🧩 1. تعريف الفئات (Classes)

يوجد في الملف عدة فئات:

- **`MedicalDemo`** (تحتوي على `main()`): هي الفئة الأساسية لتشغيل البرنامج.
- **`Patient`**: تمثل مريضًا، وتحتوي على الخصائص والسلوكيات الخاصة به.
- **`Medication`**: تمثل دواء (كلاس غير قابلة للتعديل – immutable).
- **`ClinicUtils`**: تحتوي على دالة مساعدة static للبحث في قائمة الأدوية.

🔒 2. معدّلات الوصول (Access Modifiers)

- `public`: يمكن الوصول إليه من أي مكان.
- `private`: يمكن الوصول إليه فقط من داخل نفس الفئة.
- `protected`: يمكن الوصول إليه من نفس الحزمة أو الفئات الوريثة.
- **مثال:**
- `private int age;` // لا يمكن الوصول إليه إلا داخل `Patient`
- `protected String patientId;` // يمكن وراثته

⚙️ 3. الـ Getters و Setters (الـAccessors و Mutators)

تُستخدم لتغليف البيانات (Encapsulation):

```
public int getAge() { return age; }
public void setAge(int age) { this.age = age; }
```

🧮 4. القيم الافتراضية (Default Values)

عند إنشاء كائن دون تمرير قيم، يتم تعيين القيم الافتراضية:

- `int` → 0
- `boolean` → false
- `String` أو كائن → null

🏗️ 5. الـConstructors

- **بدون معاملات (no-arg):**
- `public Patient() {`
- `    this.patientId = "UNKNOWN";`
- `}`

- **بمعاملات (parameterized):**
- ```
  public Patient(String id, int age, String email, boolean active) {
      this.patientId = id;
      this.age = age;
      this.email = email;
      this.active = active;
  }
  ```

💡 6. إنشاء كائن من الفئة

```
Patient p1 = new Patient();          // يستخدم constructor الافتراضي
Patient p2 = new Patient("P-102", 50, "mary@health.sa", true);
```

🧍 7. الأعضاء الثابتة والمثيل (Static vs Instance Members)

- **Static**: مشترك بين جميع الكائنات.
- `private static String hospital = "Medical Center";`
- `System.out.println(Patient.getHospital());`
- **Instance**: خاص بكل كائن (مثل العمر أو البريد الإلكتروني).

🧭 8. الكلمة this

تستخدم للإشارة إلى الكائن الحالي:

```
this.age = age;
```

🔐 9. الفئة غير القابلة للتغيير (Immutable Class)

```
final class Medication {
    private final String name;
    private final String dose;
    // القيم ثابتة بعد الإنشاء — لا يوجد set
}
```

🔄 10. تمرير/إرجاع الكائنات

```
Medication found = ClinicUtils.findByName(p1.getMedications(), "Ibuprofen");
```

الدالة `findByName()` ترجع كائن من نوع `Medication` إذا تم العثور عليه.

```

📋 11. ArrayList

قائمة ديناميكية لتخزين الأدوية:

```
p1.addMedication(med1);
p1.setMedication(1, new Medication("Ibuprofen", "200mg"));
p1.removeMedication(0);
System.out.println(p1.medicationCount());
```

🔁 12. Enhanced for loop

حلقة مطوّرة للتكرار عبر عناصر القائمة:

```
for (Medication m : p1.getMedications()) {
    System.out.println("- " + m.getName() + " : " +
m.getDose());
}
```

🧠 أنواع الأسئلة في الملف

1. كتابة سطر كود (مثل `private int x;`)
2. أسئلة اختيار من متعدد (MCQ)
3. إظهار المخرجات (Show Output)
4. تصحيح الأخطاء (Correct the code)