Harmonisation/Propédeutique POO : Où en êtes-vous après 2 séances ?

QUESTIONS : Compréhension de code

1. A partir du code suivant répondez aux questions ci-dessous :

```
/*
 //On considere que on doit passer par build office Name

 public void setOffice(String officeName) {
        if (isOfficeInformationValid(officeName)) {
            this.office = officeName;
        }
        else {
            this.office = DEFAULT_OFFICE;
        }
    }

 */
```

a. Est-ce que ce code est le code

- ❑ D'un constructeur
- ▪ **D'une méthode**
- ❑ D'un accesseur en lecture
- ▪ **D'un accesseur en écriture**

Si vous avez répondu seulement méthode, votre réponse n'est pas assez précise. Il s'agit d'un accesseur en écriture.

Si vous avez répondu seulement accesseur en écriture, ce n'est pas très grave puisque votre réponse est précise. Cependant tous les accesseurs sont des méthodes d'instance spécifiques de la classe.

b. Quelle est la signature de cette méthode ?

public void setOffice(String officeName)

public : la visibilité - les accesseurs doivent pouvoir être utilisés hors de la classe.

void : un accesseur en écriture ne renvoie pas de valeur.

setOffice : convention de nommage.

String : type du paramètre.

officeName : nom du paramètre.

Notez que le terme de signature peut être défini autrement pour expliciter la surcharge Java (https://docs.oracle.com/javase/tutorial/java/javaOO/methods.html***). Si vous vous êtes appuyé sur cette définition à la place de celle donnée en cours nous acceptons la réponse. Dans la leçon, la signature correspond aux éléments que vous trouvez dans l'API Java.***

c. Quelle variable d'instance est modifiée dans ce code ? ***office***

Donnez en une déclaration possible à partir de son utilisation : ***private String office;***

private : visibilité les variables d'instances ne doivent pas être utilisées directement hors de la classe. Le type String est déduit de l'instruction d'affectation : this.office=officeName;

d. A quoi correspond DEFAULT_OFFICE

- **Une constante**
- ❑ Une variable d'instance
- ❑ Un paramètre

Hormis le problème du non respect de la convention de nommage, vous auriez pu dire variable d'instance mais constante est le meilleur choix ici.

Donnez une instruction possible qui permet de déclarer DEFAULT_OFFICE

private static final String DEFAULT_OFFICE = "A 0";

final implique que la valeur ne pourra jamais être changée. private ou public selon la visibilité

La valeur par défaut est au choix du développeur ou dépend de la spécification.

e. A quoi correspond *officeName*

- ❑ Une constante
- ❑ Une variable d'instance
- **Un paramètre**

Pourrait-on modifier l'instruction *this.office = officeName par office = officeName ?* **oui** Justifiez votre réponse?

Dans ce cas l'instruction office=officeName; car le nom de la variable d'instance et le nom du paramètre sont différents.

f. L'instruction *isOfficeInformationValid(officeName)*

Correspond-elle

- **à un appel de méthode ?**

❑ à une déclaration de méthode ?

On aurait pu écrire this.isOfficeInformationValid(officeName) pour expliciter l'appel de la méthode / envoi de message

Est-ce que cette méthode est déclarée dans la même classe que setOffice ?

Oui car elle s'applique sur une instance de la classe sinon l'utilisation du "." serait obligatoire pour désigner l'objet qui doit répondre au message

g. Donnez une signature possible de cette méthode

public boolean isOfficeInformationValid((String office)

Si la méthode a une visibilité public, elle peut être appelée hors de la classe ; si on la met private, on ne peut l(utiliser que dans la classe où elle est définie. On peut déduire du code qu'elle renvoiesoit true soit false (un booleen). Le nom du paramètre peut être n'importe quel nom du moment qu'il est clair. Le type String se déduit du code.

2. A partir du code suivant répondez aux questions ci-dessous :

```
//Part I : Question 37 ------------------------
public Teacher(String name, String firstName) {
    this.name = name;
    this.firstName = firstName;
    this.office = DEFAULT_OFFICE;
    buildEmail();
}

//Part I : Question 38 ------------------------
public Teacher(String name, String firstName, String office) {
    this(name, firstName);
    this.office = office;
}
```

a. Est-ce que ce code est le code

- **De constructeurs**

❑ De méthodes

❑ D'accesseurs en lecture

❑ D'accesseurs en écriture

Vous pouvez avoir également coché méthodes mais les constructeurs ont un rôle très spécifique de créer des instances (objets)

b. Donnez-en les signatures :

public Teacher(String name, String firstName)

public Teacher(String name, String firstName, String office)

