

<b>Matière</b>	Programmation orientée objets
<b>Enseignant</b>	M. NAJIB
<b>Date</b>	31-10-2023
<b>Durée</b>	1h50

#### Documents non-autorisés

Course Outcomes (Syllabus) avec le pourcentage			
Course Outcomes	Evaluer	Exercice	Pourcentage
CO1: Master the basic concepts of OOP	X	1	50%
CO2: Master the visibility of attributes, methods, inheritance principles, polymorphism, abstract classes, and interfaces	X	2, 3	33%
CO3: Develop ergonomic graphic interfaces and events programming using JAVA	-	-	-

### Exercice 1 : (3 pts)

#### Questions de cours :

- Dans une relation d'héritage, une classe fille peut hériter les attributs protégés de la classe mère. (Vrai ou faux) **(1pt)**
- Une classe Java peut hériter d'une interface. (Vrai ou faux) **(1pt)**
- Expliquez le rôle de la machine virtuelle JAVA (JVM). **(1pt)**

### Exercice 2 : (9 pts)

Pour le développement d'une application de gestion des projets, nous souhaitons utiliser les classes suivantes :

- ProjetDevLogiciel** (*id* (entier auto-inc), *nom*, *budget* (double), *langageProg* (**string privé**), *tblDev* (tableau de 4 développeurs max (length = 4)))
- Projet** (*id* (entier auto-inc), *nom*, *budget* (double))
- ProjetMarketing** (*id* (entier auto-inc), *nom*, *budget* (double), *publicCiblé* (string), *responsable* (type Développeur))

#### Contraintes :

- Id** du projet est un entier **auto-incrémenté**
- On suppose que la classe **Développeur** caractérisée par (**idDev**, **nomDev**) est **donnée** avec les **constructeurs** et **littérateur** (toString).
- Vous pouvez utiliser les **listes** à la place des tableaux.
- Toutes les classes sont déclarées dans le même package

#### Questions :

- Donnez un schéma qui illustre la relation d'héritage entre les trois classes projets. (Précisez les attributs de chaque classe). **(2pts)**

- Donnez le code de la classe « **Projet** » avec **constructeur** full-paramètres, **getters** et **setters** (en cas de besoins) et la méthode **toString**. (2pts)
- Donnez le code de la classe « **ProjetDevLogiciel** » avec **constructeur** full-paramètres, **getters** et **setters** (en cas de besoins) et la méthode **toString**. (3pts)
- Dans la méthode Main, donnez le code nécessaire pour la création d'un objet de type « **ProjetDevLogiciel** ». Initialiser le tableau des développeurs de ce projet (**tblDev**) avec deux développeurs (hard-coded, sans l'utilisation des Scanners) (2pts)

### Exercice 3 : (8 pts)

Nous supposons que tous les projets de développement logiciels sont enregistrés dans le tableau « **tProjDev** » déclarée dans la méthode main comme suite :

```
public class Test {
    public static void main (String[] args) {
        ProjetDevLogiciel[] tProjDev = new ProjetDevLogiciel[20];
    }
}
```

**Questions :** (les méthodes demandées sont déclarées dans la classe « **ProjetDevLogiciel.java** »)

- Donnez le code de la méthode qui permet de récupérer un tableau (**tProjDev**) trié par ordre croissant du budget du projet. (2pt)

```
public ProjetDevLogiciel[] trierProjets(ProjetDevLogiciel[] tProjDev)
```

- Donnez le code de la méthode qui permet de récupérer le nombre de développeurs affectés à un projet de développement logiciel. (2pts)

```
public int countDev()
```

- Donnez le code de la méthode qui permet de supprimer les projets de développement logiciel sans développeurs affectés (0 développeur) et de récupérer le tableau des projets après suppression (2pts)

```
public ProjetDevLogiciel[] suppProjetVide(ProjetDevLogiciel[] tProjDev)
```

- Donnez le code de la méthode qui permet d'afficher le nombre de projets auxquels participe chaque développeur, comme suit (2pt) :

Id developpeur	Nom	Nbr de projets
1	- Rachidi	- 2 projet(s)
2	- Taha	- 5 projet(s)

**Bonne chance**