CONSIGNES 1 : LE CATALOGUE

ÉTAPE 1 : CRÉATION DE LA BD

Créez votre BD et les deux tables nécessaires (vos produits et les catégories de produits). Insérez les données minimales.

Votre page http://420.cstj.qc.ca/prenomnom/**index.php** affichera votre catalogue de produits. Chaque item de ce catalogue correspondra à un enregistrement de votre table produit.

ÉTAPE 2 : ISOLER L'EN-TÊTE ET LA COLONNE DE GAUCHE

Minimalement, l'index.php devrait comporter un en-tête, une section de gauche, une section principale. Je suggère que vous utilisiez les <section> <article> pour structurer l'apparence de votre site. En cas de doute, inspirez-vous du site de démonstration.

L'en-tête ainsi que la section gauche vous serviront dans toutes vos autres pages. Il serait avisé de les coder dans des fonctions accessibles à toutes les pages PHP que vous afficherez. Mettez le code de ces fonctions dans un fichier contenant les fonctions communes. Il sera accessible par des **require_once("nomDeVotreFichier.php").**

ÉTAPE 3: LE CŒUR DE LA PAGE : LE CATALOGUE D'ARTICLES

Créez deux nouvelles classes php contribueront à la clarté de l'organisation du catalogue.

LA CLASSE PRODUIT

Les produits sont les constituants de base du catalogue. On va donc créer une classe **Produit** (ou un autre nom plus adapté à ce que vous vendez comme **Livre** ou **Moto** par exemple)**.** Elle fera le lien entre la bd et les scripts PHP de votre site.

En plus des attributs provenant de la table produit, il serait intéressant d'ajouter un attribut pour la catégorie, même si elle provient d'une autre table.

La classe proposée ne contiendra que deux méthodes : le constructeur et une méthode pour afficher un produit.

LE CONSTRUCTEUR

Le constructeur a pour but de mettre dans les attributs d'un produit les valeurs réelles obtenues de la bd. Voyons deux façons de le concevoir.

La première façon (**ne l'utilisez pas**) recevrait simplement en paramètres toutes les valeurs qu'on veut conserver dans l'objet instancié. Par exemple :

```
public function __construct ($noFilm, $titre, $description…) {
   $this->noFilm = $noFilm;
   $this->titre = $titre;
   ...
}
```

L'énoncé qui l'instancierait serait du genre

```
$produit = new Produit($numero, $nomproduit, … );
```

en supposant que les paramètres ont été obtenus de la base de données.

La seconde façon, **que je recommande**, utilise un tableau qu'on aura construit avant d'appeler l'instanciation. Comme dans l'exemple plus bas :

```
public function __construct ($tableau) {
   $this->noFilm = $tableau['noFilm'];
   $this->titre = $tableau['titre'];
   ...
}
```

Même si vous n'êtes pas à l'aise avec les tableaux pour le moment, utilisez quand même cette façon de faire. Vous constaterez sous peu qu'elle se révèle la plus facile.

LA MÉTHODE D'AFFICHAGE DU PRODUIT

Pour le moment, contentez-vous, dans cette méthode, d'afficher les parties du produit qui apparaissent dans le catalogue (numéro, nom du produit, prix, etc.).

Comme le résultat est affiché dans une page Web, vos énoncés seront du genre

```
echo "$this->titre";
echo number_format($this->prix,2),"\$";
```

Remarquez la fonction **number_format(…,2)** qui permet d'assurer qu'on aura deux chiffres après le point quand on affiche des montants. Utilisez-la **toujours** pour afficher des valeurs monétaires. Notez aussi que puisque le **$** est réservé en PHP pour les noms de variables, il faut le précéder d'une oblique inverse si on veut l'afficher entre des **guillemets**. Entre apostrophes, il ne faut pas mettre d'oblique inverse.

Sauvegardez cette classe dans un fichier dont le nom est identique à la classe(si elle s'appelle **Produit**, nommez le fichier **Produit.php**) et rangez-la dans votre répertoire **Classes**.

Avant d'aller plus loin, prenez le temps d'écrire un petit programme pour tester votre classe produit. Cela vous évitera des pertes de temps importantes. il suffira de vérifier si ses deux méthodes sont fonctionnelles.

