+addInnerTreatmentToList()

المكاتب التي تم استخدامها:

- java.util.ArrayList
- java.util.Scanner
- java.time.LocalDate

the classes:

Date:

```
    private int year , month , day ;
    public static Date presentDate ;
    public Date(int year, int month, int day) {
        this.year = year;
        this.month = month;
        this.day = day;
    }

    public int getYear() {
        return year;
    }

    public int getMonth() {
        return month;
    }

    public int getDay() {
        return day;
    }

    public static Date getPresentDate() {
        return presentDate;
    }

    public static void setPresentDate(int year, int month , int day) {
        Date.presentDate = new Date(year, month, day);
    }


    @Override
    public String toString() {
        return "Date{"+ year + "/" + month + "/" + day + '}';
    }
}
```

المتغيرات:
تم تعريف المتغيرات day, month, year من نوع int و متغير من نوع Date باسم presentDate

البواني:
باني بالقيم

التوابع:
مجموعة توابع get للمتغيرات و تابع set للمتغير presentDate و تابع toString للطباعة

Hospital:

```
5   public class Hospital {
        static final String department[] = {"NULL","ambulance department","opthalmology department","respiratory department"
        static final String clinic[] = {"NULL","internal medicine clinic","opthalmology clinic","gastroenterology clinic"};
8       static ArrayList <Doctor> doctorList = new ArrayList() ;
9       static ArrayList <Patient> patientList = new ArrayList() ;
10
11      public static void getDepartment() {
12          for (int i = 1; i < 5; i++) {
13              System.out.println("[" + i + "] " + Hospital.department[i]);
14          }
15      }
16
17      public static void getClinic() {
18          for (int i = 1; i < 4; i++) {
19              System.out.println("[" + i + "] " + Hospital.clinic[i]);
20          }
21        }
22
23      public static void getDoctorLIst() {
            for(Doctor e : doctorList)
25              System.out.println(e);
26          System.out.println("");
27      }
28
29      public static void getPatientLIst() {
            for(Patient e : patientList)
31              System.out.println(e);
32          System.out.println("");
33      }
34
```

المتغيرات:
تم تعريف المصفوفة department من نوع string و إعطائها قيم نهائية (أسماء الاقسام)

تم تعريف المصفوفة clinic من نوع string و إعطائها قيم نهائية (أسماء العيادات الخارجية)

تم تعريف قائمة من نوع Doctor باسم doctorList

تم تعريف قائمة من نوع Patient باسم patientList

البواني :
باني افتراضي

التوابع:
مجموعة توابع git للمصفوفات و قوائم

Doctor:

```java
public class Doctor {
    private static int counter = 1 ;
    private final int id = counter++;
    private String name ;
    private Date birthDate ;

    public Doctor(String name, Date birthDate) {
        this.name = name;
        this.birthDate = birthDate;
    }
    public Doctor(String name, int year , int month, int day) {
        this.name = name;
        birthDate = new Date(year, month, day) ;
    }

    public int getId() {
    return id ;
    }

    public String getName() {
        return name;
    }

    public Date getBirthDate() {
        return birthDate;
    }


    public String getInfo(){
    return "id=" + id + ", name=" + name + ", birthDate=" + birthDate ;
    }

    @Override
    public String toString() {
        return "Doctor{" + "id=" + id + ", name=" + name + ", birthDate=" + birthDate + '}';
    }


}
```

المتغيرات:

تم تعريف متغير static للمعمل كعداد

تم تعريف id من نوع int و ربطه مع العداد

تم تعريف name من نوعٍ String

تم تعريف birthdate من نوع Date

البواني:

تم تعريف بانيان بالقيم الأول يستقبل تاريخ الميلاد obj و الثاني كمتحولات

التوابع:

تم تعريف مجموعة من توابع git للمتغيرات بالإضافة لتابعي gitInfo toString للطباعة

تم توريث class Doctor لثلاثة أبناء:

- ContractedDoctor
- InternDoctor
- SpecializedDoctor

ContractedDoctor:

```java
public class ContractedDoctor extends Doctor {
    private float remuneration = 0 ;
    private Date startDate ;
    public ContractedDoctor(Date startDate, String name, Date birthDate) {
        super(name, birthDate);
        this.startDate = startDate;
    }

    public ContractedDoctor(int syear, int smonth, int sday, String name, int year, int month, int day) {
        super(name, year, month, day);
        this.startDate = new Date(syear, smonth, sday);
    }

    public void addRemuneration(float price) {
    remuneration += price / 2 ;
    }


    @Override
    public String toString() {
        return "ContractedDoctor{" + super.getInfo() + ", remuneration=" + remuneration + ", startDate=" + startDate + '}';
    }

}
```

المتغيرات:

تم تعريف remuneration من نوع float و إعطائه قيمة ابتدائية تساوي 0

تم تعريف startDate من نوع Date

البواني :

تم تعريف بانيان بالقيم الأول يستقبل تاريخ بدء التعاقد obj و الثاني كمتحولات

التوابع:

تم تعريف تابع لحساب الراتب بالإضافة لتابع toString للطباعة

InternDoctor:

```java
public class InternDoctor extends Doctor {
    private float salary ;
    private Date startDate ;
    private Date endDate ;

    public InternDoctor(int syear, int smonth, int sday, String name, int year, int month, int day) {
        super(name, year, month, day);
          salary = SpecializedDoctor.salary ;
        int servYear = Date.presentDate.getYear() - syear;

          if (servYear==0) {
            salary = salary / 2 ;
        }
        if (servYear==1) {
            salary = salary * 3 / 4 ;
        }
        this.startDate = new Date(syear, smonth, sday);
        endDate = new Date(2+startDate.getYear(),startDate.getMonth(),startDate.getDay()) ;
    }

    @Override
    public String toString() {
        return "InternDoctor{" + super.getInfo() + ", salary=" + salary + ", startDate=" + startDate + ", endDate=" + endDate
    }
```

المتغيرات:
تم تعريف salary من نوع float

تم تعريف startDate و endDate من نوع Date

البواني:

تم تعريف باني بالقيم يوم بداخله بانشاء endDate حسب قيم startDate

التوابع:
تابع toString للطباعة

SpecializedDoctor:

```
3   public class SpecializedDoctor extends Doctor {
4       private int departmentIndex ;
        final static float salary = 10000;
6
7       public SpecializedDoctor(int departmentIndex, String name, Date birthDate) {
8           super(name, birthDate);
9           this.departmentIndex = departmentIndex;
10      }
11
12      public SpecializedDoctor(int departmentIndex, String name, int year, int month, int day) {
13          super(name, year, month, day);
14          this.departmentIndex = departmentIndex;
15      }
16
17      public SpecializedDoctor(Doctor i , int departmentIndex) {
18      this(departmentIndex, i.getName() , i.getBirthDate());
19      }
20
21      @Override
        public String toString() {
23          return "SpecializedDoctor{"+ super.getInfo() + ", salary=" + salary + ", departmentIndex=" + Hospital.department[depar
24      }
```

المتغيرات:
تم تعريف departmentIndex من نوع int

تم تعريف salary موحدة و إعطائها قيمة نهائية تساوي 10000

البواني:
تم تعريف ثلاث بواني بالقيم الأول يستقبل تاريخ الميلاد obj و الثاني كمتحولات و الثالث يستقبل obj من نوع Doctor

التوابع:
تابع toString للطباعة

Patient:

```java
public class Patient {
    private static int counter = 1 ;
    private final int id = counter++;
    private String name ;
    private String address ;
    private Date birthDate ;

    public Patient(String name, String address, int year, int month, int day) {
        this.name = name;
        this.address = address;
        this.birthDate = new Date(year, month, day);
    }

    public int getId() {
        return id;
    }

    public String getName() {
        return name;
    }

    public String getAddress() {
        return address;
    }

    public Date getBirthDate() {
        return birthDate;
    }

    public void getTreatmentList() {
    }

    public String getInfo() {
        return "id=" + id + ", name=" + name + ", address=" + address + ", birthDate=" + birthDate + '}';
    }

    @Override
    public String toString() {
        return "Patient{" + "id=" + id + ", name=" + name + ", address=" + address + ", birthDate=" + birthDate + '}';
    }
```

المتغيرات :

تم تعريف متغير static للمعمل كعداد

تم تعريف id من نوع int و ربطه مع العداد

تم تعريف name من نوعٍ String

تم تعريف address من نوع String

تم تعريف birthdate من نوع Date

البواني :

