A. Formulaire de gestion des ventes : description de l'exemple

1. Présentation de l'exemple

Nous sommes l'entreprise **SacEni**, un distributeur qui commercialise des sacs de sports, un sac en taille L, un autre en taille XL. L'activité débute et l'entreprise souhaite se doter d'un simple fichier pour suivre ses ventes et son stock.

L'outil que vous allez mettre en place va permettre au vendeur de cette petite entreprise de créer, stocker et télécharger des factures.

L'outil va se présenter comme un formulaire à remplir par le vendeur. Il sera accessible à partir d'un fichier Excel et du bouton **Accéder à l'outil de gestion des ventes** qui sera positionné sur la feuille d'ouverture du classeur.

Voici l'outil une fois qu'il sera finalisé :

2. Présentation du fichier

Pour réaliser cet exemple, vous allez utiliser le fichier Excel **Enoncé_3-ABC.xlsm**. Le format **XLSM** signifie que c'est un fichier Excel qui prend en charge les macros (permettant l'utilisation de VBA, souvent désactivée par défaut).

Ce fichier contient trois feuilles Excel (appelées *sheets* en Visual Basic).

Feuille Accueil

Cette feuille est destinée à contenir uniquement le bouton **Accéder à l'outil de gestion des ventes.** Il permettra l'accès à l'outil de gestion des ventes.

Initialement, cette feuille est vide.

Feuille Produits

La feuille **Produits** recense les produits vendus par la société. Elle contient trois colonnes :

	Colonne A	Colonne B	Colonne C
Ligne 1	Nom du produit	Prix hors taxe	Quantité disponible
Ligne 2	Sac taille L	20 €	90

	Colonne A	Colonne B	Colonne C
Ligne 3	Sac taille XL	30 €	50

Feuille Factures

Cette feuille contiendra les factures créées avec l'outil de gestion des ventes. Elles seront référencées par numéro et vous trouverez également la date/heure de l'édition, ainsi que le montant.

	Colonne A	Colonne B	Colonne C
Ligne 1	Numéro de facture	Date et heure	Montant de la facture

3. Fonctionnalités

L'objectif est donc d'avoir un outil qui permet de gérer les ventes des sacs de l'entreprise. Voici la retranscription des besoins sous forme d'exigences.

Exigences métiers

Les exigences métiers correspondent à la description des fonctionnalités globales de l'application, c'est le niveau de détail le plus faible :

- Créer une facture
- Tracer la facture
- Mettre à jour les stocks de produits

Retranscription de ces exigences métiers en fonctionnalités

Il s'agit de détailler les exigences métiers en fonctionnalités qui correspondent aux actions utilisateurs et aux traitements du système. Cette liste doit être exhaustive afin de permettre de réaliser ces fonctionnalités sous forme d'application.

- Ajouter une ligne de commande.
  - Choisir la quantité.
  - Choisir le produit.
  - Valider la ligne de commande.
- Afficher la commande.
- Supprimer une ligne de commande.
- Faire le total et permettre l'application d'une remise de 10 %.
- Mettre à jour les stocks.

B. Formulaire de gestion des ventes : notions de cours

Cet exemple contient de nombreux nouveaux concepts liés à la programmation avec le langage Visual Basic... Quelques repères sont donc utiles pour pouvoir démarrer sereinement.

1. Concept de programmation

Voici une description simplifiée de quelques notions de base de programmation permettant une meilleure approche des exemples proposés.

Objet et classe

Un **objet** est une entité informatique, il peut être de toute forme et chaque objet est unique. Il est caractérisé selon son type.

La **classe** correspond à la définition de l'objet, elle servira de canevas pour la création de nouveaux objets. Par conséquent, tous les objets d'une même classe auront les mêmes propriétés, ils se différencieront par les valeurs de leurs propriétés.

Exemple :

Classe	Objets
Cell (cellule d'une feuille Excel)	▸ Cells("A1") : Cellule A1 de la feuille en cours ; ▸ Cells("C4") : Cellule C4 de la feuille en cours.
Sheet (feuille de classeur)	▸ Sheets(0) : première feuille du classeur en cours ; ▸ Sheets(1) : deuxième feuille du classeur en cours.
ThisWorkbook (ce classeur dans lequel nous écrivons le code VBA)	▸ ThisWorkbook.Sheets(0).Cells("A1") : cellule A1 de la feuille de calcul indice 0 de ce classeur.
Textbox (zone de texte saisissable dans un formulaire)	▸ Textbox1 : zone de texte saisissable nommée Textbox1 par l'utilisateur ; ▸ Textbox2 : zone de texte saisissable nommée Textbox2 par l'utilisateur.

