University of Babylon
College of Information Technology
Department of Cyber Security

Object-Oriented Programming
Group Project

Students Names:

Ayat Alaa Hamdi

Batool Ali Talib

Tabark Hussien Hashem

Ayham Mahdi Hassan

Study: Second Year / Morning Study
Class: First Class (A1)

مشروع إدارة صحة المرضى القلبية

يهدف هذا المشروع إلى تطوير نظام بسيط لإدارة صحة المرضى القلبية من خلال تقديم مجموعة من الميزات التي تركز على:

1. إدارة بيانات المرضى:

- تسجيل معلومات المرضى مثل الاسم، العمر، مكان العمل، والمحافظة.
- متابعة الحالات الصحية لكل مريض من خلال سجلات ضغط الدم ودقات القلب.

2. تحليل البيانات الصحية:

- تحليل سجلات ضغط الدم (الانقباضي والانبساطي) لتحديد مستوى الخطورة.
- تحليل عدد دقات القلب لتحديد الحالة الصحية ومستوى الخطورة.

3. تقديم النصائح والأدوية المناسبة:

- توفير نصائح مخصصة لكل حالة مثل "تجنب الكافيين" أو "تقليل تناول الملح".
- اقتراح الأدوية المناسبة بناءً على نوع الحالة الصحية.

4. إدارة العيادة الطبية:

- تسجيل قائمة بالمرضى المتابعين في العيادة.
- تسجيل قائمة بالأطباء المتخصصين في علاج أمراض القلب.
- إسناد طبيب متخصص لكل مريض حسب الحاجة.

5. إدارة شاملة لسجلات المريض:

- دمج جميع السجلات (النصائح، الأدوية، وسجلات ضغط الدم ودقات القلب) لتوفير صورة كاملة عن الحالة الصحية للمريض.

1.Class BloodPressure

يمثل قياس ضغط الدم وحساب مستوى الخطورة المرتبط به

systolic: قيمة ضغط الدم الانقباضي (أعلى قراءة في ضغط الدم) | Diastolic: قيمة ضغط الدم الانبساطي (أدنى قراءة في ضغط الدم) | riskLevel : مؤشر يحدد مستوى الخطورة

```
    // Fields
    private int systolic;       // systolic blood pressure value
    private int diastolic;      // Diastolic blood pressure value
    private String riskLevel;   // The risk level (e.g., "High Risk", "Medium Risk",
or "Normal")
```

calculateRiskLevel() : يتم استدعاؤها فور إنشاء الكائن لحساب مستوى الخطورة

```
   // Constructor
    public BloodPressure(int systolic, int diastolic) {
        this.systolic = systolic;        // Set the systolic value
        this.diastolic = diastolic;      // Set the diastolic value
        calculateRiskLevel();            // Calculate the risk level based on the
blood pressure values
    }
```

دالة لتحديد مستوى الخطورة بناءا على قيم المدخلة لضغط الدم الانقباضي و ضغط الدم الانبساطي

```
// Determines the risk level based on systolic and diastolic values
     protected void calculateRiskLevel() {
        // High risk if systolic is greater than 140 or diastolic is greater than 90
        if (systolic > 140 || diastolic > 90) {
            riskLevel = "High Risk";
        }
        // Medium risk if systolic is between 120 and 140, or diastolic is between 80
and 90
        else if (systolic >= 120 || diastolic >= 80) {
            riskLevel = "Medium Risk";
        }
        // Normal if systolic is less than 120 and diastolic is less than 80
        else {
            riskLevel = "Normal";
        }
    }
```

دالة عند استدعائها تقوم بارجاع معلومات ضغط الدم (الانقباضي و الانبساطي)

```
    // Returns a string representation of the blood pressure values
    public String getPressureInfo() {
        return "Systolic: " + systolic + ", Diastolic: " + diastolic;
    }
```

دالة عند استدعائها تقوم بارجاع مستوى الخطورة والذي يتم تحديده مسبقا في ال constructor عند استدعاء دالة calculateRiskLevel()

```
    // Returns the current risk level
    public String getRiskLevel() {
        return riskLevel;
    }
```

