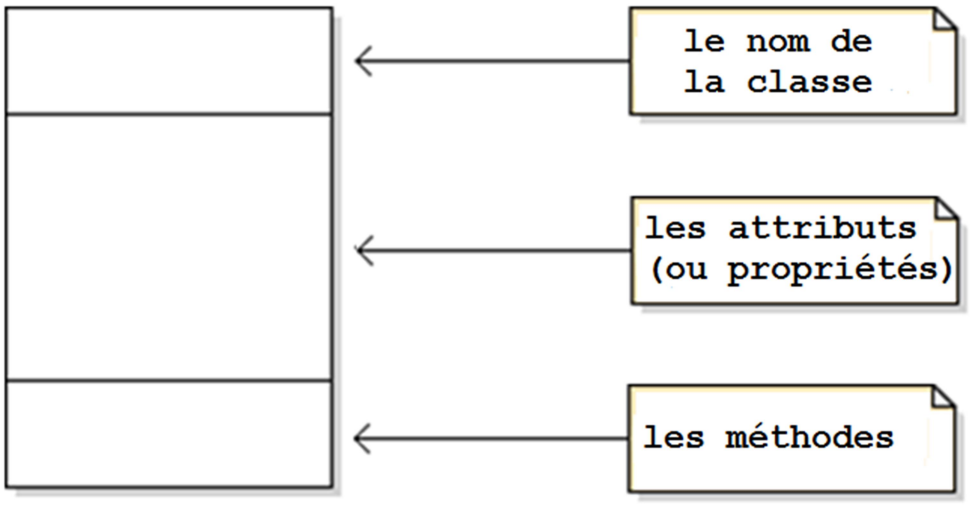
Proposition de correction

1) Remplir le questionnaire du **document 3**

1) Complétez le schéma suivant :

Ce rectangle représente : une classe en diagramme de classe UML



2) Expliquez le terme "encapsulation" ?

L'encapsulation est le fait **d'embarquer** dans une même capsule (l'objet) des **attributs** (sa description) et des **méthodes** (son savoir-faire) et de protéger sa partie sa partie privée.

3) Que signifie le terme "privé" pour un attribut ?

Un attribut privé n'est pas visible ni utilisable directement lors de la manipulation de l'objet. Sa manipulation est uniquement possible lors de la conception de la classe.

4) Expliquez le terme "accesseur" (ou méthode d'accès)

Un accesseur est une fonction ou une procédure qui permet un accès en lecture (de type get) et/ou en écriture (de type set) à un attribut privé.

5) Comment appelle-t-on l'ensemble des méthodes publiques ?

L'ensemble des méthodes publiques s'appelle "**l'interface**" et constitue la seule possibilité de manipuler l'objet.

6) Quel est le rôle d'un constructeur ?

Le rôle d'un constructeur est **d'instancier (de créer)** un objet en **valorisant** ses attributs.

7) Quel est le mécanisme qui permet d'avoir plusieurs constructeurs de même nom ?

Il s'agit de la "**surcharge**", les constructeurs portent le même nom (celui de la classe) mais sont différenciés par le type et le nombre des paramètres (la signature).

2) Codez entièrement la classe *Ordinateur* en langage java en tenant compte des règles de gestion présentes dans le **document 1**.

```
public class Ordinateur{

     //les attributs privés
     private String reference;
     private boolean portable;
     private String processeur;
     private int ram;
     private int disque;
     private String service;

     //constructeur de base
     public Ordinateur(String ref, boolean p, String pr, int d){
          this.reference = ref;
          this.portable = p;
          this.processeur = pr;
          this.ram = 4;        //valeur par défaut
          this.disque = d;
          this.service = null; //le service sera affecté ultérieurement
     }

     //constructeur alternatif
     public Ordinateur(String ref, boolean p, String pr, int ram, int d){
          this.reference = ref;
          this.portable = p;
          this.processeur = pr;
          this.ram = ram;
          this.disque = d;
          this.service = null; //le service sera affecté ultérieurement
     }

     //les méthodes publiques

     //Les accesseurs
     public String getReference(){
          return this.reference;
     }

     public void setService(String s){
          this.service = s;
     }

     public void setRam(int r){
          this.ram = r;
     }
```

Ici je suis en conception de classe !

Ici je suis en conception de classe !

```
    public boolean mvOK(){
        return (this.ram >= 8 && this.disque >= 100);
    }
```

code équivalent

```
boolean res = false;
if(this.ram >= 8 && this.disque >= 100){
    res = true;
}
return res;
```

```
    @Override
    public String toString(){
        String res = "Référence : " + this.reference + "\n";
        if(this.portable){
            res = res + "\nOrdinateur portable\n";
        }
        else{
            res = res + "\nOrdinateur de bureau\n";
        }
        res = res + "\nProcesseur : " + this.processeur + "\n";
        res = res + "\nRam : " + this.ram + " Go\n";
        res = res + "\nDisque : " + this.disque + " Go\n";
        if(this.service == null){
            res = res + "\nService : Non affecté\n";
        }
        else{
            res = res + "\nService : " + this.service + "\n";
        }
        return res;
    }
}
```

3) Codez le scénario présent dans le **document 2** dans un programme de test.

Ici je suis en utilisation d'objets !

```
public class Test{
   public static void main(String args[]){

      //Déclaration des variables
      Ordinateur o1,o2;

      //Instanciation des objets
      //appel du constructeur de base
      o1 = new Ordinateur("B-01",false, "AMD Rysen 3", 500);

      //appel du constructeur alternatif
      o2 = new Ordinateur("P-02",true, "Intel Core i7", 8, 1000);

      //Affectation des services
      o1.setService("Comptabilité");
      o2.setService("DSI");

      //Affichage des descriptions
      System.out.println(o1.toString());
      System.out.println(o2.toString());

      //Modification de la ram
      o2.setRam(16);

      //il n'est pas possible d'afficher uniquement la ram de o2 car cet attribut est privé !!

      //Test machine virtuelle
      if(o1.mvOK()){
           System.out.println (o1.getReference() + " peut faire fonctionner une machine virtuelle ");
      }
      else{
           System.out.println(o1.getReference()  +  "  ne  peut  pas  faire fonctionner une machine virtuelle");
      }
   }
}
```