Propriétés

Une **propriété** correspond à un attribut d'une classe. Lorsqu'un objet est créé, il a donc des valeurs assignées à ses propriétés.

Exemple : la cellule d'une feuille comporte de nombreuses propriétés, comme par exemple la valeur : `Cells("A1").value`

Méthode

Une **méthode** correspond à une action qui peut être réalisée par un objet. Par exemple, l'objet feuille de calcul (`Sheets`) propose une méthode `Add` qui permet d'ajouter une feuille.

Exemple : Sheets.Add

Collections

Une **collection** est une liste d'objets d'une même classe. Par exemple, la collection `Sheets` correspond à l'ensemble des feuilles. En Visual Basic, les collections sont des objets à part entière avec leurs propres méthodes et propriétés.

Variables

Une **variable** est une entité informatique qui permet de stocker des informations au sein de l'application, elle se déclare de la manière suivante :

- `Dim` : permet de définir la variable (`Public` pour une variable publique) ;
- `Nom_variable` : permet de donner un nom à la variable ;
- `As TypeVariable` : permet de typer la variable.

Exemple :

```
Dim MaVariable As String
```

Cela signifie que la variable `MaVariable` est déclarée en tant que chaîne de caractères.

Les variables sont les suivantes :

- Publiques : elles sont accessibles sur l'ensemble de l'application. Elles sont déclarées en dehors de toute procédure de code.
- Privées : elles sont accessibles uniquement dans la procédure où elles sont déclarées (sur une procédure donnée).

Les variables sont typées principalement pour les trois motifs suivants :

- Cela permet d'avoir des méthodes (voir précédemment) adaptées à la variable : une addition de chaînes de caractères correspond à la concaténation alors qu'une addition de nombres correspond à la somme des valeurs :

Opérations	Valeur de la variable chaîne de caractères	Valeur de la variable nombre entier
`MaVar = "A" + "E"`	`"AE"`	`Erreur`
`MaVar = 1 + 2`	`"12"`	`3`

- Cela facilite le développement et l'usage de variable, le contenu de la variable est attendu.
- Chaque type de variable a une quantité de mémoire allouée, par conséquent, utiliser le bon type de variable permet d'économiser de la mémoire.

Voici les types de variable et le détail de chacune :

Nom	Type	Détails
Byte	Numérique	Nombre entier de 0 à 255
Integer	Numérique	Nombre entier de -32'768 à 32'767
Long	Numérique	Nombre entier de - 2'147'483'648 à 2'147'483'647
Currency	Numérique	Nombre à décimale fixe de -922'337'203'685'477.5808 à 922'337'203'685'477.5807
Single	Numérique	Nombre à virgule flottante de -3.402823E38 à 3.402823E38
Double	Numérique	Nombre à virgule flottante de -1.79769313486232D308 à 1.79769313486232D308
String	Texte	Texte
Date	Date	Date et heure
Boolean	Boolean	True (vrai) ou False (faux)
Object	Objet	Objet Microsoft (exemple cellule, plage de cellule, feuille)
Variant	Tous	Valeur par défaut si non déclarée

2. Concept de formulaire

Formulaire

Un **formulaire** (dit *form*) est une fenêtre d'interaction entre l'utilisateur et le système. Il s'agit d'une interface visuelle permettant de restituer et/ou collecter de l'information dans le but de faire des traitements.

L'objet formulaire est le contenant des autres objets visuels. Cela signifie qu'il comporte d'autres objets visuels, ceux-là même proposés dans la boîte à outils.

Une même application peut contenir plusieurs formulaires.

Les contrôles

Un formulaire est un contenant de contrôles. Ces **contrôles** sont des objets visuels qui permettent l'interaction avec l'utilisateur.

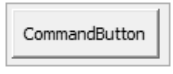
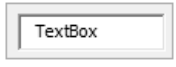
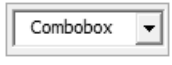
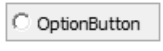
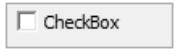
Par défaut, une quinzaine de contrôles sont proposés dans la boîte à outils, et dans le cadre de cet exemple, sept d'entre eux seront détaillés. Toutefois, il est possible d'importer de nouveaux contrôles, mais il s'agit d'une utilisation plus avancée de l'application VBA.

Un contrôle est créé lorsqu'il est déposé sur un formulaire. Un formulaire peut contenir plusieurs contrôles du même type, les contrôles étant des objets, ceux-ci sont uniques.