2.Class HeartRate

يمثل معدل نبضات القلب وحساب مستوى الخطورة المرتبط به

bpm (Beats Per Minute): معدل ضربات القلب في الدقيقة | riskLevel: يشير إلى مستوى الخطورة (مثل "High Risk" أو "Normal").

```
    // Fields
    private int bpm;              // Heart rate in beats per minute (BPM)
    private String riskLevel;     // The risk level based on the heart rate
```

calculateRiskLevel() : يتم استدعاؤها فور إنشاء الكائن لحساب مستوى الخطورة

```
    // Constructor
    public HeartRate(int bpm) {
        this.bpm = bpm;             // Set the heart rate value (BPM)
        calculateRiskLevel();       // Calculate the risk level based on the BPM
value
    }
```

دالة لتحديد مستوى الخطورة بناءا على قيم المدخلة لعدد ضربات القلب في الدقيقة الواحدة

```
    // Determines the risk level based on the heart rate (BPM)
     protected void calculateRiskLevel() {
        // High risk if BPM is less than 60 or greater than 100
        if (bpm < 60 || bpm > 100) {
            riskLevel = "High Risk";
        }
        // Medium risk if BPM is between 60 and 70 (inclusive)
        else if (bpm >= 60 && bpm <= 70) {
            riskLevel = "Medium Risk";
        }
        // Normal if BPM is between 71 and 100 (inclusive)
        else {
            riskLevel = "Normal";
        }
    }
```

تُرجع نصًا يحتوي على كل من معدل ضربات القلب ومستوى الخطورة بصيغة"BPM: <value>, Risk Level: <value>

```
    // Returns a string with both BPM and risk level information
    public String getHeartRateInfo() {
        return "BPM: " + bpm + ", Risk Level: " + riskLevel;
    }
```

دالة عند استدعائها تقوم بارجاع مستوى الخطورة والذي يتم تحديده مسبقا في ال constructor عند استدعاء دالة calculateRiskLevel()

```
    // Returns the current risk level
    public String getRiskLevel() {
        return riskLevel;
    }
```

3.Class Advice

يمثل النصائح او التوصيات المرتبطة بأنواع المخاطر (مثل ضغط الدم المرتفع والخ ...)

riskType: يمثل نوع المخاطر (مثل "High Blood Pressure") | adviceList: قائمة ديناميكية (ArrayList) تحتوي على النصائح المرتبطة بهذا النوع من المخاطر.

```
    // Fields
    private String riskType;            // Stores the risk type (e.g., "High Blood
Pressure")
    private ArrayList<String> adviceList; // Holds a list of advice related to the
risk
```

يتم تحديد قيمة نوع الخطورة واحلال المصفوفة الخاصة للتوصيات او النصائح المرتبطة بنوع الخطورة

```
    // Constructor
    public Advice(String riskType) {
        this.riskType = riskType;      // Set the risk type (e.g., "High Blood
Pressure")
        adviceList = new ArrayList<>(); // Initialize the list to store advice
```

دالة لاضافة عناصر للمصفوفة

```
// Adds a piece of advice to the list
    public void addAdvice(String advice) {
        adviceList.add(advice); // Add the advice to the list
    }
```

استرجاع قائمة النصائح إذا تطابق نوع المخاطر المرسل مع الحقلriskType المحدد في constructor

```
    // Retrieves the list of advice for the given risk type
    public ArrayList<String> getAdvice(String riskType) {
        // If the risk type matches, return the advice list
        if (this.riskType.contains(riskType)) {
            return adviceList;
        }
        // Return null if the risk type doesn't match
        return null;
    }
```

يُرجع تمثيلًا نصيًا للكائن يحتوي على نوع المخاطر وقائمة النصائح المرتبطة بنوع الخطورة

```
    // String representation of the object
    @Override
    public String toString() {
        // Build the string representation by concatenating the risk type and advice
list
        String result = "Risk Type: " + riskType + ", Advice: " + adviceList + "\n";
        return result;
    }
```

