

Higher School of Computer Science 08 May 1945 - Sidi Bel Abbes

Oriented Object Programming

Mini Project

Hospital Patient Record System



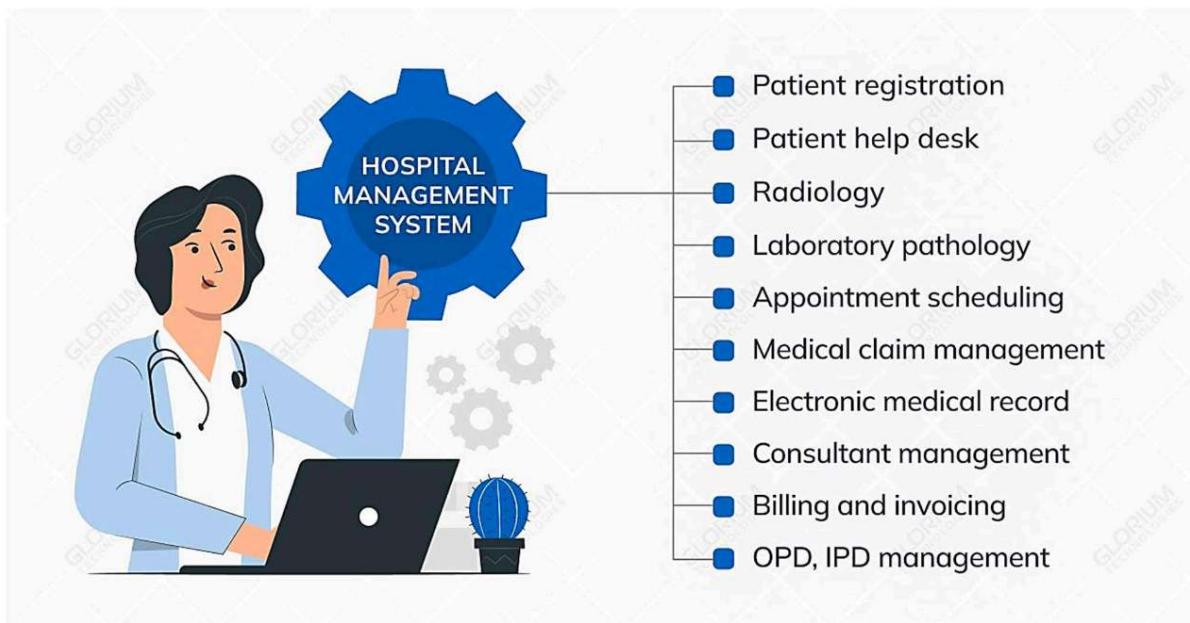
Created By:
Dr, Serhane Oussama

المدرسة العليا لعلوم الحاسب 08 مايو 1945 - سيدى بلعباس

برمجة الكائنات الموجهة

مشروع صغير

نظام سجلات المرضى في المستشفى



تم إنشاؤه بواسطة:
د. سرحان أسامة

Objective:

This mini-project is designed to build a simplified Hospital Patient Record System using Java programming language, illustrating core **Object-Oriented Programming (OOP)** principles such as **inheritance**, **polymorphism**, and **integrating Gui interfaces**.

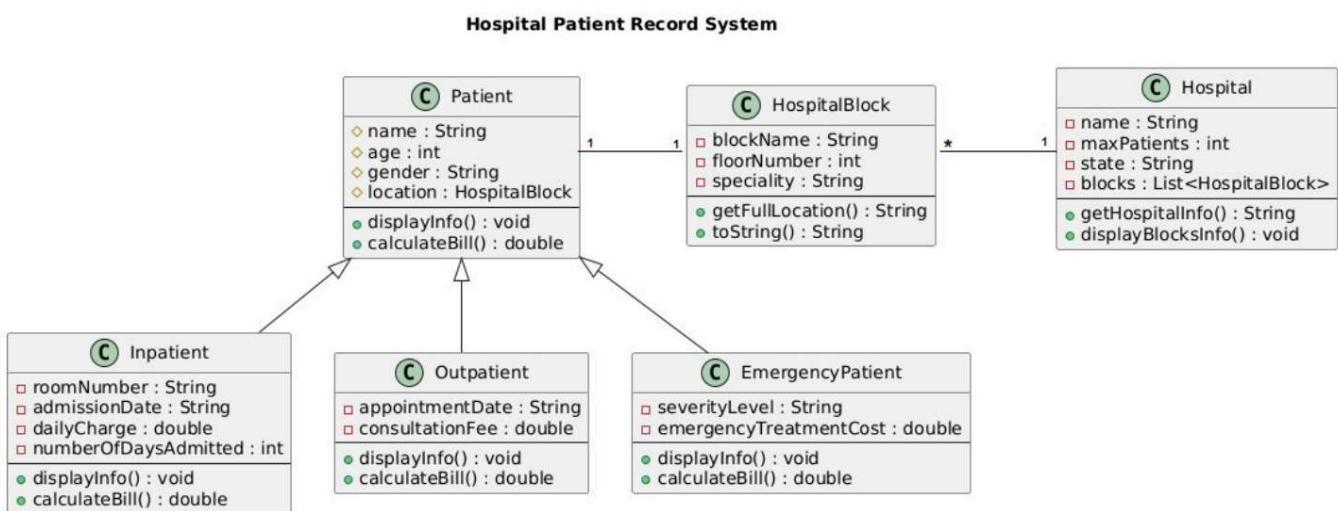
Main Objectives:

- Manage patient records, categorizing patients as **Inpatients**, **Outpatients**, and **Emergency Patients**, each with specialized attributes and behaviors.
- Organize hospital resources by defining a **Hospital** entity, containing multiple specialized **Hospital Blocks** (e.g., Cardiology, Pediatrics, Emergency).
- Implement methods that demonstrate polymorphic (**polymorphism**) behaviors, especially in calculating patient bills and displaying patient information.
- Provide a clear, structured codebase showcasing OOP best practices, serving as a foundation for further GUI integration.

Key Tasks:

- Define relationships clearly between the hospital, hospital blocks, and patients.
- Apply inheritance by creating specialized patient types.
- Use polymorphism to handle patient-specific functionalities.
- Integrate location-based attributes clearly within the system.

