

**Module : M202 – Approche Agile**

**Filière : Développement digital – option web full stack**

**Activité 10.2**

**Objectifs :**

- Rappel des concepts de la POO
- Initiation aux tests unitaires

Soit la classe `salarie.php` dont le code est fourni à la fin de ce document ;

La classe `salarie` est caractérisé par les attributs suivants :

- `matricule`;
- `nomComplet`;
- `salaire`;
- `static tauxCS = 20`;
- `$dateEmbauche`; (sous forme de m/d/y)

1. Ajouter une méthode « `primeAnnuelle` » permettant de calculer la prime de fin d'année en se basant sur le salaire et le nombre d'années d'expériences, en suivant la formule ci-après :

**$\text{Salaire} \times 0.8 + 100 \times \text{nombre d'années d'expériences}$**

2. Ecrire le code permettant de tester les méthodes :

- a) `setMatricule`
- b) `getDateEmbauche`
- c) `experience`
- d) `calculerSalaireNet`
- e) `primeAnnuelle`

3. Analyse de la qualité du code et de la couverture du code :

- Générer la configuration de `phpunit` ;
- A l'aide de `phpunit` et `xdebug`, générez les rapports d'analyse des tests ;
- Créer le fichier « `sonar-project.properties` » dans la racine du projet et configurer les différentes paramètres concernant la clé du projet ainsi que les rapports d'analyse ;
- Créer un nouveau projet sous Sonarqube (avec la même clé du projet fournie) ;
- Récupérer la commande `sonar-scanner` permettant de lancer l'analyse ;

- Lancer le scan en exécutant la commande ;
- Consulter les résultats sous Sonarqube.

La classe « salarié » :

```
<?php
use DateTime;

class salarie
{
    private int $matricule;
    private string $nomComplet;
    private float $salaire;
    public static float $tauxCS = 20;
    private DateTime $dateEmbauche;

    public function __construct(int $matricule = 0, string $nomComplet = "",
float $salaire = 0, $dateEmbauche = "")
    {
        $this->matricule = $matricule;
        $this->nomComplet = $nomComplet;
        $this->salaire = $salaire;
        // $dateEmbauche est une chaine sous forme de m/d/y
        $this->dateEmbauche = $dateEmbauche != "" ? new
DateTime($dateEmbauche) : new DateTime("now");
    }
    //
    public function setMatricule(int $matricule): void
    {
        if (!preg_match("/^\d{3,7}$/", $matricule))
            throw new Exception("Matricule invalide! ");
        $this->matricule = $matricule;
    }

    public function getMatricule(): int
    {
        return $this->matricule;
    }

    ///
    public function __toString()
    {
        return "Salarié : Matricule: $this->matricule, Nom complet: $this-
>nomComplet, Salaire: $this->salaire <br>";
    }
}
```

```

public function experience()
{
    $today = new DateTime("now");
    $difference = $this->dateEmbauche->diff($today);
    return $difference->format("%y");
}
//
public function calculerSalaireNet(): float
{
    return $this->salaire - ($this->salaire * self::$tauxCS / 100);
}

/**
 * Get the value of dateEmbauche
 */
public function getDateEmbauche()
{
    return $this->dateEmbauche;
}
}

```