4.Class Medication

يمثل الأدوية الموصوفة لحالة طبية معينة

condition: يمثل اسم الحالة الطبية التي ترتبط بها الأدوية | medications: قائمة ديناميكية (ArrayList) تحتوي على أسماء الأدوية الموصوفة للحالة

```
    // Fields
    private String condition;          // The medical condition for which the
medication is prescribed
    private ArrayList<String> medications;  // List of medications prescribed for the
condition
    // Constructor
    public Medication(String condition) {
        this.condition = condition;    // Initialize the condition
        medications = new ArrayList<>(); // Initialize the list of medications
    }
```

دالة لاضافة عناصر للمصفوفة

```
   // Adds a medication to the list
    public void addMedication(String medication) {
        medications.add(medication);    // Add the medication to the list
    }
```

استرجاع قائمة الأدوية الموصوفة إذا كانت الحالة المرسلة مطابقة (أو تحتوي على) اسم الحالة المخزنة في الكائن

```
    // Returns a list of medications for a given condition
    public ArrayList<String> getMedications(String condition) {
        // Check if the current condition contains the given condition
        if (this.condition.contains(condition)) {
            return medications;  // Return the list of medications if the condition
matches
        }
        return null;  // Return null if the risk type doesn't match
    }
```

إرجاع تمثيل نصي للكائن يحتوي على اسم الحالة وقائمة الأدوية الموصوفة

```
    // Overridden toString method to display the medication details
    @Override
    public String toString() {
        // Initialize the result string with the condition
        String result = "Medication for " + condition + ":\n";
        // Loop through the medications list and append each medication to the result
string
        for (String med : medications) {
            result += "- " + med + "\n";
        }
        return result;  // Return the result string
    }
```

5.Class Patient

يُمثل بيانات مريض في النظام، مثل اسمه، عمره، مكان عمله، المدينة التي يسكن بها، وسجلاته الصحية (ضغط الدم، نبض القلب، الأدوية، والنصائح)

```
    // Fields
    private String name;  // Name of the patient
    private int age;      // Age of the patient
    private String workplace;  // Workplace of the patient
    private String city;       // City where the patient lives
    private ArrayList<BloodPressure> bloodPressureRecords;  // List of blood pressure
records
    private ArrayList<HeartRate> heartRateRecords;          // List of heart rate
records
    private ArrayList<Medication> medications;              // List of medications
prescribed
    private ArrayList<Advice> advices;                      // List of advices given
to the patient
```

تهيئة كائن المريض ببياناته الأساسية (اسم، عمر، مكان العمل، المدينة) | إنشاء قوائم فارغة لكل من سجلات ضغط الدم، نبض القلب، الأدوية، والنصائح

```
    // Constructor
    public Patient(String name, int age, String workplace, String city) {
        this.name = name;
        this.age = age;
        this.workplace = workplace;
        this.city = city;
        bloodPressureRecords = new ArrayList<>();  // Initialize blood pressure
records list
        heartRateRecords = new ArrayList<>();     // Initialize heart rate records
list
        medications = new ArrayList<>();          // Initialize medications list
        advices = new ArrayList<>();              // Initialize advices list
    }
```

استرجاع بيانات المريض أو القوائم (المصفوفات) المرتبطة به

```
    // Getters
    public String getPatientINFO() {
        return "Name: " + name + " , Age: " + age + " , Workplace: " + workplace + "
, City: " + city;
    }

    public int getAge() {
        return age;
    }

    public String getCity() {
        return city;
    }

    public String getName() {
        return name;
    }
```

```
    public String getWorkplace() {
        return workplace;
    }

    public ArrayList<Medication> getMedications() {
        return medications;
    }

    public ArrayList<BloodPressure> getBpRecords() {
        return bloodPressureRecords;
    }

    public ArrayList<HeartRate> getHrRecords() {
        return heartRateRecords;
    }

    public ArrayList<Advice> getAdvices() {
        return advices;
    }
```

تعديل بيانات المريض (مثل العمر أو المدينة)

```
    // Setters
    public void setAge(int age) {
        if (age > 0) {
            this.age = age;
        } else {
            System.out.println("Age cannot be negative");
        }
    }

    public void setCity(String city) {
        this.city = city;
    }

    public void setWorkplace(String workplace) {
        this.workplace = workplace;
    }

    public void setName(String name) {
        this.name = name;
    }
```