Class Diagram: The class diagram for the **Hospital Patient Record System** contains the primary class **Hospital**, which can manage multiple specialized **HospitalBlocks**. Each **HospitalBlock** accommodates various types of patients represented through the superclass **Patient** and its subclasses: **Inpatient**, **Outpatient**, and **EmergencyPatient**. The diagram highlights **inheritance** (Patient subclasses) and **composition** (Hospital to HospitalBlock). And represent as following:



هدف:

تم تصميم هذا المشروع المصغر لبناء نظام مبسط لسجلات المرضى في المستشفى باستخدام لغة برمجة Java ، مما يوضح مبادئ البرمجة الموجهة للકائنات (OOP) الأساسية مثل الوراثة وتعدد الأشكال ودمج واجهات المستخدم الرسومية.

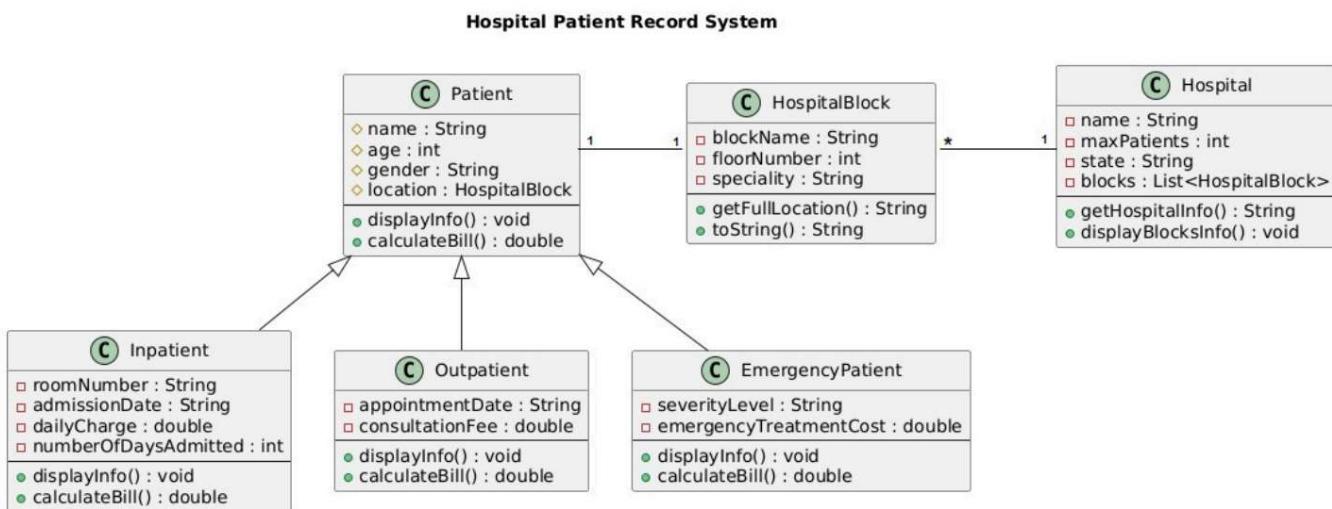
الأهداف الرئيسية:

- إدارة سجلات المرضى ، وتصنيف المرضى على أنهم مرضى داخليون ومرضى خارجيون ومرضى طوارى ، وكل منهم سمات وسلوكيات متخصصة.
- تنظيم موارد المستشفى من خلال تحديد كيان المستشفى ، الذي يحتوي على العديد من كتل المستشفيات المتخصصة (على سبيل المثال ، أمراض القلب وطب الأطفال والطوارى).
- تطبيق الأساليب التي تظهر السلوكيات متعددة الأشكال (تعدد الأشكال) ، خاصة في حساب فواتير المرضى وعرض معلومات المريض. □ توفير قاعدة تعليمات برمجية واضحة ومنظمة تعرض أفضل ممارسات OOP ، وتعمل كأساس لمزيد من تكامل واجهة المستخدم الرسومية.

المهام الرئيسية:

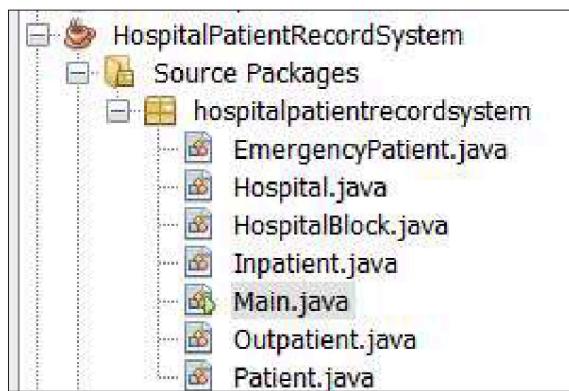
- تحديد العلاقات بوضوح بين المستشفى وكتل المستشفيات والمريض.
- تطبيق الميراث من خلال إنشاء أنواع متخصصة من المرضى.
- استخدام تعدد الأشكال للتعامل مع الوظائف الخاصة بالمريض. • دمج السمات القائمة على الموقع بوضوح داخل النظام.

مخطط الفئة: يحتوي الرسم التخطيطي للفئة لنظام سجل المرضى في المستشفى على مستشفى الفئة الأولية، والذي يمكنه إدارة العديد من HospitalBlocks. ينتسب كل HospitalBlock إلى مستشفى على مستشفى الفئة الأولية، والذي يمكنه إدارة العديد من HospitalBlocks.HospitalBlock هو نوعاً مختلفاً من المرضى الممثلين من خلال الفئة الفرعية للمريض وفاته الفرعية: المرضى الداخليين والخارجيين ومرضى الطوارى. يسلط الرسم البياني الضوء على الوراثة (الفئات الفرعية للمريض) والتكتونين (من المستشفى إلى المستشفى). وتمثل على النحو التالي: HospitalBlock



Phase N1: Core implementation and Testing

In this step, you should implement all classes including the main class, the class structure should appear as following:



To test the core software main functionalities. In the following an example of Main class code:

```
package hospitalpatientrecordsystem;

public class Main {
    public static void main(String[] args) {
        // Create Hospital
        Hospital hospital = new Hospital("Green Valley Hospital", 500, "New York");

        // Create Blocks
        HospitalBlock cardioBlock = new HospitalBlock("A", 1, "Cardiology");
        HospitalBlock pediatricBlock = new HospitalBlock("B", 2, "Pediatrics");
        HospitalBlock emergencyBlock = new HospitalBlock("E", 0, "Emergency");

        hospital.addBlock(cardioBlock);
        hospital.addBlock(pediatricBlock);
        hospital.addBlock(emergencyBlock);

        System.out.println(hospital.getHospitalInfo());
        hospital.displayBlocksInfo();

        System.out.println("\n---- Patient Details ----");

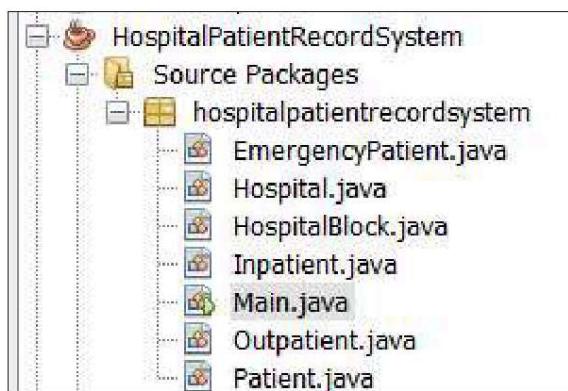
        Patient inpatient = new Inpatient("Alice Smith", 50, "Female", cardioBlock, "A101", "2025-05-01", 300.0, 3);
        Patient outpatient = new Outpatient("Bob Johnson", 35, "Male", pediatricBlock, "2025-05-02", 120.0);
        Patient emergencyPatient = new EmergencyPatient("Chris Lee", 27, "Male", emergencyBlock, "High", 700.0);

        Patient[] patients = {inpatient, outpatient, emergencyPatient};

        for (Patient p : patients) {
            p.displayInfo();
            System.out.println("Bill: $" + p.calculateBill());
            System.out.println("-----");
        }
    }
}
```

المرحلة N1: التنفيذ الأساسي والاختبار

في هذه الخطوة ، يجب عليك تنفيذ جميع الفئات بما في ذلك الفئة الرئيسية ، يجب أن تظهر بنية الفصل على النحو التالي:



لاختبار الوظائف الرئيسية للبرنامج الأساسي. في المثال التالي لرمز الفئة الرئيسية:

```
package hospitalpatientrecordsystem;

public class Main {
    public static void main(String[] args) {
        // Create Hospital
        Hospital hospital = new Hospital("Green Valley Hospital", 500, "New York");

        // Create Blocks
        HospitalBlock cardioBlock = new HospitalBlock("A", 1, "Cardiology");
        HospitalBlock pediatricBlock = new HospitalBlock("B", 2, "Pediatrics");
        HospitalBlock emergencyBlock = new HospitalBlock("E", 0, "Emergency");

        hospital.addBlock(cardioBlock);
        hospital.addBlock(pediatricBlock);
        hospital.addBlock(emergencyBlock);

        System.out.println(hospital.getHospitalInfo());
        hospital.displayBlocksInfo();

        System.out.println("\n---- Patient Details ----");

        Patient inpatient = new Inpatient("Alice Smith", 50, "Female", cardioBlock, "A101", "2025-05-01", 300.0, 3);
        Patient outpatient = new Outpatient("Bob Johnson", 35, "Male", pediatricBlock, "2025-05-02", 120.0);
        Patient emergencyPatient = new EmergencyPatient("Chris Lee", 27, "Male", emergencyBlock, "High", 700.0);

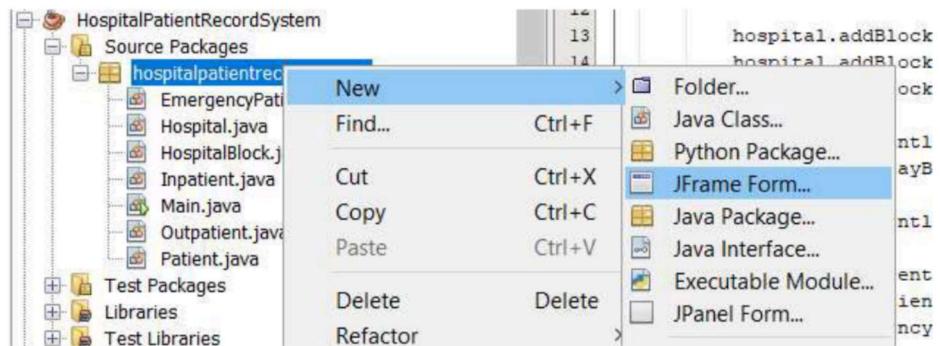
        Patient[] patients = {inpatient, outpatient, emergencyPatient};

        for (Patient p : patients) {
            p.displayInfo();
            System.out.println("Bill: $" + p.calculateBill());
            System.out.println("-----");
        }
    }
}
```

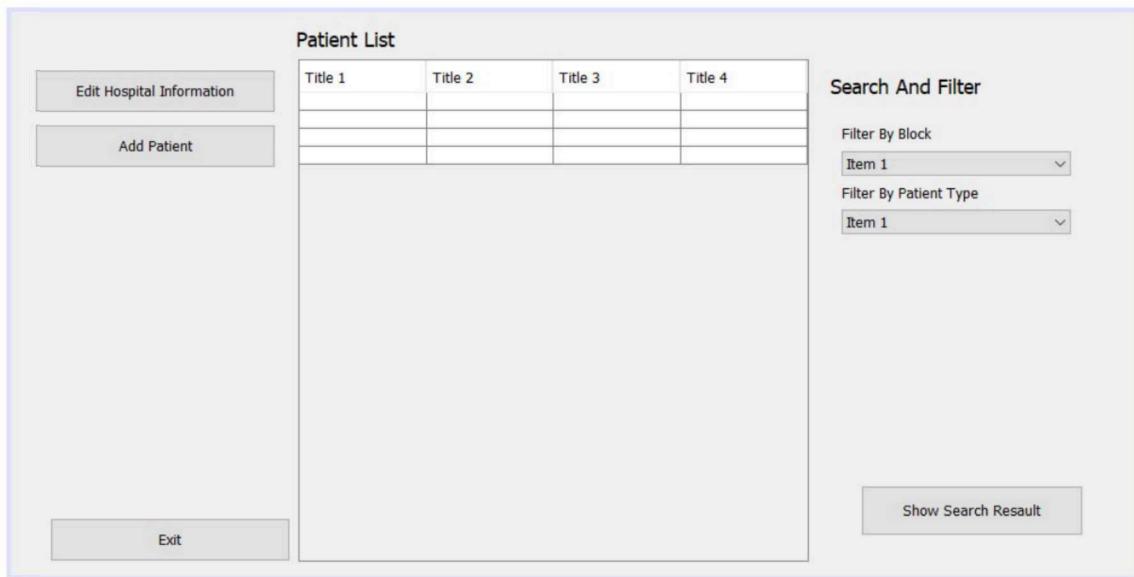
Phase 2: Software Graphical Interface

In this phase, we are going to prepare a graphical interface that contain three frames (window).