La propriété « *Name* » (nom) de chaque contrôle doit être unique au sein d'un même formulaire. Il est en revanche possible de trouver un contrôle avec la même valeur pour la propriété *Name* dans un autre formulaire :

```
' Un objet Control1 positionné sur le Formulaire1
Formulaire1.Control1
' Un objet Control1 positionné sur le Formulaire2
Formulaire2.Control1
```

Voici quelques types de contrôles et leur utilisation :

Nom de l'objet	Icône dans la boîte à outil	Affichage sur un formulaire	Utilisation
`CommandButton`			Bouton de formulaire généralement déclencheur d'actions.
`Label`			Zone de texte non saisissable par l'utilisateur : elle peut être modifiée par le système.
`Textbox`			Zone de texte saisissable par l'utilisateur.
`ComboBox`			Liste déroulante avec la possibilité de choisir une seule valeur.
`OptionButton`			Bouton radio permettant de sélectionner une valeur dans un groupe : usuellement, il n'est pas utilisé seul (exemple : homme ou femme).
`CheckBox`			Case à cocher permettant de sélectionner une valeur ou plus dans un groupe (exemple : sélection de loisirs).

Nom de l'objet	Icône dans la boîte à outil	Affichage sur un formulaire	Utilisation
`ListBox`			Il s'agit d'une liste avec la possibilité de voir la liste complète à la différence d'un contrôle ComboBox où seule une valeur est affichée. De plus, cette liste permet de sélectionner plusieurs valeurs (à paramétrer dans les attributs de l'objet).

3. Rédaction du code

Cette sous-partie va vous apprendre les instructions de base pour coder. Où, quand et comment écrire du code ?

Module

Un **module** est une feuille dans laquelle il est possible d'écrire du code. Les modules peuvent contenir plusieurs procédures.

Procédure

Une **procédure** représente une portion de code nommée par un titre. Une procédure peut être appelée à tout moment dans l'application par l'instruction `Call`.

Elle peut avoir des paramètres en entrée appelés **arguments**. Ceux-ci peuvent être facultatifs.

Une procédure commence par l'instruction `Sub` avec le nom de la procédure puis termine par l'instruction `End Sub`. Par défaut la portée de celle-ci est publique, cela signifie qu'elle est visible (et donc qu'il est possible de l'appeler) à tout endroit dans l'application. Toutefois il est possible de spécifier la portée de la procédure en écrivant `Public` ou `Private` (privée) avant l'instruction `Sub`. L'instruction `Private Sub` limitera la visibilité de la procédure au module où elle est déclarée.

Voici un exemple de procédure affichant le produit d'un calcul au sein d'un module.

```
Public Sub MaProcedure()
'Définition de mes variables A et B en tant que nombre entier
Dim A As Integer
Dim B As Integer
'Affectation des valeurs aux variables
A = 3
B = 4
'Appel à la procédure 'Calculer' avec les 2 arguments A et B définie ci-après
Call Calculer(A, B)
```

```
End Sub

Public Sub Calculer(Valeur1 As Integer, Valeur2 As Integer)
'Définition de la variable Produit qui représentera le calcul
Dim Produit As Integer
Produit = Valeur1 * Valeur2
'Affichage dans une pop-up de la valeur de la variable Produit
MsgBox (Produit)
End Sub
```

Résultat de l'exécution de la procédure MaProcedure :

Fonction

À l'instar de la procédure, la **fonction** est une portion de code qui possède sa portée et qui est positionnée dans un module. La fonction peut également avoir des arguments en entrée.

La fonction, contrairement à la procédure, dispose d'une valeur de retour. La valeur produite est stockée dans une variable portant le nom de la fonction.

Une fonction débute avec l'instruction `Function` et se termine par l'instruction `End Function`.

La fonction doit assigner une valeur en tant que résultat.

Voici le même exemple que précédemment mais avec l'utilisation d'une fonction :

```
Public Sub MaProcedureFunction()
'Définition de mes variables A et B en tant que nombre entier
Dim A As Integer
Dim B As Integer
'Affectation des valeurs aux variables
A = 3
B = 4
'Appel à la fonction FN_Calculer avec les 2 arguments A et B
MsgBox (FN_Calculer(A, B))
End Sub

Public Function FN_Calculer(Valeur1 As Integer, Valeur2 As Integer)
Dim Produit As Integer
Produit = Valeur1 * Valeur2
'Affectation de la valeur à la fonction
FN_Calculer = Produit
End Function
```