التحقق إذا كان المريض معرضًا للخطر بناءً على سجلات ضغط الدم أو معدل نبض القلب

```
    // Method to check if the patient is at risk based on blood pressure and heart
rate
    public boolean isAtRisk() {
        // Check Blood Pressure Risk
        for (BloodPressure bp : bloodPressureRecords) {
            if (!bp.getRiskLevel().equals("Normal")) {
                return true;  // Patient is at risk if any blood pressure is not
normal
            }
        }

        // Check Heart Rate Risk
```

```
        for (HeartRate hr : heartRateRecords) {
            if (!hr.getRiskLevel().equals("Normal")) {
                return true;  // Patient is at risk if any heart rate is not normal
            }
        }

        return false; // No risks found
    }
```

إضافة سجلات جديدة للمريض (اضافة عناصر للمصفوفات)

```
    // Methods to add records for blood pressure, heart rate, medications, and
advices
    public void addBloodPressureRecord(BloodPressure bp) {
        bloodPressureRecords.add(bp);
    }

    public void addHeartRateRecord(HeartRate hr) {
        heartRateRecords.add(hr);
    }

    public void addAdvice(Advice advice) {
        advices.add(advice);
    }

    public void addMedication(Medication medication) {
        medications.add(medication);
    }
```

إعادة تمثيل نصي (String) للكائن، يحتوي على تفاصيل المريض الأساسية (استدعاء لدالة getPatientINFO())

```
   // Override toString method to return patient info
    @Override
    public String toString() {
        return getPatientINFO();  // This will return the string with patient's
details
    }
```

6.Class Clinic

يُمثل عيادة تحتوي على معلومات المرضى، والأطباء (اختصاصيي القلب)

name: اسم العيادة | patients: قائمة ديناميكية تحتوي على كائنات المرضىPatient المسجلين في العيادة | cardiologists: قائمة ديناميكية تحتوي على أسماء اختصاصيي القلب المرتبطين بالمرضى

```
    // Fields
    private String name;                      // Name of the clinic
    private ArrayList<Patient> patients;      // List of patients in the clinic
    private ArrayList<String> cardiologists;  // List of cardiologists assigned to
patients
```

```
    // Constructor
    public Clinic(String name) {
        this.name = name;                     // Initialize the clinic's name
        this.patients = new ArrayList<>();    // Initialize the list of patients
        cardiologists = new ArrayList<>(); // Initialize the list of cardiologists
    }
```

إضافة كائن مريض جديد إلى قائمة المرضى في العيادة (اضافة عناصر للمصفوفة)

```
    // Adds a patient to the clinic
    public void addPatient(Patient patient) {
        patients.add(patient);                // Add the given patient to the list of
patients
    }
```

تعيين طبيب مختص بالقلب لمريض محدد بعد التحقق من وجود المريض في سجلات (مصفوفة) المرضى

```
    // Assigns a cardiologist to a patient
    public void assignCardiologist(Patient patient, String cardiologist) {
        // Check if the patient is registered in the clinic
        if (!patients.contains(patient)) {
            System.out.println("Patient not registered in the clinic.");
            return;  // Exit if patient is not registered
        }
        cardiologists.add(cardiologist);      // Add the cardiologist to the list of
assigned cardiologists
        System.out.println("Cardiologist assigned to patient " +
patient.getPatientINFO());
    }
```

عرض اسم العيادة وقائمة المرضى المسجلين فيها باستخدام Iterator

```
   // Displays the clinic's information
    public void displayClinicInfo() {
        System.out.println("Clinic: " + name);  // Display the clinic's name
        System.out.println("Patients in Clinic:");
        // Check if there are any patients in the clinic
        if (patients.isEmpty()) {
            System.out.println("No patients available."); // Display a message if no
patients exist
        } else {
            // Use an iterator to go through the list of patients
            Iterator<Patient> iterator = patients.iterator();
            while (iterator.hasNext()) {
```

```
                Patient p = iterator.next();  // Get the next patient in the list
                System.out.println(p);  // Print the patient's information (calls the
overridden toString method in Patient)
            }
        }
    }
```