تم تعريف باني بالقيم يستقبل birthdate كمتغيرات

التوابع :

تم تعريف مجموعة توابع git بالإضافة الى تابعي getInfo و toString للطباعة

نلاحظ وجود تابع getTreatmentList فارغ تم اجراء عليه عملية override بالابناء

تم توريث class Patient الى ابنين:

- InnerPatient
- OutterPatient

InnerPatient:

```java
public class InnerPatient extends Patient {
    private ArrayList <InnerTreatment> innerTreatmentList = new ArrayList();
    private ArrayList <OutterTreatment> outterTreatmentsList = new ArrayList();
    private boolean isDischarged ;
    private Date dischargeDate ;


    public InnerPatient(boolean isDischarged, int dyear, int dmonth, int dday, String name, String address, int year, int montl
        super(name, address, year, month, day);
        this.isDischarged = isDischarged;
        if (isDischarged) {
        this.dischargeDate = new Date(dyear, dmonth, dday);
        }
    }

    public InnerPatient(boolean isDischarged, String name, String address, int year, int month, int day) {
        super(name, address, year, month, day);
        this.isDischarged = isDischarged;
    }

    public InnerPatient(OutterPatient p) {
        super(p.getName(), p.getAddress(), p.getBirthDate().getYear(), p.getBirthDate().getMonth(), p.getBirthDate().getDay())
        this.outterTreatmentsList = p.getOutterTreatmentsList() ;
    }
    public void setDisgargeDate(int dyear, int dmonth, int dday){
    this.isDischarged = true ;
    this.dischargeDate = new Date(dyear, dmonth, dday) ;
    }

    public void addOutterTreatmentToList(int clinicIndex, Doctor attendingDoctor, int year, int month, int day, float Price) {
        outterTreatmentsList.add(new OutterTreatment(clinicIndex, attendingDoctor, year, month, day, Price)) ;
    }

    public void addInnerTreatmentToList(int departmentIndex, ArrayList attendingDoctorList , int year, int month, int day, floa
        innerTreatmentList.add(new InnerTreatment(departmentIndex, attendingDoctorList , year, month, day, Price)) ;
    }

    public boolean getIsDischarged() {
        return isDischarged;
    }

    @Override
    public void getTreatmentList(){
        for (InnerTreatment it : innerTreatmentList) {
            System.out.println(it);
        }
        for (OutterTreatment ot : outterTreatmentsList) {
            System.out.println(ot);
        }
        System.out.println("");
    }

    public ArrayList<InnerTreatment> getInnerTreatmentList() {
        return innerTreatmentList;
    }




    @Override
    public String toString() {
        return "InnerPatient{" + super.getInfo() + ", isDischarged=" + isDischarged + ", dischargeDate=" + dischargeDate + '}'
    }
}

```

المتغيرات :

تم تعريف قائمة من نوع InnerTreatment باسم innerTreatmentList

تم تعريف قائمة من نوعOutterTreatment باسم outterTreatmentsList

تم تعريف isDischarged من نوع Boolean

تم تعريف dischargeDate من نوع Date

البواني :

تم تعريف ثلاث بواني الاول اذا كان isDischarged بقيمة true فانه يطلب تاريخ dischargeDate كمتغيرات

اما الثاني فلا يطلب تاريخ dischargeDate

اما الثالث فيطلب فقط متحول من نوع OutterPatient

التوابع:

تابع setDisgargeDate لتخريج المريض

تابع addOutterTreatmentToList لاضافة معالجة على سجل المعالجة الخارجية

تابع addInnerTreatmentToList لاضافة معالجة على سجل المعالجة الداخلية

تابع getIsDischarged

تابع getTreatmentList

تابع getInnerTreatmentList

تابع toString للطباعة

OutterPatient:

```java
public class OutterPatient extends Patient {
    private ArrayList <OutterTreatment> outterTreatmentsList = new ArrayList();
    private  boolean isAccepted  ;
    private Date acceptDate ;

    public OutterPatient( int ayear, int amonth, int aday, String name, String address, int year, int month, int day) {
        super(name, address, year, month, day);
        this.acceptDate = new Date(ayear, amonth, aday);
    }

    public void addOutterTreatmentToList(int clinicIndex, Doctor attendingDoctor, int year, int month, int day, float Price) {
        outterTreatmentsList.add(new OutterTreatment(clinicIndex, attendingDoctor, year, month, day, Price)) ;
    }


    public void getTreatmentList(){
        for (OutterTreatment ot : outterTreatmentsList) {
            System.out.println(ot);
            System.out.println("");
        }
    }

    public ArrayList<OutterTreatment> getOutterTreatmentsList() {
        return outterTreatmentsList;
    }


    @Override
    public String toString() {
        return "OutterPatient{" + super.getInfo()  + ", isAccepted=" + isAccepted + ", acceptDate=" + acceptDate + '}';
    }
```