Pour le site de démonstration, le programme de test de la classe ressemble à ceci :

```
<?php
require_once "Classes/film.php";
$tableau['noFilm']='10';
$tableau['titre']='Un film pour tester ma classe';
$tableau['noCategorie']='1';
$tableau['categorie']='Horreur';
$tableau['realisateur']='Alain Martel';
$tableau['prix']='12.98';
$tableau['qteStock']='7';
$tableau['qteCommande']='2';
$tableau['descLongue']='Ce film sert à tester si la classe Films utilise'.
                       'correctement son constructeur et affiche';
$essai = new Film ($tableau);
$essai->affiche();
echo '<br/>';
?>
```

Rédigez quelque chose d'équivalent pour votre classe et assurez-vous que cela donne un affichage conforme à vos attentes avant d'aller plus loin.

LA CLASSE CATALOGUE

Le catalogue est une classe de type « composition » : il servira à traiter un **tableau** de **produits**. Cette classe disposera:

- de son constructeur,
- d'une méthode pour lui ajouter un produit
- d'une méthode pour afficher le catalogue.

Il n'y a qu'un seul attribut déclaré ainsi :

```
  private $catalogue = array();
```

Cette déclaration indique que **$catalogue** sera un tableau, vide au départ.

L'AJOUT D'UN PRODUIT AU CATALOGUE

La méthode d'ajout de produit sera simplement :

```
  private function ajouterProduit($prod) {
     $this->catalogue[]=$prod;
  }
```

PHP est d'une simplicité remarquable quand vient le temps d'ajouter une valeur à la fin d'un tableau : il suffit de ne pas mettre d'indice entre les crochets du tableau à gauche du signe **=** et l'affectation se fait toute seule au bon endroit.

LE CONSTRUCTEUR

Celui-ci sera plus complexe. Pour l'instant, il chargera en mémoire l'ensemble des produits du catalogue. Plus tard, vous verrez comment le rendre plus « sélectif » pour répondre aux demandes du client.

Il doit envoyer une requête à la BD pour obtenir les produits et les ajouter un à un au tableau.

Sa version initiale ressemblera à ceci :

```
  public function __construct() {
   global $bd; //L'objet $bd a été instancié à l'extérieur de cette fonction, « global » le rend disponible
   $strSQL="SELECT noProduit, nomProduit, ... FROM produits, categories
                                          WHERE produits.noCategorie = categories.noCategorie ";
   $resultat = $bd->Select($strSQL);
   foreach ($resultat as $ligne) {
     $produit = new Produit ($ligne);
     $this->ajouterProduit($produit);
   }
}
```

Comme le constructeur du **produit** attend un tableau, le fait de passer **$ligne** comme paramètre convient tout à fait. À vous de vous assurer que **$ligne** contient tous les éléments requis par le constructeur. Sachez tout de même que **$ligne** contient chacun des éléments identifiés dans l'énoncé **SELECT** envoyé à la BD.

L'AFFICHAGE DU CATALOGUE

Vous avez écrit, dans la classe **Produit**, une méthode d'affichage pour un produit et vous l'avez testée. Il vous reste à vous en servir ici.

Rappelez-vous que chaque élément du tableau **$catalogue** est un objet de type **Produit**. Chacun donne donc accès à la méthode qui permet d'en afficher le contenu.

Question de contrôler l'affichage quand il n'y a aucun produit — cela arrivera quand on ajoutera des critères de recherche et de sélection — on a intérêt à connaître le nombre de produits du catalogue.

La méthode ressemblera à ceci :

```
public function affiche() {
  $nbProduits = count($this->catalogue);
  if ($nbProduits == 0)
       afficher un message indiquant que rien ne satisfait aux critères
  else {
     for ($i=0; $i < $nbProduits; $i++)
        $this->catalogue[$i]->affiche();  // pour chaque produit du tableau, on fait appel à
                                          // sa méthode d'affichage
  }
}
```

Sauvegardez cette classe, également dans le répertoire **Classes**.

Comme la classe **Catalogue** fait appel à la classe **Produit**, et que la définition de cette dernière doit être accessible quand on fait appel à la classe **Catalogue**, vos **require_once** devront être dans l'ordre suivant :

```
  require_once "Classes/produit.php";
  require_once "Classes/catalogue.php"
```

Il vous reste à appeler dans votre **index.php** les énoncés d'instanciation et d'affichage du catalogue :

```
  $liste = new Catalogue();
  $liste->affiche();
```

En résumé

- La classe **Produit** construit un produit et permet de l'afficher;
- La classe **Catalogue** construit son tableau de produits et permet de les afficher un à un par leur méthode d'affichage;
- Votre **index.php**
  - fait les **require_once** des scripts nécessaires (fonctions et classes);
  - ouvre la connexion à la BD;
  - affiche l'en-tête et la colonne de gauche;
  - affiche le catalogue par les deux lignes de code données plus haut.