- 1/ Create a new JFrame named **HospitalRecordManager** (note that this frame will replace the Main Class) as following



Then, fill out the following components from the left-side palette of NetBeans. The final result of **HospitalRecordManager** window will appear as following:

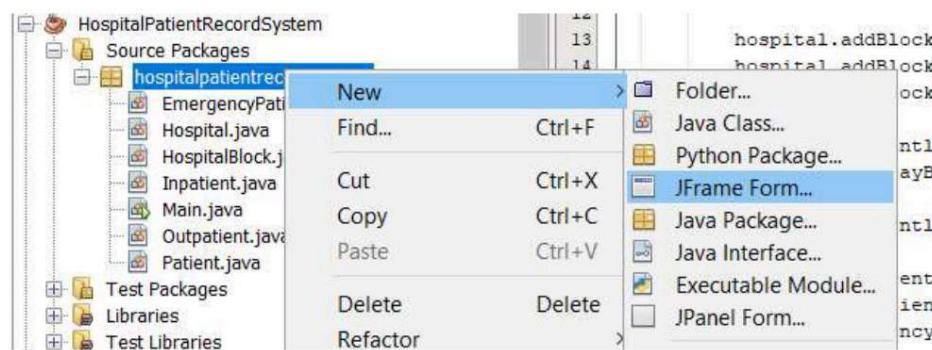


- 2/ Create second Frame that allow us to input the Hospital information, left click on your project and select create new JFrame, then name it **HospitalInformation** as following :

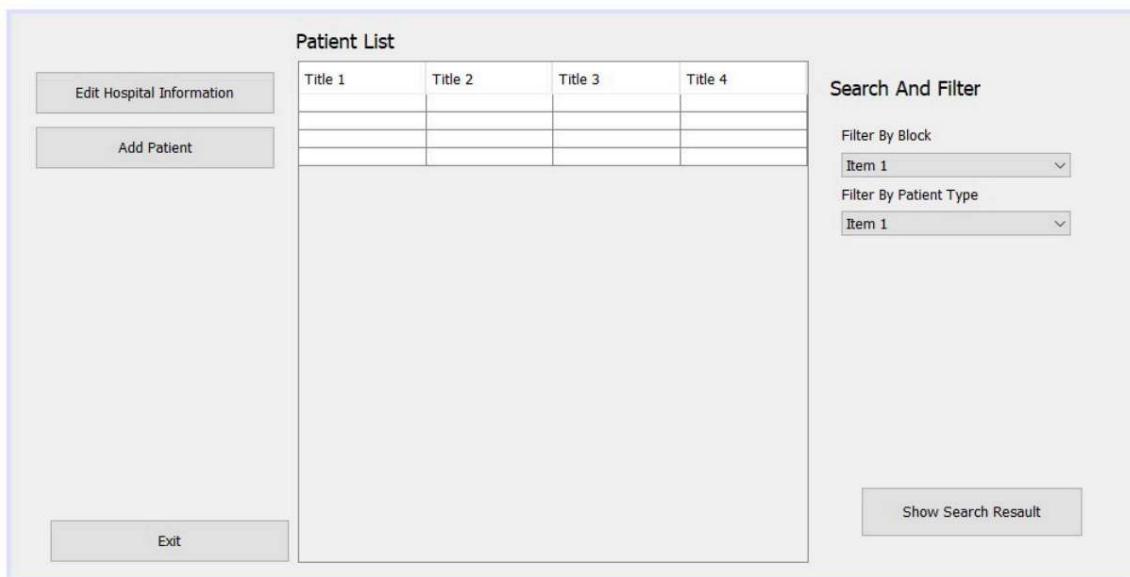
المرحلة 2: واجهة رسومية للبرامج

في هذه المرحلة ، سنقوم بإعداد واجهة رسومية تحتوي على ثلاثة إطارات (نافذة).

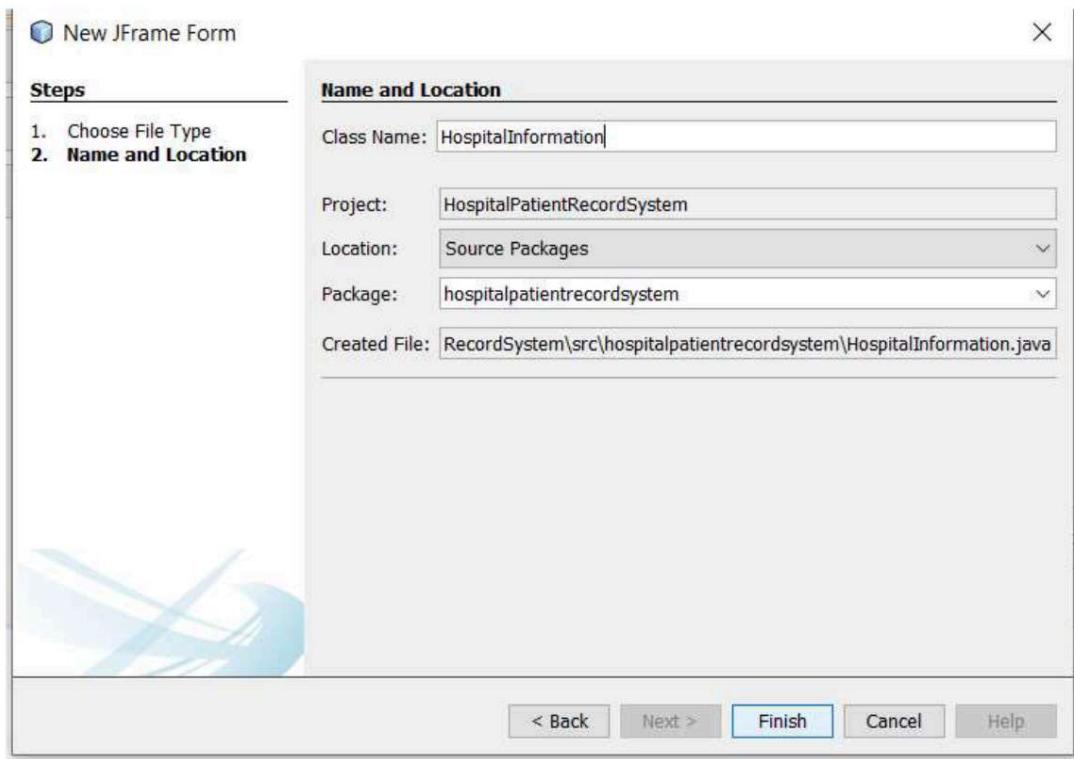
قم بإنشاء JFrame جديد باسم HospitalRecordManager (لاحظ أن هذا الإطار سيحل محل الفئة الرئيسية) على النحو التالي



بعد ذلك ، أملأ المكونات التالية من لوحة الجانب الأيسر من NetBeans. ستظهر النتيجة النهائية لنافذة HospitalRecordManager على النحو التالي:



2 / قم بإنشاء إطار ثان يسمح لنا بادخال معلومات المستشفى ، انقر بزر الماوس الأيسر على مشروعك وحدد إنشاء JFrame جديد ، ثم قم بتسميته HospitalInformation على النحو التالي:



Then, fill out the required component to create an Hospital instance (from Hospital class) as following :

This screenshot shows a form for creating a new hospital instance. It includes fields for basic information and a section for adding hospital blocks.