c. A quoi correspond *office* dans la signature

- ❑ Une constante
- ❑ Une variable d'instance
- ▪ **Un paramètre**

d. Pourrait-on modifier son nom par *officeName* ? ***Oui on peut mettre le nom que l'on souhaite***

quel est l'impact sur le code ? ***Remplacer office par officeName dans l'affectation de la variable d'instances this.office = officeName;***

e. Proposez un code avec l'appel en cascade dans le premier code au lieu du 2ieme code

```
public Teacher(String name, String firstName) {
    this(name, firstName, DEFAULT_OFFICE);
}
public Teacher(String name, String firstName, String office) {
    this.name = name;
    this.firstName = firstName;
    this.office = office;
    buildEmail();
}
```

f. Ecrire un code possible pour la classe qui contient ce code en le déduisant du code fourni ?

On peut déduire le nom de la classe, des variables d'instances et des signatures de méthodes, on peut écrire les accesseurs en lecture, une methode toString correspondant aux variables identifiées et un main simple.

```
public class Teacher {
 private String name;
 private String firstName;
 private String office;
 static final String DEFAULT_OFFICE = "A 0";
  public Teacher(String name, String firstName) {
      this.name = name;
      this.firstName = firstName;
```

```
        this.office = DEFAULT_OFFICE;
        buildEmail();
    }
  public Teacher(String name, String firstName, String office) {
        this(name, firstName);
        this.office = office;
    }

    public String getName() {  return name; }
    public String getFirstName() {return firstName;}
    public String getOffice() { return office;;}
    private void buildEmail()  { }
    @Override
    public String toString() {
        return "Teacher{" +
                "name='" + name + '\'' +
                ", firstName='" + firstName + '\'' +
                ", office='" + office + '\'' +
                '}';
    }

     public static void main(String[] args) {
        Teacher teacher1 = new Teacher("Ada", "Lovelace");
        System.out.println(teacher1);
        System.out.println(teacher1.getFirstName());
        System.out.println(teacher1.getName());
        System.out.println(teacher1.getOffice());
     }
}
```

QUESTIONS : Ecriture de code

Supposant que la classe Salle demandée au test de la semaine dernière existe (cf. rappel des spécifications de la classe)

Ecrire ci-dessous le code de la classe (variables d'instance, constructeurs, méthodes, accesseurs, toString, main) qui permet de définir un enseignement par son nom, le nombre d'étudiants inscrits et une salle privilégiée. Le nombre d'étudiants inscrits ne peut pas être supérieur à la capacité de la salle affectée. Si le nombre est supérieur affectez une salle avec comme valeur du batiment "Non affectee , le numero de salle 0 et la capacité 0. L'affectation d'une salle ayant une capacité suffisante n'est pas à traiter dans cet exercice.

Cette classe n'implémente pas d'accesseurs en écriture.

```
public class Enseignement{
 private String nom;
 private int nbEtudiantsInscrits;
 private Salle sallePrivilegiee;
public Enseignement(String theme, int nbEtudiants, Salle uneSalle) {
      this.nom = theme;
      Int places = uneSalle.placesRestantes(nbEtudiants);
       this.nbEtudiantsInscrits = nbEtudiants;
      If (places >= 0) { this.sallePrivilegiee = uneSalle; }
        else { this.sallePrivilegiee = new Salle("Non affectee",0,0);  }
   }
public String getNom() {
      return nom;
   }
public int getNbEtudiantsInscrits() {
      return  nbEtudiantsInscrits;
   }
public Salle getSallePrivilegiee () {
      return sallePrivilegiee;
   }
public String salleAffectee()  {
If (sallePrivilegiee.getBatiment().equals( "Non affectee") ) {return "pas de salle affectee"; }
else { return sallePrivilegiee.toString(); }
}
```

```
public String toString() {
        return "Enseignement{" +
                "thème='" + nom + '\'' +
                ", nombre d'étudiants inscrits ='" + nbEtudiantsInscrits+ '\'' +
                ", salle :"  + salleAffectee()+ '\''  +          '}';
    }
     public static void main(String[] args) {
        Enseignement ens1 = new Enseignement("POO", 43, new Salle("E", 310, 50));
        System.out.println(ens1);
       Enseignement ens2 = new Enseignement("Reseaus", 55, new Salle("E", 310, 50));
        System.out.println(ens2);
    }
```

Spécification de la classe Salle : Ecrire ci-dessous le code de la classe (variables d'instance, constructeurs, méthodes) qui permet de définir une salle de classe par le nom d'un bâtiment, le numéro de salle et sa capacité et qui permet d'obtenir le nombre de places restantes ou manquantes par rapport à un nombre d'étudiants donné.