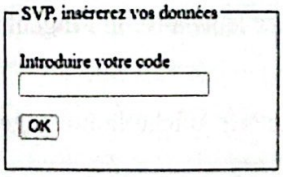
Examen : Programmation web (PHP)

— **La durée de l'examen est 1h :30**
— Document non autorisé/ Fournir une copie claire/ Utiliser le mémorandum distribué avec la copie d'examen pour se rappeler des commandes

Exercice 1 :Questions du cours

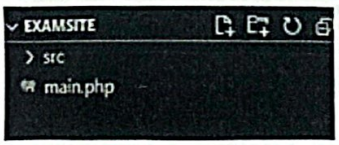
1. Définir brièvement le pattern MVC en déterminant le rôle de chacune de ses parties : Model, View et controller dans une application informatique
2. on suppose qu'on a la page **formcode.php** ayant cette forme, envoie ses informations à la page **traitCode.php**



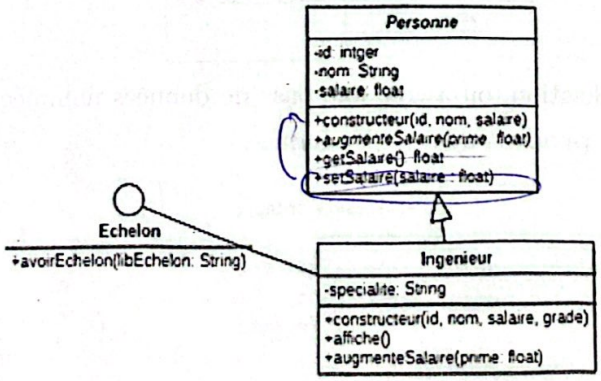
En utilisant les expressions régulières (regex), Ecrire le script PHP necessaire dans la page traitCode.php qui vérifie si le code saisi prend la forme : essai-xx-xxx-xxx avec x est un chiffre qui varie entre 0 et 9. si la sasie respecte cette forme, la page affiche "Code retenu" sinon elle affiche le message "Code erroné" et affiche encore une fois la page formcode.php

Exercice 2 :POO

On suppose que le dossier du site "examSite" est structuré de cette façon :



1. Sous le dossier src (**on suppose qu'on a l'accès**) traduire le diagramme de classe ci-dessous en langage PHP, en respectant les contraintes suivantes :



(a) la classe Personne est une classe abstraite qui contient quatre methodes :
- un constructeur qui initialiser les champs de l'objet courant
- getSalaire qui retourne le salaire de l'objet courant
- affiche() qui affiche le message 'Bonjour Mr nom (champ de l'objet), votre matricule est (champ matricule) et votre salaire est (champ salaire).
- augmentationSalaire($prime) une méthode abstraite

(b) Echelon est une interface qui contient la méthode avoirEchelon(libEchelon)

(c) la classe ingénieur hérite la classe Personne et implémente l'interface Echelon

(d) la classe Ingenieur contient trois méthodes :

i. constructeur qui initialise les champs propres et hérités de l'objet courant

ii. affiche qui appelle la méthode affiche de la classe mères puis affiche le message :votre spécialité est "champ specialite"

iii. augmenteSalaire qui augmente le salaire de l'ingénieur de la valeur de la prime passée en paramètre

(e) la méthode avoirEchelon(libEchelon) affiche le message "Félicitation, vous avez eu l'échelon iibEchelon"

2. dans la page main.php :

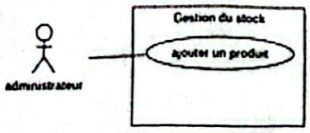
(a) importer toutes les classes et les interfaces qui existent sous le dossier src, utiliser la technique d'autoloading et créer un objet Ingenieur ing01 ayant comme id=100, nom=flen, salaire=1500 et specialite= stat

(b) augmenter le salaire de cet ingénieur de 100 dinars (utiliser la méthode augmenteSalaire) puis afficher ses différentes données (utiliser la méthode affiche)

(c) cet ingénieur a eu l'échelon A+, félicitez -le (utiliser la méthode avoirEchelon)

Exercice 3 : CRUD Application

soit le cas d'utilisation suivant :



Pour réaliser ce cas d'utilisation, on a créé une base de données nommée **test** gérée par le SGBD MySQL qui contient la table **produit** ayant cette forme :

```
create table produit(
    mat int,
    nom varchar(40),
    prix float,
    PRIMARY key(mat)
);
```

— Implémenter ce cas d'utilisation en langage PHP, tout en respectant le pattern MVC

NB :

— la page d'accueil pour cette application doit avoir cette forme :

Ajouter un produit

Faire autre chose

— le formulaire de saisie pour cette application doit prendre cette forme :

Introduire la matricule

Introduire le nom

Introduire le prix

OK

— Les données saisies doivent respectées les contraintes suivantes : la matricule doit avoir des entiers de 4 chiffres, le nom doit contenir que des caractères alphabétiques (la casse n'est pas importante) et le prix doit varier entre 100 et 2500

— utiliser les fonctions built-in PHP de conversion de String vers un entier **intval()** et vers un flottant **floatval()**

Memo PHP /HTML

```
<!doctype html>
<html>
<head>
      <!-- l'entête -->
</head>
<body>
  <!-- Le corps de la page
-->
</body>
</html>
```

```
function nom($par1,..,$parn){
      /* code */
      return $val
}
```

```
<form method="post" action="page.php">

  <label for="lb1">message : </label>
<input type="text" name="nm" id="lb1"/>
<input type="submit" value="Envoyer" />
</form>
```

```
< ?php  /*code*/ ?>
```

```
include(page.php)
```

```
Floatval(String)
```

```
Echo '<br>Messages  '.$variable
```

```
spl_autoload_register(function($name){
   require_once('Chemin'.$name.'.php');
});
```

```
$_SERVER['REQUEST_METHOD']
```

```
require_once(page.php)
```

```
Intval(String)
```

```
$var=  '/expression regulière/'
```

```
preg_match( variableRegex, variable )==Boolean
```

```
String1===String2
```

```
Boolean empty($var)
```

```
abstract public function nomMethode();
```

```
$_POST['param']
```

```
try{

  }
  catch(PDOException $e){
  echo 'Erreur '.$e->getMessage();
    die();
  }
```

```
        echo '<p style="Script css"> message </p>';
```

```
$req = $con->prepare('select requete ???,..,?');
   $req->execute(array($var1,....,$varn));
   $res = $req->fetch();
```

```
public function __construct($param1,....,$paramn)
      { $this->attribut1=$param1;....  }
$symbolesRegex= '/^[]{}|\s.?*$/i'
```

```
Class class1 extends class2 implements interfac{
...
}
```

```
$obj->membre
```

```
Class::membre
```

```
$con= new PDO('mysql:host=localhost;dbname=nom_base;charset=utf8', 'root', '');
```

```
interface nomInterface{…}
```

```
Parent::member
```

```
private static $attribut
```

```
$req=$con->prepare(' commande SQL (??,?)');
      $req=execute(array(var1,var2,…,var n)
                        $con=null;
```

```
Self::member
```

```
<a href="adresse de page"  target="_blank">textCliquable</a>
```


Ens : MANSOUR Sihem