Fields on the left:

- Hospital Name :
- Max Patients : 100
- State :

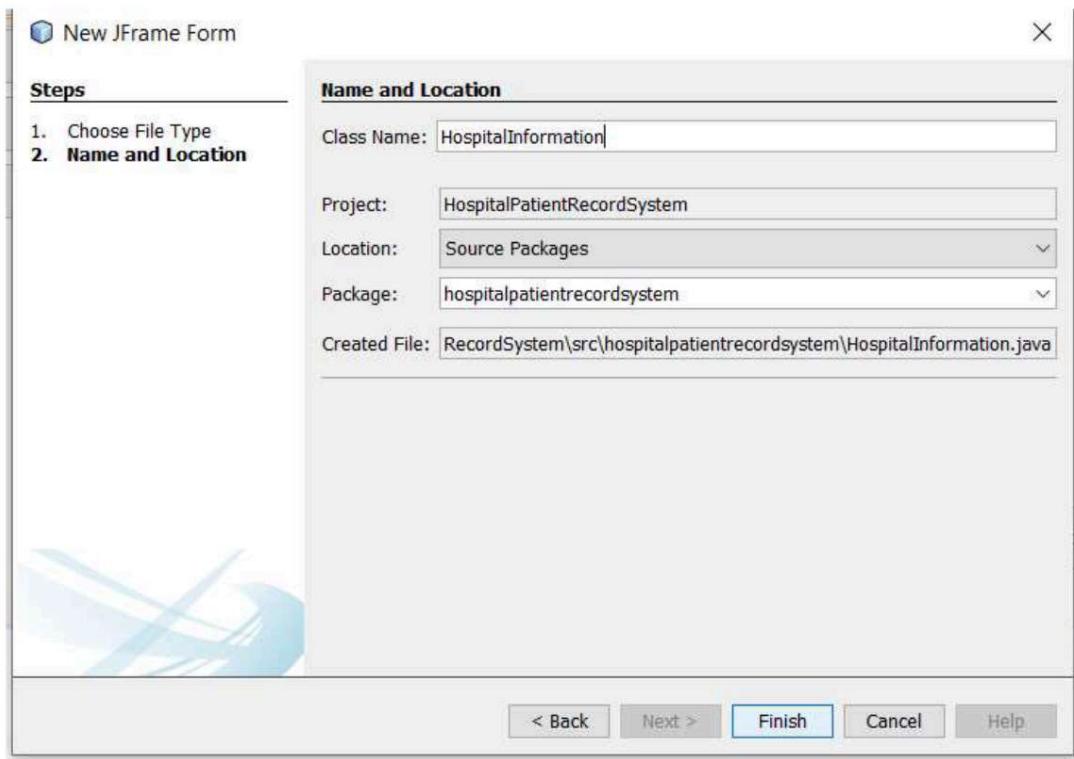
A table titled 'All Blocks' is displayed:

Title 1	Title 2	Title 3	Title 4

To the right, there is a 'Create New Block' section:

- Block Name :
- Floor Number : 1
- Speciality :

Buttons at the bottom right include 'Add Block', 'Save Hospital', and 'Exit'.



بعد ذلك، املأ المكون المطلوب لإنشاء مثيل مستشفى (من فئة المستشفى) على النحو التالي:

The screenshot shows the 'Create New Hospital' screen. It includes the following fields:

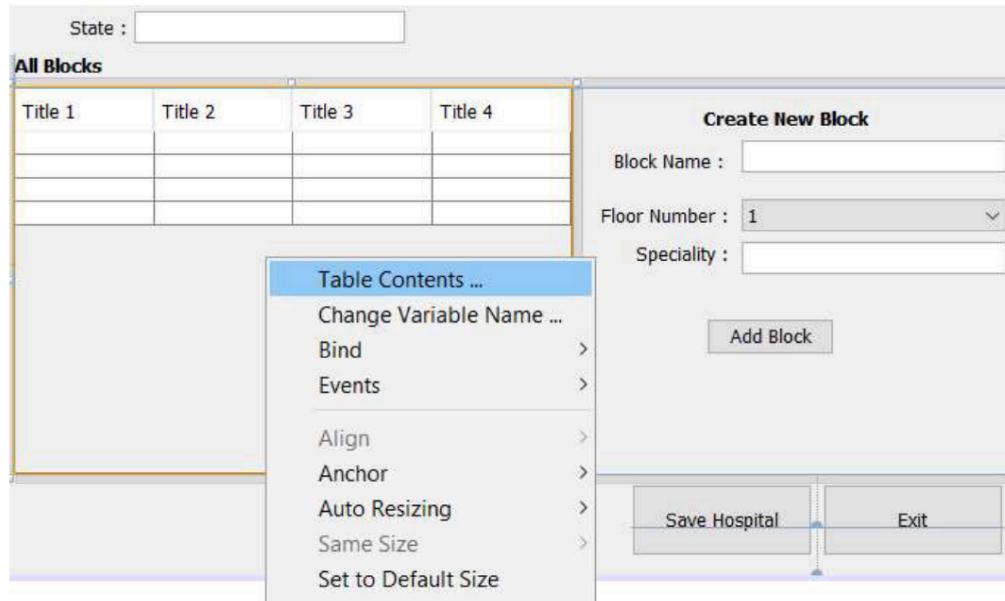
- Hospital Name :
- Max Patients : 100
- State :

A section titled 'All Blocks' displays a table with four columns labeled 'Title 1', 'Title 2', 'Title 3', and 'Title 4'. To the right, there is a 'Create New Block' form with fields for:

- Block Name :
- Floor Number : 1
- Speciality :

Below these forms are 'Save Hospital' and 'Exit' buttons.

