كيفية استخدام هذا الكود:

يمكنك ببساطة نسخ كود كل عملية ولصقه في محرر PlantUML أونلاين (مثل `plantuml.com/plantuml`) لترى المخطط البياني فوراً.

---

1. إنشاء فاتورة شراء من مصنع

الهدف: توضيح الخطوات الكاملة لتسجيل فاتورة شراء، بما في ذلك التعامل مع الأصناف الجديدة والموجودة، وتحديث أرصدة المصنع والمخزون داخل Transaction واحدة لضمان دقة البيانات.

```
@startuml
title إنشاء فاتورة شراء جديدة من مصنع

start
:يفتح المستخدم شاشة فواتير المشتريات ويضغط "فاتورة جديدة";
:يختار المصنع من قائمة ويدخل تاريخ الفاتورة;
:المستخدم يحدد حالة الفاتورة (آجل / نقدي);

repeat
  :المستخدم يضغط "إضافة صنف";
  :يبحث عن الصنف بالاسم;
  if (هل الصنف موجود مسبقاً؟) then (نعم)
    :النظام يجلب آخر سعر شراء والموزع التابع له (للقراءة فقط);
    :النظام يعرض تنبيه: "آخر سعر شراء كان X";
    :المستخدم يدخل السعر الجديد ان اراد تغييره ويدخل والكمية والخصم;
  else (لا, صنف جديد)
    :المستخدم يدخل اسم الصنف الجديد;
    :المستخدم يدخل السعر والكمية والخصم;
    :المستخدم يختار الموزع (الشركة التوزيع**) من قائمة;
  endif
  :إضافة الصنف وبياناته إلى قائمة مؤقتة في الفاتورة الحالية;
repeat while (هل يريد المستخدم إضافة صنف آخر؟) is (نعم)


:المستخدم يضغط على زر "حفظ";

partition "Database Transaction" {
    note right
      كل هذه الخطوات
      تتم كوحدة واحدة
      (إما تنجح كلها أو تفشل كلها)
```

```
    end note
    :1. إنشاء سجل جديد في جدول `Factory_Purchases_Bills`;
        :1.1. واضافة ال money befor  و money after حساب واللى هو عبارة عن
المصنع قبل وبعد العملية;
    :2. لكل صنف** في القائمة المؤقتة**:;
    if (هل الصنف جديد؟) then (نعم)
      :2.1. إنشاء سجل جديد في جدول `Product`;
    endif
    :2.2. إنشاء سجل في `Factory_Purchases_Bill_Items`;
    :2.3. تحديث `Product`: زيادة `available_quantity`;
    :2.4. تحديث `Product`: حساب وتحديث `weighted_average_cost`;

    :3. تحديث `Factories`: زيادة `current_quantity` بإجمالي كمية الفاتورة;
    if (هل الفاتورة "آجل"؟) then (نعم)
      :4. تحديث `Factories`: زيادة `current_balance` بإجمالي قيمة الفاتورة;
    endif
    :5. تحديث `Factories`: تحديث `last_purchase_date`;
}

:عرض رسالة "تم الحفظ بنجاح";
:سؤال المستخدم "هل تريد طباعة الفاتورة؟";

stop
@enduml
```