7.Class Main

الغرض الرئيسي من كلاس main هو تنسيق عمل الكلاسات المختلفة التي تم تعريفها، وتنفيذ البرنامج من خلال: إنشاء الكائنات وربط العلاقات بينها (مرضى، نصائح، أدوية) واخيرا عرض النتائج (مثل معلومات المرضى والعيادة)

هذه الدالة مخصصة لعرض النصائح والأدوية المرتبطة بمريض معين: تعرض رسالة توضح اسم المريض و كذلك تتحقق مما إذا كان المريض في حالة خطر بناءً على سجلاته الصحية (isAtRisk) وتعرض النصائح (إذا كانت متوفرة) أو رسالة تفيد بعدم وجود نصائح وايضا تعرض الأدوية (إذا كانت متوفرة) أو رسالة تفيد بعدم وجود أدوية وكذلك تظهر رسالة في حال مريض لا يحتاج لعلاج

```
           // Check if the patient is at risk
        if (patient.isAtRisk()) {
            System.out.println("This patient is at risk based on recent health
records.");
            if (patient.getAdvices().isEmpty()) {
                System.out.println("No advice available.");
            } else {
                for (Advice advice : patient.getAdvices()) {
                    System.out.println(advice);
                }
            }
            if (patient.getMedications().isEmpty()) {
                System.out.println("No medications prescribed.");
            } else {
                for (Medication medication : patient.getMedications()) {
                    System.out.println(medication);
                }
            }
        }

        else {
                    System.out.println("Patient Don't need any Medicatuions or
Adivces");
             }
    }
```

يتم استخدام Scanner لقراءة المدخلات من المستخدم عبر الكونسل. وستُستخدم هذه الكائنات لقراءة البيانات من المستخدم مثل اسم المريض، العمر، مدينة الإقامة، وكذلك سجلات الصحة (ضغط الدم ومعدل ضربات القلب)

```
        Scanner scanner = new Scanner(System.in);
```

يتم إدخال تفاصيل المريض الأول (الاسم، العمر، المهنة، المدينة) باستخدام Scanner

```
                // Create a Patient 1
        System.out.println("Enter details for Patient 1:");
        System.out.print("Name: ");
        String name1 = scanner.nextLine();
        System.out.print("Age: ");
        int age1 = scanner.nextInt();
        scanner.nextLine();  // consume the newline
        System.out.print("Workplace: ");
        String Workplace1 = scanner.nextLine();
```

```
        System.out.print("City: ");
        String city1 = scanner.nextLine();
```

يتم إنشاء كائن من فئة Patient باسم patient1 ويتم اسناد القيم (الاسم والعمر..) التي تم ادخالها في ال constructor الخاص بكائن المريض الاول

```
        Patient patient1 = new Patient(name1, age1, workplace1, city1);
```

يُطلب من المستخدم إدخال قيم ضغط الدم (الانقباضي و الانبساطي) للمريض الأول ويتم إدخال هذه القيم باستخدام scanner.nextInt()

بعد ذلك، يتم إنشاء كائن باسم bp1 من كلاس BloodPressure يمثل ضغط الدم ويتم اسناد قيم ضغطي الدم (الانقباضي والانبساطي) المُدخلة في constructor خاص به

ويقوم المستخدم بإدخال قيم جديدة لضغط الدم (الانقباضي و الانبساطي) للمريض الأول ويتم إدخال هذه القيم باستخدام scanner.nextInt()

فيتم إنشاء كائن جديد باسم bp2 من كلاس BloodPressure يمثل ضغط الدم ويتم اسناد قيم ضغطي الدم الجديدة (الانقباضي والانبساطي) المُدخلة في constructor خاص به

```
        // Blood Pressure records for Patient 1
        System.out.println("Enter Blood Pressure records for Patient 1:");
        System.out.print("Systolic (BP1): ");
        int systolic1 = scanner.nextInt();
        System.out.print("Diastolic (BP1): ");
        int diastolic1 = scanner.nextInt();
        BloodPressure bp1 = new BloodPressure(systolic1, diastolic1);

        System.out.print("Systolic (BP2): ");
        int systolic2 = scanner.nextInt();
        System.out.print("Diastolic (BP2): ");
        int diastolic2 = scanner.nextInt();
        BloodPressure bp2 = new BloodPressure(systolic2, diastolic2);
```