المتغيرات :
تم تعريف قائمة من نوع OutterTreatment باسم outterTreatmentsList

تم تعريف isAccepted من نوع Boolean

تم تعريف acceptDate من نوع Date

البواني :
تم تعريف باني بالقيم يستقبل acceptDate كمتغيرات

التوابع :
تم تعريف addOutterTreatmentToList لاضافة معالجة لسجل المعالجات الخارجية

تم تعريف getTreatmentList

تم تعريف getOutterTreatmentsList

تم تعريف toString للطباعة

Treatment:

```java
public class Treatment {
    private static int countre = 1 ;
    private  final int numbre = countre++ ;
    private Date date ;
    private float Price ;

    public Treatment(int year, int month, int day, float Price) {
        this.date = new Date(year, month, day);
        this.Price = Price;
    }

    public int getNumbre() {
        return numbre;
    }

    public Date getTreatmentDate() {
        return date;
    }

    public float getPrice() {
        return Price;
    }
    public String getInfo() {
        return "numbre=" + numbre + ", date=" + date + ", Price=" + Price + '}';
    }

    @Override
    public String toString() {
        return "Treatment{" + "numbre=" + numbre + ", date=" + date + ", Price=" + Price + '}';
    }

}
```

المتغيرات :
تم تعريف متغير static للمعمل كعداد

تم تعريف number من نوع int و ربطه مع العداد

تم تعريف date من نوع Date

تم تعريف Price من نوع float

البواني :
تم تعريف باني بالقيم

التوابع :
مجموعة توابع git بالإضافة الى تابعي gitInfo و toString للطباعة

تم توريث class Treatment الى ابنين :

- InnerTreatment
- OutterTreatment

InnerTreatment:

```java
public class InnerTreatment extends Treatment {
    private int departmentIndex ;
    private ArrayList <Doctor> attendingDoctorList = new ArrayList() ;



    public InnerTreatment(int departmentIndex, ArrayList attendingDoctorList, int year, int month, int day, float Price) {
        super(year, month, day, Price);
        this.departmentIndex = departmentIndex;
        this.attendingDoctorList.addAll(attendingDoctorList) ;
    }

    public int getDepartmentIndex() {
        return departmentIndex;
    }

    @Override
    public String toString() {
        return "InnerTreatment{" + super.getInfo() + ", departmentIndex=" + departmentIndex + ", attendingDoctorList=" + atten
    }

```

المتغيرات:
تم تعريف departmentIndex من نوع int

تم تعريف قائمة من نوع Doctor باسم attendingDoctorList

البواني :
باني بالقيم

التوابع :
تم تعريف getDepartmentIndex بالإضافة الى تابع toString للطباعة

OutterTreatment:

```java
public class OutterTreatment extends Treatment {
    private int clinicIndex ;
    private Doctor attendingDoctor ;



    public OutterTreatment(int clinicIndex, Doctor attendingDoctor, int year, int month, int day, float Price) {
        super(year, month, day, Price);
        this.clinicIndex = clinicIndex;
        this.attendingDoctor = attendingDoctor;
    }

    @Override
    public String toString() {
        return "OutterTreatment{" + super.getInfo() + ", clinicIndex=" + clinicIndex + ", attendingDoctor=" + attendingDoctor
    }

```

المتغيرات :

تم تعريف clinicIndex من نوع int

تم تعريف attendingDoctor من نوع Doctor

البواني :

باني بالقيم

التوابع :

تم تعريف تابع toString للطباعة

Main

في البداية سيتم طباعة الواجهة الرئيسية وإضافة ثلاث اطباء ومريضان من أنواع مختلفة ومن ثم الدخول الى حلقة while تحوي switch

Default:

سيقوم بطباعة invalid input

Case 0:

تقوم بكسر حلقة while وانهاء البرنامج

Case 1:

تقوم بالدخول الى switch أخرى ويطلب منك ادخال نوع الطبيب لادخاله

Default:

سيقوم بطباعة invalid input

Case 1:

سيطلب معلومات الطبيب المتدرب وعند إدخالها سيتم انشاء طبيب متدرب واضافته الى قائمة الأطباء الموجودة في hospital

Case 2:

سيطلب معلومات الطبيب المتخصص وعند إدخالها سيتم انشاء طبيب متخصص واضافته الى قائمة الأطباء الموجودة في hospital

Case 3:

سيطلب معلومات الطبيب المتعاقد وعند إدخالها سيتم انشاء طبيب متعاقد واضافته الى قائمة الأطباء الموجودة في hospital

Case 2:

سيعرض قائمة الأطباء الموجودين

سيطلب تحديد الطبيب المراد حذفه عن طريق ادخال id الخاص بالطبيب

سيتم البحث عم الطبيب الذي يحمل نفس id ويتم ازالته من قائمة الأطباء الموجودة في hospital

Case 3:

تقوم بالدخول الى switch أخرى ويطلب منك ادخال نوع الأطباء المراد عرضهم

Default:

سيقوم بطباعة invalid input

Case 1:

سيقوم بطباعة كل الأطباء الموجودة في hospital وفي حال عدم وجود اطباء سيقوم بطباعة there is no doctor

Case2:

سيقوم بطباعة كل الأطباء المتدربين الموجودين في hospital وفي حال عدم وجود اطباء سيقوم بطباعة there is no intern doctor

Case3:

سيقوم بطباعة كل الأطباء المختصين الموجودين في hospital وفي حال عدم وجود اطباء سيقوم بطباعة there is no specialized doctor

Case 4:

سيقوم بطباعة كل الأطباء المتعاقدين الموجودين في hospital وفي حال عدم وجود اطباء سيقوم بطباعة there is no contracted doctor

Case 4:

سيقوم بطباعة كل الأطباء المتدربين الموجودين في hospital

يطلب تحديد الطبيب المراد بإدخال id الخاص به

يطلب تحديد القسم المراد التخصص به وبعدها يتم تغيير نوع الطبيب من متدرب الى متخصص

في حال ادخال id خاطئ سيقوم طباعة invalid id

في حال عدم وجود اطباء سيقوم بطباعة there is no doctor

Case 5:

Default:

سيقوم بطباعة invalid input

Case 1:

سيقوم بطباعة عدد كل الأطباء الموجودة في hospital

Case2:

سيقوم بطباعة عدد كل الأطباء المتدربين الموجودين في hospital

Case3:

سيقوم بطباعة عدد كل الأطباء المختصين الموجودين في hospital

Case 4:

سيقوم بطباعة عدد كل الأطباء المتعاقدين الموجودين في hospital

Case 6:

سيطلب اختيار نوع المريض داخلي كان ام خارجي

Default:

سيقوم بطباعة invalid input

Case 1:

سيطلب معلومات المريض الداخلي

هل هو متخرج ام لا

عند إدخال المعلومات المطلوبة سيتم انشاء مريض داخلي واضافته الى قائمة المرضى الداخليين في hospital

Case 2:

سيطلب معلومات المريض الداخلي

عند إدخال المعلومات المطلوبة سيتم انشاء مريض داخلي واضافته الى قائمة المرضى الداخليين في hospital

Case 7:

سيقوم بطباعة كل المرضى الداخليين الغير مخرجين الموجودين في hospital

يطلب تحديد المريض بإدخال id الخاص به

سيطلب تاريخ التخريج

في حال ادخال id خاطئ سيقوم طباعة invalid id

في حال عدم وجود مرضى داخليين غير مخرجين سيقوم بطباعة there is no patient to discharge

Case 8:

سيقوم بطباعة كل المرضى الخارجيين الموجودين في hospital

سيطلب تحديد المريض الخارجي بإدخال id الخاص به

سيطلب تاريخ القبول وبعدها يتحول المريض الخارجي الى داخلي

في حال ادخال id خاطئ سيقوم طباعة invalid id

في حال عدم وجود مرضى خارجيين سيقوم بطباعة there is no outter patient

Case 9:

تقوم بالدخول الى switch أخرى ويطلب ادخال نوع المرضى المراد عرضهم

Default:

سيقوم بطباعة invalid input

Case 1:

سيقوم بطباعة كل المرضى الموجودة في hospital وفي حال عدم وجود مرضى سيقوم بطباعة there is no patient

Case2:

سيقوم بطباعة كل المرضى الداخليين الموجودين في hospital وفي حال عدم وجود مرضى داخليين سيقوم بطباعة there is no inner patient

Case3:

سيقوم بطباعة كل المرضى الخارجيين الموجودين في hospital وفي حال عدم وجود مرضى خارجيين سيقوم بطباعة there is no outter patient

Case 10:

سيقوم بعرض جميع المرضى ويطلب تحديد المريض عن طريق ادخال id

سيطلب كلفة وتاريخ المعالجة

اذا كان المريض الذي تم اختياره داخلي سيدخل الى switch و سيطلب اختيار نوع المعالجة

معالجة داخلية:

يطلب اختيار القسم الذي تمت فيه المعالجة وعدد الأطباء

يطلب اختيار الأطباء وفي حال كان عدد الأطباء المتواجدين اقل من عدد الأطباء المطلوبين سيتم طباعة there is no enough doctor

سيتم إضافة معالجة الى سجل المعالجات الخاص بالمريض

معالجة خارجية:

يطلب تحديد الطبيب الذي قام بالمعالجة

اختيار العيادة التي تمت فيها المعالجة

يتم إضافة المعالجة الى سجل المعالجات الخاص بالمريض

اما اذا كان المريض الذي تم اختياره خارجي:

فيطلب تحديد الطبيب الذي قام بالمعالجة

اختيار العيادة التي تمت فيها المعالجة

سيتم إضافة المعالجة الى سجل المعالجات الخاص بالمريض

في حال ادخال id خاطئ سيقوم طباعة invalid id

Case 11:

تقوم بالدخول الى switch أخرى

Default:

سيقوم بطباعة invalid input

Case 1:

تقوم بعرض جميع المرضى المتواجدين في الأقسام في جميع الاوقات

Case 2:

سيطلب تاريخ بداية وتاريخ نهاية للفترة الزمنية المراد تحديد المرضى المتواجدين في الأقسام خلالها

سيقوم بمقارنة سنة البداية وسنة النهاية مع سنة المعالجة الداخلية للمرضى الداخليين

سيقوم بطباعة المرضى

في حال تساوي سنة النهاية او تساوي سنة البداية مع سنة المعالجة الداخلية للمرضى الداخليين سيقوم بمقارنة شهر البداية وشهر النهاية مع شهر المعالجة الداخلية للمرضى الداخليين و يقوم بطباعة المرضى

في حال تساوي شهر النهاية او تساوي شهر البداية مع شهر المعالجة الداخلية للمرضى الداخليين سيقوم بمقارنة يوم البداية ويوم النهاية مع يوم المعالجة الداخلية للمرضى الداخليين و يقوم بطباعة المرضى

Case 12:

يطلب تحديد المريض عن طريق ادخال id الخاص به

يتم طباعة سجل المعالجات الخاص بالمريض

في حال ادخال id خاطئ سيقوم طباعة invalid id

Case 13:

تقوم بالدخول الى switch أخرى

Default:

سيقوم بطباعة invalid input

Case 1:

يعرض عدد المرضى في جميع الأقسام في جميع الأوقات

Case 2:

سيطلب تاريخ بداية وتاريخ نهاية للفترة الزمنية المراد تحديد عدد المرضى المتواجدين في الأقسام خلالها

سيقوم بمقارنة سنة البداية وسنة النهاية مع سنة المعالجة الداخلية للمرضى الداخليين

سيقوم بطباعة عدد المرضى

في حال تساوي سنة النهاية او تساوي سنة البداية مع سنة المعالجة الداخلية للمرضى الداخليين سيقوم بمقارنة شهر البداية وشهر النهاية مع شهر المعالجة الداخلية للمرضى الداخليين ويقوم بطباعة عدد المرضى

في حال تساوي شهر النهاية او تساوي شهر البداية مع شهر المعالجة الداخلية للمرضى الداخليين سيقوم بمقارنة يوم البداية ويوم النهاية مع يوم المعالجة الداخلية للمرضى الداخليين ويقوم بطباعة عدد المرضى