You can Edit the Table information (Header) by clicking left and edit Table Contents



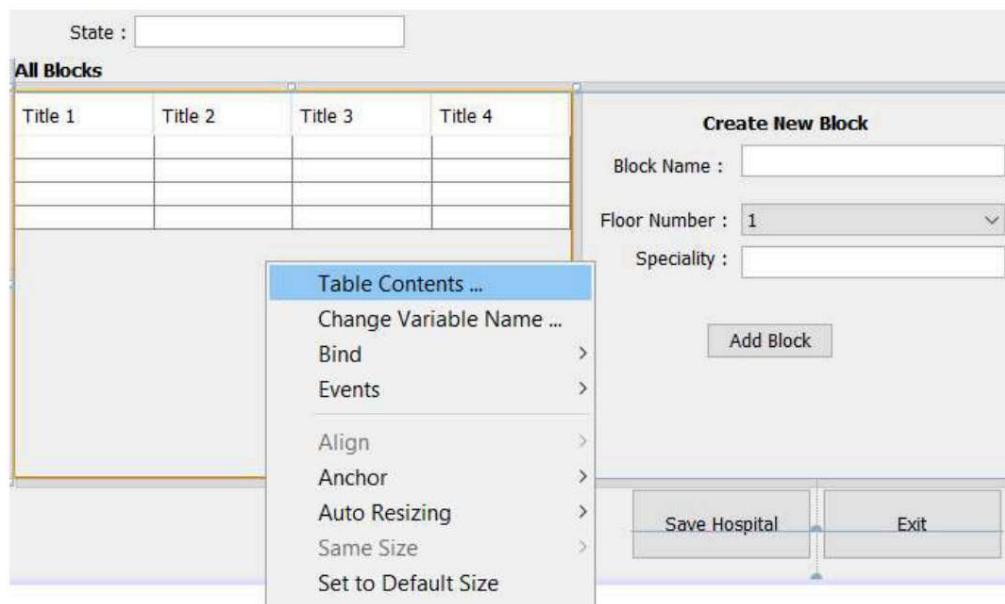
3/ Create Third Frame name it **NewPatient** that allow us to input the patient information as following

The screenshot shows a form titled "Add New Patient". It includes fields for "Patient Name" (text input), "Age" (text input), "Gender" (dropdown menu with "M" selected), "Block Location" (dropdown menu with "Item 1" selected), and three checkboxes for "Emergency Patient", "In patient", and "Out patient". At the bottom is an "Add" button.

Patient Name	<input type="text"/>
Age	<input type="text"/>
Gender	<input type="button" value="M"/>
Block Location	<input type="button" value="Item 1"/>

Emergency Patient In patient Out patient

يمكنك تحرير معلومات الجدول (الرأس) بالنقر على اليسار وتحرير محتويات الجدول



3 / قم بإنشاء إطار ثالث أطلق عليه اسم NewPatient الذي يسمح لنا بادخال معلومات المريض على النحو التالي

Add New Patient

Patient Name

Age

Gender

Block Location

Emergency Patient In patient Out patient

Phase 3: Software Functionalities

In this part we are going to link the functionality of our system

- 1- We need to be able to create and fill the information of our hospital that include hospital information and block information. Double click on **Save Hospital** Button of **HospitalInformation** Frame then we need to gather user input and call Hospital constructor as showing the following code:

```
private void SaveHospitalButtonActionPerformed(java.awt.event.ActionEvent evt) {  
  
    String HospitalName = HospitalNameInput.getText();  
    int HospitalMaxPatientsNumber = Integer.parseInt(HospitalMaxPatients.getSelectedItem().toString());  
    String State = HospitalLocationState.getText();  
    this.NewHospital = new Hospital(HospitalName, HospitalMaxPatientsNumber, State);  
  
}
```

- 2- The trap in this part is the **HospitalRecordManager** (Frame or class) should know about the created object NewHospital, the above code missed one line to add.
- 3- Now, we need to allow the user to create and fill the hospital blocks, then display them in the block table.

```
private void AddBlockButtonActionPerformed(java.awt.event.ActionEvent evt) {  
    String BlockName = BlockNameInput.getText();  
    int BlockFloor = Integer.parseInt(BlockFloorNumber.getSelectedItem().toString());  
    String BlockSpecialityTxt = BlockSpeciality.getText();  
    HospitalBlock NewBlock = new HospitalBlock(BlockName, BlockFloor, BlockSpecialityTxt);  
    this.NewHospital.addBlock(NewBlock);  
    DisplayHospitalBlocks();  
}
```

- 4- The last line of the above code call DisplayHospitalBlocks function, that function loops over all available blocks at our hospital then displays them at the hospital Table, DisplayHospitalBlocks is given as following :

```
public void DisplayHospitalBlocks() {  
  
    DefaultTableModel model = (DefaultTableModel) BlockTable.getModel();  
    if (NewHospital != null) {  
        ArrayList<HospitalBlock> BlockList = NewHospital.getBlocks();  
        if (BlockList.isEmpty()) {  
            model.setRowCount(0);  
        } else {  
            model.setRowCount(BlockList.size());  
            for (HospitalBlock MyBlock : BlockList) {  
                model.removeRow(0);  
                Object[] rowData = {0, MyBlock.getBlockName(), MyBlock.getFloorNumber(), MyBlock.getSpeciality()};  
                model.addRow(rowData);  
            }  
        }  
    }  
}
```

- 5- Now, we are going to implement add patient functionality, when user click on **Add Patient** Button it shows **NewPatient** window the code to shows **NewPatient** given as following:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    NewPatient newWindow = new NewPatient();  
    newWindow.show();  
}
```

- 6- The user now should fill the information of the new patient then save it. After saving the new patient the table of patients should updated by calling **DisplayPatients()** function, the code of saving patient is given as follows:

المرحلة 3: وظائف البرامج

في هذا الجزء سنقوم بربط وظائف نظامنا

1- نحتاج إلى أن تكون قادرين على إنشاء وملء معلومات مستشفانا التي تتضمن معلومات المستشفى ومعلومات الحظر. انقر نقرًا مزدوجًا فوق زر حفظ المستشفى في إطار معلومات المستشفى ، ثم نحتاج إلى جمع مدخلات المستخدم والاتصال بالمستشفى

المنشئ كما يظهر التعليمات البرمجية التالية:

```
private void SaveHospitalButtonActionPerformed(java.awt.event.ActionEvent evt) {  
  
    String HospitalName = HospitalNameInput.getText();  
    int HospitalMaxPatientsNumber = Integer.parseInt(HospitalMaxPatients.getSelectedItem().toString());  
    String State = HospitalLocationState.getText();  
    this.NewHospital = new Hospital(HospitalName, HospitalMaxPatientsNumber, State);  
  
}
```

2- يجب أن يعرف الملامنة في هذا الجزء HospitalRecordManager (الإطار أو الفئة) الكائن الذي تم إنشاؤه HospitalRecordManager ، وقد غاب الرمز أعلاه عن سطر واحد لإضافته.

3- الآن ، نحتاج إلى السماح للمستخدم بإنشاء كتل المستشفى وتعبئتها ، ثم عرضها في جدول الكتلة.