يتم إضافة سجلات ضغط الدم التي تم إدخالها (bp1 , bp2) إلى كائن patient1 باستخدام دالة addBloodPressureRecord() الى مصفوفة سجلات الدم الخاصة بالمريض الاول

```
        patient1.addBloodPressureRecord(bp1);
        patient1.addBloodPressureRecord(bp2);
```

تكرر العملية في ادخال معدل ضربات القلب حيث يعاد الطلب من المستخدم إدخال قيمة معدل ضربات للمريض الأول ويتم إدخال هذه القيم باستخدام scanner.nextInt()

وينشئ كائن باسم heartRate1 من كلاس HeartRate يمثل معدل ضربات قلب ويتم اسناد قيمة معدل ضربات القلب المُدخلة في constructor خاص به

فيقوم المستخدم مجددا بإدخال قيمة جديدة لمعدل ضربات القلب للمريض الأول ويتم إدخال هذه القيمة الجديدة باستخدام scanner.nextInt()

وينشئ كائن جديد مجددا باسم heartRate2 من كلاس HeartRate يمثل معدل ضربات القلب ويتم اسناد قيمة معدل ضربات القلب المُدخلة جديداً في constructor خاص به

```
        // Heart Rate records for Patient 1
        System.out.println("Enter Heart Rate records for Patient 1:");
        System.out.print("Heart Rate (HR1): ");
        int hr1 = scanner.nextInt();
        HeartRate heartRate1 = new HeartRate(hr1);

        System.out.print("Heart Rate (HR2): ");
        int hr2 = scanner.nextInt();
        HeartRate heartRate2 = new HeartRate(hr2);
```

ويتم إضافة سجلات معدل ضربات القلب التي تم إدخالها (heartRate1 , heartRatw2) إلى كائن patient1 باستخدام دالة addBloodPressureRecord() الى مصفوفة سجلات معدل ضربات القلب الخاصة بالمريض الاول

```
        patient1.addHeartRateRecord(heartRate1);
        patient1.addHeartRateRecord(heartRate2);
```

يتم إدخال تفاصيل المريض الثاني (الاسم، العمر، المهنة، المدينة) باستخدام Scanner

```
           // Create a Patient 2
        scanner.nextLine(); // consume the newline
        System.out.println("\nEnter details for Patient 2:");
        System.out.print("Name: ");
        String name2 = scanner.nextLine();
        System.out.print("Age: ");
        int age2 = scanner.nextInt();
        scanner.nextLine();  // consume the newline
        System.out.print("Workplace: ");
        String workplace2 = scanner.nextLine();
        System.out.print("City: ");
        String city2 = scanner.nextLine();
```

يتم إنشاء كائن ثاني من فئة Patient باسم patient2 ويتم اسناد القيم (الاسم والعمر..) التي تم ادخالها في ال constructor الخاص بكائن المريض الثاني

```
    Patient patient2 = new Patient(name2, age2, Workplace2, city2);
```

يُطلب من المستخدم إدخال قيم ضغط الدم (الانقباضي و الانبساطي) للمريض الثاني ويتم إدخال هذه القيم باستخدام scanner.nextInt()

بعد ذلك، يتم إنشاء كائن باسم bp3 من كلاس BloodPressure يمثل ضغط الدم ويتم اسناد قيم ضغطي الدم (الانقباضي والانبساطي) المُدخلة في constructor خاص به

ويقوم المستخدم بإدخال قيم جديدة لضغط الدم (الانقباضي و الانبساطي) للمريض الثاني ويتم إدخال هذه القيم باستخدام scanner.nextInt()

