Exercice commande – noté

Renvoyez un fichier PHP à lafabriquedecode@gmail.com. Le sujet du mail sera « Exercice IW3 » et votre fichier sera nommé prenom-nom.php

Vous construisez un outil très simple de règlement de commande. En voici les différents acteurs :

- des produits : téléphone et tablette
- des moyens de paiement : Paypal et Carte
- des commandes
- un gestionnaire de commande

Les produits

Vous vendez deux types de produits : des téléphones et des tablettes. Ces produits ont les caractéristiques suivantes:

- une désignation, de type chaîne de caractères
- un prix, de type décimal
- un type, de type chaîne de caractères (avec deux valeurs possibles : téléphone ou tablette)

Ces produits se conformeront à l'interface suivante:

```
interface ProduitInterface
{
    public function donnerPrix(): float;
}
```

Leur méthode `afficherCaracteristiques` sera une Template Method, elle réalisera l'affichage suivant pour un téléphone :

```
Je suis un type Téléphone
Je suis un téléphone iPhone 20
Je coûte 999.99 €
```

et l'affichage suivant pour une tablette :

```
Je suis un type Tablette
Je suis une tablette  iPad 8
```

Vous voyez que le prix n'est affiché que pour les téléphones. Prévoyez dans cette méthode template l'appel aux méthodes `afficherType`, `afficherDesignation` et `afficherPrix` que vous implémenterez.

La commande

C'est une classe qui sera concrète. Elle gérera des produits stockés dans un tableau nommé `produits`.

Elle comportera un accesseur `donnerProduits` qui renverra ce tableau et une méthode `ajouterProduit`.

Les moyens de paiement

Ces moyens de paiement sont au nombre de deux et sont déterminés par le contenu du tableau superglobal _GET. Cette variable résidant dans _GET a pour nom typePaiement et si elle n'est pas valorisée, elle sera mise par défaut à la valeur 'carte'. Voici les différentes manières de payer :

http://votresite.com/votrescript.php?typePaiement=paypal

http://votresite.com/votrescript.php?typePaiement=carte

http://votresite.com/votrescript.php (équivaut à l'URL précédent)

Ces moyens de paiement nommés PaiementPaypal et PaiementCarte sont des stratégies. Ils doivent être obtenus en invoquant la méthode fabriquer d'une fabrique « hybride » qui renverra ces objets se conformant à l'interface PaiementInterface dont voici le contenu:

```
interface PaiementInterface
{
   public function payer(float $montant): bool;
}
```

Cette fabrique « hybride » se nommera FabriqueDeMoyensDePaiement et sa méthode fabriquer sera statique. Je l'appelle *hybride* car une fabrique traditionnelle ne fabrique qu'un type d'objet alors que la notre instanciera plusieurs classes en formant dynamiquement leur nom (c'est à dire en concaténant « Paiement » avec ce qui est contenu dans la variable typePaiement).

La méthode payer de nos moyens de paiement affichera à l'écran le type de paiement choisi et le montant à payer avant de retourner vrai (on peut supposer qu'en réalité elle retournerait faux si par exemple l'appel à l'API Paypal ou d'une banque venait à échouer).

Le gestionnaire de commande

Nommé GestionnaireCommande, il est composé d'une commande et d'un moyen de paiement et son unique méthode publique sera nommée regler. Cette méthode calculera le prix total des objets de la commande et demandera au moyen de paiement d'effectuer le paiement du montant préalablement déterminé. Ce gestionnaire de commande sera l'objet *contexte* du design pattern Stratégie que vous connaissez.

Voici *une partie seulement* du code client, qui vous aidera à définir vos objets et design patterns :

```
$typePaiement = $_GET['typePaiement'] ...

try {
   ....
} catch (Throwable $exception) {
   echo $exception->getMessage();
}

$telephone = new Telephone('iPhone 20', 999.99);
$telephone->afficherCaracteristiques();

$tablette = new Tablette('iPad 8', 799.99);
```

```
$tablette->afficherCaracteristiques();

$commande = new Commande();
$commande->ajouterProduit($telephone)
$commande->ajouterProduit($tablette);

$gestionnaireCommande = new GestionnaireCommande ...
$gestionnaireCommande->regler();
```