```
private void AddBlockButtonActionPerformed(java.awt.event.ActionEvent evt) {  
    String BlockName = BlockNameInput.getText();  
    int BlockFloor = Integer.parseInt(BlockFloorNumber.getSelectedItem().toString());  
    String BlockSpecialityTxt = BlockSpeciality.getText();  
    HospitalBlock NewBlock = new HospitalBlock(BlockName, BlockFloor, BlockSpecialityTxt);  
    this.NewHospital.addBlock(NewBlock);  
    DisplayHospitalBlocks();  
}
```

4- السطر الأخير من وظيفة استدعاء التعليمات البرمجية أعلاه DisplayHospitalBlocks ، والتي تدور هذه الوظيفة عبر جميع الكتل المنشاة في مستشفانا ثم يتم عرضها بعد ذلك على طاولة المستشفى ، يتم إعطاء DisplayHospitalBlocks على النحو التالي:

```
public void DisplayHospitalBlocks() {  
  
    DefaultTableModel model = (DefaultTableModel) BlockTable.getModel();  
    if (NewHospital != null) {  
        ArrayList<HospitalBlock> BlockList = NewHospital.getBlocks();  
        if (BlockList.isEmpty()) {  
            model.setRowCount(0);  
        } else {  
            model.setRowCount(BlockList.size());  
            for (HospitalBlock MyBlock : BlockList) {  
                model.removeRow(0);  
                Object[] rowData = {0, MyBlock.getBlockName(), MyBlock.getFloorNumber(), MyBlock.getSpeciality()};  
                model.addRow(rowData);  
            }  
        }  
    }  
}
```

5- الآن ، سنقوم بتنفيذ وظيفة إضافة المريض ، عندما ينقر المستخدم على زر إضافة مريض ، فإنه يظهر نافذة NewPatient على النحو التالي:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    NewPatient newWindow = new NewPatient();  
    newWindow.show();  
}
```

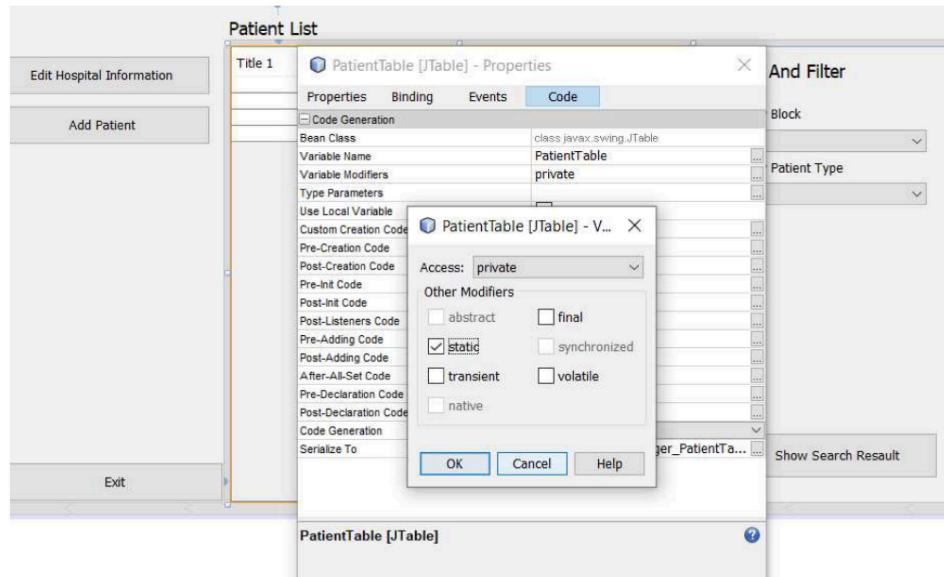
6- يجب على المستخدم الآن ملء معلومات المريض الجديد ثم حفظها. بعد حفظ المريض الجديد ، يجب تحديث جدول المرضى عن طريق استدعاء وظيفة DisplayPatients () ، ويتم إعطاء رمز إنفاذ المريض على النحو التالي:

```

11  public class HospitalRecordManager extends javax.swing.JFrame {
12
13     static Hospital CurHospital;
14     static ArrayList<Patient> PatientList;
15
16     public HospitalRecordManager() {
17         initComponents();
18     }
19
20     public static void DisplayPatients() {
21         DefaultTableModel model = (DefaultTableModel) PatientTable.getModel();
22         model.setRowCount(PatientList.size());
23         for (Patient MyPatient : PatientList) {
24             model.removeRow(0);
25             Object[] rowData = {0, MyPatient.name, MyPatient.gender, MyPatient.location.getBlockName()};
26             model.addRow(rowData);
27         }
28     }

```

Note: You need to change PatientTable modifier to static by access to table property/Code as follows:



By clicking **Add** button on new patient frame, the flow code should call a constructor based on patient type, then create a new patient instance, then save it in global patient list of **HospitalRecordManager** class, the code is given as follows:

- The following code shows what will happen when a user click on Add patient. It first collect the input information then create appropriate object instance (from different patient classes) call constructor based on selected patient type. And finally call **DisplayPatients()** function to update the table of all patients.

```

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    String PatientName = PatientNameInput.getText();
    int PatientAge = Integer.parseInt(PatientAgeInput.getText());
    String GenderName = GenderNameInput.getSelectedItem().toString();
    int selectedBlockIndex = BlockLocationInput.getSelectedIndex();
    boolean isEmergency = isEmergencyInput.isSelected();
    boolean isInPatient = isInPatientInput.isSelected();
    boolean isOutPatient = isOutPatientInput.isSelected();

    HospitalBlock PatientLocation = BlockList.get(selectedBlockIndex);

    if (isEmergency) {
        Patient emergencyPatient = new EmergencyPatient(PatientName, PatientAge, GenderName, PatientLocation, "Low", 100.0);
        HospitalRecordManager.PatientList.add(emergencyPatient);
    } else if (isInPatient) {
        Patient inPatient = new Inpatient(PatientName, PatientAge, GenderName, PatientLocation, "Room 1", "15/03/2025", 10.0, 10);
        HospitalRecordManager.PatientList.add(inPatient);
    } else if (isOutPatient) {
        Patient outPatient = new Outpatient(PatientName, PatientAge, GenderName, PatientLocation, "15/03/2025", 10.0);
        HospitalRecordManager.PatientList.add(outPatient);
    }
    HospitalRecordManager.DisplayPatients();
}

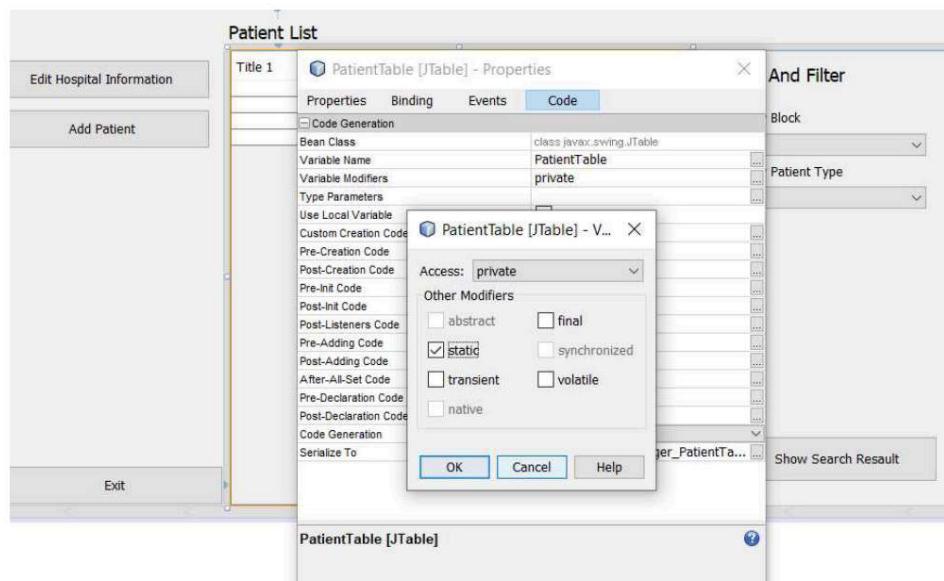
```

```

11  public class HospitalRecordManager extends javax.swing.JFrame {
12
13     static Hospital CurHospital;
14     static ArrayList<Patient> PatientList;
15
16     public HospitalRecordManager() {
17         initComponents();
18     }
19
20     public static void DisplayPatients(){
21         DefaultTableModel model = (DefaultTableModel) PatientTable.getModel();
22         model.setRowCount(PatientList.size());
23         for (Patient MyPatient : PatientList) {
24             Object[] rowData = {0, MyPatient.name, MyPatient.gender, MyPatient.location.getBlockName()};
25             model.addRow(rowData);
26         }
27     }
28
29

```

يتم إعطاء رمز إنقاذ المريض على النحو التالي: ملاحظة: تحتاج إلى تغيير معدل PatientTable إلى ثابت من خلال الوصول إلى ملائمة الجدول / الرمز على النحو التالي:



بالنقر فوق الزر "إضافة" في إطار المريض الجديد ، يجب أن يستدعي رمز التدفق منشئ بناء على نوع المريض ، ثم قم بإنشاء مثيل مريض جديد ، ثم يحفظه في قائمة المرضى العمومية لفئة HospitalRecordManager ، يتم إعطاء التعليمات البرمجية على النحو التالي: 7. يوضح الكود التالي ما سيحدث عندما ينقر المستخدم على إضافة مريض. يقوم أولاً بجمع معلومات الإدخال ثم إنشاء مثيل كائن مناسب (من فئات مختلفة من المرضى) بناء على نوع المريض المحدد. وأخيراً استدعاء وظيفة DisplayPatients() لتحديث جدول جميع المرضى.

```

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    String PatientName = PatientNameInput.getText();
    int PatientAge = Integer.parseInt(PatientAgeInput.getText());
    String GenderName = GenderNameInput.getSelectedItem().toString();
    int selectedBlockIndex = BlockLocationInput.getSelectedIndex();
    boolean isEmergency = isEmergencyInput.isSelected();
    boolean isInPatient = isInPatientInput.isSelected();
    boolean isOutPatient = isOutPatientInput.isSelected();

    HospitalBlock PatientLocation = BlockList.get(selectedBlockIndex);

    if (isEmergency) {
        Patient emergencyPatient = new EmergencyPatient(PatientName, PatientAge, GenderName, PatientLocation, "Low", 100.0);
        HospitalRecordManager.PatientList.add(emergencyPatient);
    } else if (isInPatient) {
        Patient inPatient = new Inpatient(PatientName, PatientAge, GenderName, PatientLocation, "Room 1", "15/03/2025", 10.0, 10);
        HospitalRecordManager.PatientList.add(inPatient);
    } else if (isOutPatient) {
        Patient outPatient = new Outpatient(PatientName, PatientAge, GenderName, PatientLocation, "15/03/2025", 10.0);
        HospitalRecordManager.PatientList.add(outPatient);
    }
    HospitalRecordManager.DisplayPatients();
}

```

8. Your final task now is to implement search and filter functionalities that allow us to:
 - a. Perform a search on list of patient by name
 - b. Filter the patients list based on their block

Mini Project submission and evaluation :

This mini project is due before the final exam as a maximum deadline. Adding more functionalities, idea, styling will be counted.

The submission should be at Lab session (you are not allowed to submit it on other group sessions), you are free to submit it at any session yet the evaluation will differ based on the following dates:

Submission Date (until)	Max Mark
12/04/2025	18/20
19/04/2025	16/20
26/04/2025	14/20
03/05/2025	12/20
10/05/2025	10/20

Note: That is the maximum mark, not the deserved mark. Your mark will reflect your work and will be decided by your lab supervisor.

8. مهمتك النهائية الآن هي تنفيذ وظائف البحث والتصفيية التي تسمح لنا بما يلي: أ. قم بإجراء بحث في قائمة المرضى بالاسم

b. تصفيية قائمة المرضى بناء على الحظر الخاص بهم

تقديم المشروع المصغر وتقديره:

من المقرر أن يصدر هذا المشروع المصغر قبل الاختبار النهائي كحد أقصى للموعد النهائي. سيتم احتساب إضافة المزيد من الوظائف وال فكرة والتصميم.

يجب أن يكون التقديم في جلسة المختبر (لا يسمح لك بتقديمه في جلسات جماعية أخرى) ، وأنتم حر في تقديمها في أي جلسة ولكن سيختلف التقييم بناء على التواريخ التالية:

تاریخ التقديم (حتی)	Max Mark
12/04/2025	18/20
19/04/2025	16/20
26/04/2025	14/20
03/05/2025	12/20
10/05/2025	10/20

ملاحظة: هذه هي العالمة القصوى ، وليس العالمة المستحقة. ستعكس علامتك عمالك وسيقررها مشرف المختبر الخاص بك.