فيتم إنشاء كائن جديد باسم bp4 من كلاس BloodPressure يمثل ضغط الدم ويتم اسناد قيم ضغطي الدم الجديدة (الانقباضي والانبساطي) المُدخلة في constructor خاص به

```
        // Blood Pressure records for Patient 2
        System.out.println("Enter Blood Pressure records for Patient 2:");
        System.out.print("Systolic (BP1): ");
        int systolic3 = scanner.nextInt();
        System.out.print("Diastolic (BP1): ");
        int diastolic3 = scanner.nextInt();
        BloodPressure bp3 = new BloodPressure(systolic3, diastolic3);

        System.out.print("Systolic (BP2): ");
        int systolic4 = scanner.nextInt();
        System.out.print("Diastolic (BP2): ");
        int diastolic4 = scanner.nextInt();
        BloodPressure bp4 = new BloodPressure(systolic4, diastolic4);
```

يتم إضافة سجلات ضغط الدم التي تم إدخالها (bp4 , bp3) إلى كائن patient2 باستخدام دالة addBloodPressureRecord() الى مصفوفة سجلات الدم الخاصة بالمريض الثاني

```
        patient2.addBloodPressureRecord(bp3);
        patient2.addBloodPressureRecord(bp4);
```

تكرر العملية في ادخال معدل ضربات القلب حيث يعاد الطلب من المستخدم إدخال قيمة معدل ضربات للمريض الثاني ويتم إدخال هذه القيم باستخدام scanner.nextInt()

وينشئ كائن باسم heartRate3 من كلاس HeartRate يمثل معدل ضربات قلب ويتم اسناد قيمة معدل ضربات القلب المُدخلة في constructor خاص به

فيقوم المستخدم مجددا بإدخال قيمة جديدة لمعدل ضربات القلب للمريض الثاني ويتم إدخال هذه القيمة الجديدة باستخدام scanner.nextInt()

وينشئ كائن جديد مجددا باسم heartRate4 من كلاس HeartRate يمثل معدل ضربات القلب ويتم اسناد قيمة معدل ضربات القلب المُدخلة جديداً في constructor خاص به

```
           // Heart Rate records for Patient 2
        System.out.println("Enter Heart Rate records for Patient 2:");
        System.out.print("Heart Rate (HR1): ");
        int hr3 = scanner.nextInt();
        HeartRate heartRate3 = new HeartRate(hr3);

        System.out.print("Heart Rate (HR2): ");
        int hr4 = scanner.nextInt();
        HeartRate heartRate4 = new HeartRate(hr4);
```

ويتم إضافة سجلات معدل ضربات القلب التي تم إدخالها (heartRate3 , heartRatw4) إلى كائن patient2 باستخدام دالة addBloodPressureRecord() الى مصفوفة سجلات معدل ضربات القلب الخاصة بالمريض الثاني

```
         patient2.addHeartRateRecord(heartRate3);
        patient2.addHeartRateRecord(heartRate4);
```

يتم إنشاء كائنات النصائح (Advice) والأدوية (Medication) للحالات المختلفة ليتم اضافتها في عناصر المصفوفات :

نصائح للحالات العالية (ارتفاع الضغط - ارتفاع معدل نبضات القلب)

```
        // Add Advice for high conditions
        Advice highBpAdvice = new Advice("High Blood Pressure");
        highBpAdvice.addAdvice("Reduce salt intake.");
        highBpAdvice.addAdvice("Exercise regularly.");

        Advice highHrAdvice = new Advice("High Heart Rate");
        highHrAdvice.addAdvice("Avoid caffeine.");
        highHrAdvice.addAdvice("Practice stress-relief techniques.");
```

ادوية للحالات العالية (ارتفاع الضغط - ارتفاع معدل نبضات القلب)

```
        // Add Medications for high conditions
        Medication highBpMedication = new Medication("High Blood Pressure");
        highBpMedication.addMedication("Amlodipine");
        highBpMedication.addMedication("Lisinopril");

        Medication highHrMedication = new Medication("High Heart Rate");
        highHrMedication.addMedication("Beta-blockers");
```

ومرة اخرى يتم إنشاء كائنات النصائح (Advice) والأدوية (Medication) ليتم اضافتها في عناصر المصفوفات :