---

3. تسجيل دفعة سداد لمصنع

الهدف: عملية مالية بسيطة توضح كيف يتم تحديث رصيد المصنع ورصيد الخزنة معاً في Transaction واحدة.

```
@startuml
title تسجيل دفعة سداد لمصنع

start
:يفتح المستخدم شاشة "المدفوعات للمصانع";
:يضغط "دفعة جديدة";
:يختار المصنع من قائمة;
:يدخل المبلغ المدفوع وتاريخ الدفع;
:يختار الخزنة (Wallet) التي تم الدفع منها;
:يضغط "حفظ";

:تحقق: هل المبلغ أكبر من 0؟;
if (هل المبلغ أكبر من 0؟) then (نعم)
  :تحقق: هل المبلغ أقل من أو يساوي الرصيد الحالي للمصنع؟;
```

```
  if (هل المبلغ أقل من أو يساوي رصيد المصنع؟) then (نعم)
    :تحقق: هل المبلغ أقل من أو يساوي رصيد الخزنة؟;
    if (هل المبلغ أقل من أو يساوي رصيد الخزنة؟) then (نعم)
      partition "Database Transaction" {
        :1. إنشاء سجل جديد في `Factory_Pays`;
            :1.1. واضافة ال money befor  و money after حساب واللى هو عبارة عن
المصنع قبل وبعد العملية;
        :2. تحديث `Factories`: خصم المبلغ المدفوع من `current_balance`;
        :3. تحديث `Wallets`: خصم المبلغ المدفوع من `current_balance`;
      }
      :عرض رسالة "تم الحفظ بنجاح";
      :سؤال المستخدم "هل تريد طباعة الإيصال؟";
      stop
    else (لا)
      :عرض رسالة خطأ "المبلغ أكبر من رصيد الخزنة المتاح";
      :عودة إلى شاشة الإدخال;
    endif
  else (لا)
    :عرض رسالة خطأ "المبلغ أكبر من المبلغ المستحق للمصنع";
    :عودة إلى شاشة الإدخال;
  endif
else (لا)
  :عرض رسالة خطأ "المبلغ المدفوع يجب أن يكون أكبر من الصفر";
  :عودة إلى شاشة الإدخال;
endif

stop
@enduml
```

---

3. تسجيل مرتجع بضاعة إلى مصنع

الهدف: توضيح عملية المرتجع التي تؤثر على ثلاثة أجزاء: رصيد المصنع المالي، إجمالي عدد القطع لديه، وكمية المنتج في المخزن.

```
@startuml
title تسجيل مرتجع بضاعة إلى مصنع

start
:يفتح المستخدم شاشة "مرتجعات المصانع";
:يضغط "مرتجع جديد";
:يختار المصنع من قائمة ويدخل التاريخ;
:يضيف الأصناف المرتجعة وكمياتها;
note left
  النظام يجلب سعر آخر عملية شراء
```

```
  لهذا الصنف كسعر للمرتجع من هذا المصنع. ويكون قابل للتعديل.
end note
:النظام يحسب إجمالي قيمة المرتجع;
:يضغط "حفظ";

partition "Database Transaction" {
    :1. إنشاء سجل جديد في `Factory_Returns`;
        :1.1. واضافة ال money befor  و money after حساب واللى هو عبارة عن
المصنع قبل وبعد العملية;
    :2. **لكل صنف** في المرتجع:;
    :2.1. إنشاء سجل في `Factory_Return_Items`;
    :2.2. تحديث `Product`: خصم `available_quantity`;

    :3. تحديث `Factories`: خصم قيمة المرتجع من `current_balance`;
    :4. تحديث `Factories`: خصم كمية المرتجع من `current_quantity`;
}

:عرض رسالة "تم الحفظ بنجاح";
:سؤال المستخدم "هل تريد طباعة إيصال المرتجع؟";

stop
@enduml
```

---

4. إنشاء فاتورة بيع لمكتب

الهدف: توضيح سير العمل الخاص بالمبيعات، والذي يبدأ باختيار الموزع أولاً، ثم فلترة العملاء والمنتجات بناءً عليه، وتحديث حساب العميل المحدد مع هذا الموزع.

```
@startuml
title إنشاء فاتورة بيع جديدة لمكتب

start
:يفتح المستخدم شاشة "المكاتب";
: (الموزع)** قائمة من التوزيع شركة أولاً **يختار;
note right: هذه الخطوة تفلتر كل البيانات التالية

:يضغط على "فاتورة جديدة";
:يختار المكتب (العميل) من قائمة العملاء التابعين للموزع المختار فقط;
:يدخل تاريخ الفاتورة;

repeat
  :المستخدم يضغط "إضافة صنف";
  :يبحث عن الصنف (من ضمن أصناف الموزع المختار فقط);
```

```
  تلقائياً او صفر ان لم يوجد ويكون (`selling_price`) النظام يجلب سعر البيع:
قابل للتعديل ان كان صفر;
  المستخدم يدخل الكمية والخصم:;
  إضافة الصنف إلى قائمة مؤقتة في الفاتورة الحالية:;
repeat while (هل يريد المستخدم إضافة صنف آخر؟) is (نعم)

المستخدم يحدد حالة الفاتورة (آجل / نقدي):;
المستخدم يضغط على زر "حفظ":";

partition "Database Transaction" {
    1. إنشاء سجل جديد في `Customer_Sales_Bills`:;
        1.1. واضافة ال money befor و money after حساب واللى هو عبارة عن
المكتب قبل وبعد العملية;
    2. **لكل صنف** في القائمة المؤقتة:;
    2.1. إنشاء سجل في `Customer_Sales_Bill_Items`:;
    2.2. تحديث `Product`: خصم `available_quantity`;

    3. تحديث `Customer_Distributor_Accounts`: زيادة `current_quantity`:;
    if (هل الفاتورة "آجل"؟) then (نعم)
      4. تحديث `Customer_Distributor_Accounts`: زيادة `current_balance`:;
    endif
}

عرض رسالة "تم الحفظ بنجاح":;
سؤال المستخدم "هل تريد طباعة الفاتورة؟":";

stop
@enduml
```

