# Exercice Java (10 points)

Une Clio a un moteur diesel, lorsqu’on démarre la voiture, le moteur fait « vroum ».

Une Mégane a un moteur électrique, il fait « bzzz ».

L’Usine créé une voiture à partir du nom du modèle (on pourra remplacer la chaîne de caractère en entrée par un énuméré, pour être plus « propre »).

On souhaite obtenir l’affichage suivant :

Je démarre ma Clio.

Vroum

Je démarre ma Mégane.

Bzzzz

A partir du code Java suivant :

**public class** Test **{**

**public static void main(**String[] args**) {**

Usine.créerVoiture("CLIO").démarrer();

Usine.créerVoiture("MEGANE").démarrer();

**}**

**}**

1. Coder en java les classes et méthodes nécéssaires (en privilégiant la lisibilité et maintenabilité);
2. Ecrire le diagramme de classe.

# Exercice SQL (4 points)

Les questions suivantes sont basées sur les tables ci-dessous :

(Un développeur livre un correctif pour un client)

DEVELOPPEUR

|  |  |  |
| --- | --- | --- |
| ID | Nom | Salaire |
| 1 | Guillaume | 20000 |
| 2 | Vincent | 30000 |
| 3 | Thomas | 40000 |
| 4 | Julie | 100000 |

CLIENT

|  |  |  |
| --- | --- | --- |
| ID | Nom | Ville |
| 1 | UEM | Metz |
| 2 | ES | Strasbourg |
| 3 | EDF | Paris |
| 4 | GEG | Grenoble |

CORRECTIF

|  |  |  |
| --- | --- | --- |
| DEVELOPPEUR\_ID | CLIENT\_ID | DATE |
| 2 | 3 | 1/10/2015 |
| 2 | 4 | 2/10/2015 |
| 2 | 1 | 3/10/2015 |
| 3 | 1 | 3/10/2015 |
| 4 | 2 | 3/10/2015 |
| 3 | 2 | 4/10/2015 |
| 2 | 2 | 4/10/2015 |

Coder en SQL :

1. Quels sont les noms des développeurs ayant livré au moins un correctif pour le client « UEM » ?
2. Quels sont les noms des développeurs n’ayant jamais livré de correctifs (tous clients confondus) ?
3. Quels sont les noms des développeurs ayant livré au moins 2 correctifs ?
4. Créez une table *champion(id, nom)*, contenant les développeurs dont le salaire est supérieur ou égal à 40000.

# Questions ouvertes (6 points)

Répondre à chaque question en 3-4 lignes maximum :

1. Qu’aimeriez-vous voir changé en Java ?
2. Plusieurs traitements de mon application ont besoin de consulter des données contenues dans un objet. Pour économiser de la mémoire, on pourrait partager cet objet pour qu’il ne soit instancié qu’une seule fois. Quel *design pattern* utilisez-vous ?
3. Un objet *Server* contient des données importantes pour tous les clients qui y sont connectés. Ça serait trop couteux pour chaque client de demander, de manière répétée, si l’état du *Server* a changé. A la place, on veut que le *Server* avertisse les clients. Quel *design pattern* utilisez-vous ?
4. Décrivez le *design pattern MVC*.
5. Quelle est la différence entre « == » et « equals » ?
6. En SQL, à quoi sert *UNION* et quelle est la différence avec *UNION ALL* ?
7. Quelle est la différence entre *INNER* et *OUTER JOIN* ? Expliquez avec un exemple.
8. Qu’est-ce que J2EE ?

Un environnement pour développer et déployer des applications web « entreprise » : fourni des services, APIs, protocoles, etc.

1. Que contient un fichier JAR ? un fichier WAR ? un fichier EAR ?

JAR : librairies Java génériques, ressources, etc.

WAR : une application web complète ;

EAR : une « enterprise application » : un ensemble de jar, ressources, applications web, etc.