نصائح للحالات المنخفضة (انخفاض الضغط - انخفاض معدل نبضات القلب)

```
        // Add Advice for low conditions
        Advice lowBpAdvice = new Advice("Low Blood Pressure");
        lowBpAdvice.addAdvice("Increase salt intake.");
        lowBpAdvice.addAdvice("Drink more fluids.");
        lowBpAdvice.addAdvice("Wear compression stockings.");

        Advice lowHrAdvice = new Advice("Low Heart Rate");
        lowHrAdvice.addAdvice("Avoid alcohol.");
        lowHrAdvice.addAdvice("Increase physical activity.");
        lowHrAdvice.addAdvice("Eat small meals more frequently.");
```

ادوية للحالات المنخفضة (انخفاض الضغط - انخفاض معدل نبضات القلب)

```
        // Add Medications for low conditions
        Medication lowBpMedication = new Medication("Low Blood Pressure");
        lowBpMedication.addMedication("Fludrocortisone");
        lowBpMedication.addMedication("Midodrine");

        Medication lowHrMedication = new Medication("Low Heart Rate");
        lowHrMedication.addMedication("Atropine");
        lowHrMedication.addMedication("Isoproterenol");
```

يتم إضافة النصائح والأدوية إلى المريض الاول باستخدام دوال addAdvice و addMedication الخاصة بمصفوفات (سجلاتهِ) الطبية

```
        // Associate advice and medications with patient1
        patient1.addAdvice(highBpAdvice);
        patient1.addAdvice(highHrAdvice);
        patient1.addMedication(highBpMedication);
        patient1.addMedication(highHrMedication);
```

يتم إضافة النصائح والأدوية إلى المريض الثاني باستخدام دوال addAdvice و addMedication الخاصة بمصفوفات (سجلاتهِ) الطبية

```
        // Associate advice and medications with patient2
```

```
        patient2.addAdvice(lowBpAdvice);
        patient2.addAdvice(lowHrAdvice);
        patient2.addMedication(lowBpMedication);
        patient2.addMedication(lowHrMedication);
```

يتم إنشاء كائن عيادة من كلاس Clinic لتمثيل العيادة واسناد اسم اليها ليتم إضافة المرضى إلى العيادة باستخدام دالة addPatient الخاصة بمصفوفة (سجلات) المرضى

```
        // Create a Clinic and add patients
        Clinic clinic = new Clinic("Baghdad Medical Center");
        clinic.addPatient(patient1);
        clinic.addPatient(patient2);
```

يتم عرض معلومات العيادة باستخدامdisplayClinicInfo ، التي تعرض قائمة (عناصر مصفوفة) المرضى ومعلوماتهم الاساسية

```
        // Display clinic information
        clinic.displayClinicInfo();
```

يتم استخدام الدالة displayAdviceAndMedications لعرض النصائح والأدوية للمريض الاول والمريض الثاني

```
        // Display advice and medications for each patient using the new method
        displayAdviceAndMedications(patient1);
        displayAdviceAndMedications(patient2);
```

Output:

```
Clinic: Baghdad Medical Center
Patients in Clinic:
Name: Ali Ahmed , Age: 45 , Workplace: Engineer , City: Baghdad
Name: Nasser Hussan , Age: 25 , Workplace: CyberSceurity , City: Hilla
Advice and Medications for Ali Ahmed:
This patient is at risk based on recent health records.
Risk Type: High Blood Pressure, Advice: [Reduce salt intake., Exercise regularly.]

Risk Type: High Heart Rate, Advice: [Avoid caffeine., Practice stress-relief
techniques.]

Medication for High Blood Pressure:
- Amlodipine
- Lisinopril

Medication for High Heart Rate:
- Beta-blockers

Advice and Medications for Nasser Hussan:
This patient is at risk based on recent health records.
Risk Type: Low Blood Pressure, Advice: [Increase salt intake., Drink more fluids.,
Wear compression stockings.]

Risk Type: Low Heart Rate, Advice: [Avoid alcohol., Increase physical activity., Eat
small meals more frequently.]

Medication for Low Blood Pressure:
- Fludrocortisone
- Midodrine

Medication for Low Heart Rate:
- Atropine
- Isoproterenol
```