---

5. تسجيل دفعة تحصيل من مكتب

الهدف: توضح هذه العملية ضرورة تحديد الموزع ليتم الخصم من الحساب الصحيح للعميل، بالإضافة لتحديث رصيد الخزنة.

```
@startuml
title تسجيل دفعة تحصيل من مكتب

start
يفتح المستخدم شاشة "المكاتب":";
يختار أولاً شركة التوزيع (الموزع)** من قائمة** :;
يذهب إلى "تحصيلات المكاتب" ويضغط "تحصيل جديد":";
يختار المكتب (العميل) من قائمة (عملاء هذا الموزع فقط):;
يدخل المبلغ المستلم وتاريخه:;
التي تم إيداع المبلغ فيها (Wallet) يختار الخزنة:;
```

```
:حفظ" يضغط";

partition "Database Transaction" {
    :1. إنشاء سجل جديد في `Customer_Pays`;
        :1.1. واضافة ال money befor  و money after والى هو عبارة عن حساب
المكتب قبل وبعد العملية;
    :2. تحديث `Customer_Distributor_Accounts`: خصم المبلغ من `current_balance`
للحساب المحدد;
    :3. تحديث `Wallets`: زيادة المبلغ في `current_balance` للخزنة المحددة;
    :4. تحديث `Customer_Distributor_Accounts`: تحديث `last_payment_date`;
}

:تم الحفظ بنجاح" عرض رسالة";
:هل تريد طباعة الإيصال؟" سؤال المستخدم";

stop
@enduml
```

---

6. تسجيل مرتجع بضاعة من مكتب

الهدف: توضيح عملية مرتجع العميل التي تؤثر على حسابه لدى موزع معين، وتزيد من كمية المخزون.

```
@startuml
title تسجيل مرتجع بضاعة من مكتب

start
:المكاتب" شاشة المستخدم يفتح";
: من قائمة** (الموزع)** يختار أولاً شركة التوزيع;
:مرتجع جديد" ويضغط "مرتجعات العملاء" يذهب إلى";
:(عملاء هذا الموزع فقط) من قائمة (العميل) يختار المكتب;
:يدخل تاريخ المرتجع;
repeat
  :إضافة صنف" المستخدم يضغط";
  :(من ضمن أصناف الموزع المختار فقط) يبحث عن الصنف;
  :النظام يجلب اخر سعر البيع للمنتج للمكتب ده;
  :المستخدم يدخل الكمية;
  :إضافة الصنف إلى قائمة مؤقتة في الفاتورة الحالية;
repeat while (هل يريد المستخدم إضافة صنف آخر؟) is (نعم)
:(بسعر البيع) النظام يحسب إجمالي قيمة المرتجع;
:حفظ" يضغط";

partition "Database Transaction" {
    :1. إنشاء سجل جديد في `Customer_Returns`;
```

```
        :1.1. واضافة ال money befor  و money after والللى هو عبارة عن حساب
المكتب قبل وبعد العملية;
    :2. **لكل صنف** في المرتجع:;
    :2.1. إنشاء سجل في `Customer_Return_Items`;
    :2.2. تحديث `Product`: زيادة `available_quantity`;

    :3. تحديث `Customer_Distributor_Accounts`: خصم قيمة المرتجع من
`current_balance`;
    :4. تحديث `Customer_Distributor_Accounts`: خصم كمية المرتجع من
`current_quantity`;
}

:تم الحفظ بنجاح "عرض رسالة";
stop
@enduml
```