Assurez-vous que tout fonctionne avant de passer à la prochaine étape.

ÉTAPE 4 : LES CRITÈRES DE SÉLECTION DES PRODUITS

Votre catalogue permettra de faire la sélection par catégorie ou par critère de recherche.

Le code va devenir un peu plus touffu, mettez-y tout de suite les commentaires qui vous permettront de vous y retrouver.

En vous inspirant du site de démonstration et en tenant compte du fait que les critères n'apparaissent que sur la page d'accueil, vous allez ajouter un formulaire dans votre page. Vous pouvez le mettre à gauche — mais distinct de ce qui y est déjà, — ou en haut ou à droite mais assurez-vous qu'il soit visible sans avoir besoin d'utiliser les barres de défilement.

> **Note :** On introduit ici une technique qui va vous servir à plusieurs reprises dans votre site; prenez la peine de bien la comprendre et de demander des explications au besoin.

Dans le HTML de votre page, créez un formulaire dont le paramètre **action** appelle la page **elle-même** :

```
<form action='index.php' method='get'>
```

Construisez la liste déroulante (**elle doit être dynamique pour respecter les exigences de l'énoncé du TP**). Le **SELECT SQL** doit donc retourner le numéro de catégorie et le texte de la catégorie. Au début du **<select>** (celui en HTML), forcez une entrée dont le numéro est zéro et dont le texte est « Toutes » comme ceci :`<option value='0'>Toutes</option>`. Construisez les autres **<option >** par programmation de façon à afficher le texte des catégories en retournant le numéro. Assurez-vous de donner un paramètre **name=** à votre balise **<select>**.

Mettez ensuite un champ pour permettre une recherche. Assurez-vous de lui donner un paramètre **name=**. Vous pouvez mettre un seul bouton **submit.** Fermez votre formulaire par la balise </form>.

Si vous testez cela, vous devriez constater que quand on clique sur le bouton, la page s'affiche à nouveau avec des valeurs dans le **get** qui indiquent ce que vous venez de demander. Vous pourrez constater le résultat attendu sur le site de démonstration. Il reste à traiter ces valeurs en faisant en sorte qu'elles aboutissent dans le constructeur du catalogue et que celui-ci s'adapte à la demande.

QUAND UNE PAGE S'APPELLE ELLE-MÊME

Quand une page s'appelle elle-même, elle doit être conçue pour traiter les cas où elle est appelée d'ailleurs — sans aucune valeur dans le **get** — tout aussi bien que quand, au contraire il y a des paramètres **get**.

Vous allez ajouter un peu de code entre l'affichage du côté gauche et l'appel du catalogue. Ce code a pour but de paramétrer la construction du catalogue selon les demandes du client.

Supposons que le **<select>** (en HTML) porte le nom **listeCategorie**. Pour tenir compte du fait que la page peut-être appelée la première fois sans paramètre, le code suivant ne met de catégorie que si requis.

```
          $categ=0;
          if(isset($_GET['listeCategorie']))
              $categ=$_GET['listeCategorie'];
```

Vous faites de même pour le paramètre de recherche mais en tenant compte qu'il ne contient pas une valeur numérique mais une chaîne.

```
          $recherche='';
          if(isset($_GET['recherche']))
              $recherche=$_GET['recherche'];
```

Modifiez maintenant votre appel de catalogue simplement en lui passant ces deux valeurs en paramètres comme ceci :

```
$liste = new Catalogue($categ, $recherche);
```

MODIFIER LE CONSTRUCTEUR DU CATALOGUE

Il reste à modifier le constructeur :

```
public function __construct($categorie, $critere) {
```

Vous avez déjà un **SELECT** en SQL qui extrait tous les produits. Vous allez le modifier en lui ajoutant des conditions dans le WHERE.

```
    $condition="";
    if($categorie !=0)
        $condition=" AND produits.noCategorie = ".$categorie;
```

Puis on modifie l'énoncé SELECT de base plus bas:

```
    $strSQL= "SELECT noProduit, nomProduit, categorie... FROM produits, categories WHERE films.noCategorie =
categories.noCategorie".$condition;
```

La variable **$condition** contient une chaîne vide si on veut tous les produits et un bout de phrase SQL si on veut seulement une catégorie. En concaténant cette condition au SELECT, on adapte l'extraction selon la demande du client.

Testez avec tous les cas.

Déduisez de cette méthode ce que vous devez faire pour ajouter une recherche par **critère**, c'est-à-dire par la présence d'un mot soit dans le **nom** soit dans la **description** de votre produit. Vous aurez alors besoin du mot clé **LIKE** et vous ajouterez simplement une autre condition à celle que vous avez construite pour la sélection par catégorie.

Si, par exemple, le client demande de rechercher « abc », débrouillez-vous pour que votre **$condition** se termine par l'équivalent pour vous de …
AND (*nomProduit* LIKE '%abc%' OR *descriptionProduit* LIKE '%abc%')

ÉTAPE 5 : L'ILLUSTRATION DU CATALOGUE

Trouvez les images de vos produits sans perdre de temps inutilement. Trouvez-vous aussi une image avec la mention « pas de photo disponible » ou quelque mention du genre.

Je vous suggère ici une méthode pour nommer les images.

- Utilisez vos numéros de produits (la clé de votre table) pour renommer ces images. Si vos produits sont numérotés **1**, **2**, **3**…, assurez-vous d'avoir des images nommées **1.jpg**, **2.jpg**, **3.jpg**, etc. Si vos produits ont pour numéro ***Produit*001**, ***Produit*002**, nommez vos images ***Produit*001.jpg**, ***Produit*002.jpg**. Nommez aussi votre image spéciale **pasImage.jpg**.

  En vous inspirant du code suivant, ajoutez l'affichage des images à votre page. Attention, adaptez le **$chemin** à la réalité de vos répertoires.

```
$chemin="Images/Films/";
$nomImage=$chemin."pasImage.jpg";
if(file_exists($chemin.$this->noProduit.".jpg"))
     $nomImage = $chemin.$this->noProduit.".jpg";
echo "<img src='".$nomImage. "'>";
```

  Un peu de HTML et vous devriez maintenant avoir un catalogue illustré.

ÉTAPE 6 : LE BOUTON D'AJOUT AU PANIER

Pour compléter, aller vous chercher une image de panier ou un bouton **Ajouter au panier** en format .png ou .jpg. Sauvegardez sur votre compte.

En supposant que vous allez bientôt écrire un programme portant le nom de **panierGestion.php** et en vous inspirant du code suivant, complétez votre page en mettant le bouton en question à côté de chaque produit dans la méthode d'affichage du produit.

```
echo "<a href='panierGestion.php?quoiFaire=ajout&noproduit=",$this->noProduit,"'>";
echo "<img src='Images/ajoutPanier.png' border='0'></a>";
```

Question de vérifier le travail, écrivez un programme **panierGestion.php** qui ne fera, pour l'instant, qu'afficher les paramètres « **quoiFaire** » et « **noproduit** » qu'il reçoit, en vue de leur traitement éventuel.

Pour des considérations de sécurité, ces valeurs n'apparaissent pas dans le GET quand on ouvre la page du panier sur le site de démonstration. Les explications viendront avec la série des consignes qui portent sur le panier.

ÉTAPE 7 : LA FENÊTRE INDÉPENDANTE AVEC DESCRIPTION ET AUTRE PHOTO

Pour permettre d'afficher une description plus complète de chacun de vos produits, sur chacune de vos images, vous ajoutez un événement **onClick** qui appelle une fonction Javascript ou encore, vous associez l'image à une balise d'hyperlien (<a>).

Le code ressemblera à

```
echo "<img src='".$nomImage. "' onClick='afficheDetails(".$this->noProduit. ")'/">
```

ou encore

```
echo "<a href=javascript:afficheDetails(".$this->noProduit.") ><img src='".$nomImage. "'/"></a>
```

Dans les deux cas, il s'agit d'appeler une fonction Javascript en lui transmettant en paramètre le numéro du produit courant. La deuxième façon de faire a l'avantage que le curseur passe sous la forme d'une main quand on passe sur l'image et indique à l'utilisateur qu'il y a là un hyperlien.

Consultez le code source du site de démonstration pour voir la suite par rapport au JavaScript. Vous verrez que je transmets à un programme indépendant le numéro de produit correspondant à la photo.

Le programme appelé (**filmsDetails.php** dans le cas du site de démonstration) reçoit donc le numéro du film dans son GET. Il ouvre la BD, va chercher les informations relatives à ce film (ou ce produit) et les affiche.

Dans un contexte orienté objet, ce dernier affichage se fera en instanciant un objet ***Produit*** avec ce que vous lisez depuis la BD puis en appelant une nouvelle méthode **afficheFenetreDetail()**.

Ceci complète votre catalogue.