IPO, TD, annale examen 2021

1 Questions de cours

Question 1 En supposant les noms bien choisis, associer les quatre noms suivants à leur rôle (interface, classe, méthode ou variable) :
`Furniture`
`animal`
`Deletable`
`driveStraight`

Question 2 Soit la classe Vehicule :

```
1  public class Vehicule {
2     protected String immatriculation;
3     private int puissance;
4
5     public Vehicule(String immatriculation, int puissance) {
6        this.immatriculation = immatriculation;
7        this.puissance = puissance;
8     }
9
10    public String getImmatriculation() {
11       return immatriculation;
12    }
13
14    public void setImmatriculation(String immatriculation) {
15       this.immatriculation = immatriculation;
16    }
17
18    public int getPuissance() {
19       return puissance;
20    }
21 }
```

et la classe Voiture :

```
1  public class Voiture extends Vehicule {
2     private int nbPortes;
3
4     public Voiture(String immatriculation, int puissance, int nbPortes) {
5        super(immatriculation, puissance);
6        this.nbPortes = nbPortes;
7     }
8
9     public int getNbPortes() {
10       return nbPortes;
11    }
12 }
```

Après les déclarations suivantes (dans un `main`) :

```
1        Vehicule moto = new Vehicule("ZZZ", 4);
2        Voiture voiture = new Voiture("ABC", 6, 5);
3        Vehicule def = new Voiture("DEF", 6, 3);
```

que donneraient les instructions suivantes ? Expliquer brièvement pourquoi.

1. `System.out.println(voiture.getImmatriculation());`
2. `System.out.println(voiture.getNbPortes());`
3. `System.out.println(moto.getImmatriculation());`
4. `System.out.println(moto.getNbPortes());`
5. `System.out.println(def.getImmatriculation());`
6. `System.out.println(def.getNbPortes());`
7.
```
Vehicule[] mesVehicules = {moto, voiture, def};
for (int i =0; i<3; i++){
   System.out.println(mesVehicules[i].getPuissance());
}
```

Question 3 Soient les interfaces et classe suivantes :

```
1  public interface Identifiable {
2     String getIdentite();
3  }
```

```
1  public interface AppelableParTelephone {
2     String getTelephone();
3  }
```

```
1  public interface Localisable {
2     String getAdresse();
3  }
```

```
1  public abstract class IdentiteBancaire implements Identifiable {
2     protected final String rib;
3     protected String nom;
4     private double fonds;
5
6     public IdentiteBancaire(String rib, String nom) {
7        this.rib = rib;
8        this.nom = nom;
9        this.fonds = 0;
10    }
11
12    public abstract String procedureDeDemarchage();
13
14    public void crediter(double montant){
15       fonds += montant;
16    }
17
18    public boolean debiter(double montant){
19       if (montant > fonds){
20          return false;
21       }
22       fonds -= montant;
23       return true;
24    }
25
26 }
```

Écrire une classe `Societe` minimale héritant d'`IdentiteBancaire` et implémentant `AppelableParTelephone` et `Localisable`

Question 4 Quel(s) intérêt(s) voyez-vous à utiliser Git pour un projet que l'on réalise seul ?

2 Création et composition de classes : Gestion de stock (11 points)

On cherche ici à développer une ébauche de gestion de stock d'une application de vente pour un magasin. On crée pour cela des produits, qui vont être associés à des quantités pour apparaître dans l'inventaire du magasin ou dans les achats des clients. On distingue un type de produit particulier : les aliments, qui ont une date de péremption (la même pour tout le lot).

Le code de la classe `Produit`, le squelette de la classe `Achat` et un exemple de main de test vous sont fournis en annexe 2.

Un produit a toujours une désignation (le nom de l'article), un prix, et un code produit qui l'identifie sans ambiguïté. Il peut posséder une description.

Question 12 Écrire le constructeur `public Produit(String designation, String description, String codeProduit, String prixStr)` de la classe `Produit`. Ce constructeur doit appeler le constructeur principal en convertissant le prix donné sous forme de chaîne de caractère en double, via la méthode de le classe `Double` : `public static double parseDouble(String s)`.
Nous supposons la chaîne au bon format (c'est-à-dire qu'il n'est pas demandé de traiter les cas d'erreurs).

Question 13 Écrire une classe `ProduitEnQuantite` qui devra contenir :

- un attribut `produit`, donnant un référence vers le produit concerné
- un attribut entier `quantite`
- un constructeur, prenant un produit et un quantité en paramètre
- des getters pour produit et quantite

Question 14 Écrire, pour la classe `ProduitEnQuantite` les méthodes suivantes :

- `public void setQuantite(int quantite)` qui fixe la quantité du produit à celle indiquée, à condition que celle-ci soit positive. Sinon, la quantité est fixée à 0.
- `public void ajouterAuStock(int nb)` qui ajoute `nb` à la quantité disponible
- `public boolean vendre(int nb)` qui retire `nb` à la quantité disponible s'il y en a assez et renvoie vrai, sinon ne fait rien et renvoie faux.

Question 15 Quelle est l'utilité d'avoir à la fois un setter et des méthodes pour ajouter et diminuer la quantité ?

Question 16 Écrire un deuxième constructeur pour `ProduitEnQuantite`, prenant un unique paramètre de type `Produit` en en fixant la quantité à 1 par défaut. Ce constructeur doit appeler le constructeur précédemment défini.

Les questions qui suivent vous demandent de définir votre classe `Inventaire` avec vos propres choix d'implémentation et de structures. Vous serez notés sur la justesse de votre code mais également sur la pertinence de vos choix.
Conseil : l'énoncé des questions 17 à 19 peut vous aider à comprendre ce que l'on vous demande et donc à faire des choix d'implémentation pertinents.

Question 17 Écrire une classe inventaire, qui doit contenir un ensemble de produits en stock. Elle doit contenir un constructeur, une méthode pour y ajouter en produit en une quantité voulue et une méthode pour vendre (c'est à dire retirer du stock) un produit, dans une quantité indiquée.

Question 18 Écrire une méthode `purge` qui supprime de l'inventaire tous les produits présents avec une quantité de 0.

Question 19 Écrire la méthode `public String toString()` qui affiche l'état courant de l'inventaire, en listant tous les produits présents, suivis de leurs quantités.
Un rendu possible :

```
trombone, 0.38euros, 4 en stock
stylo, un stylo noir à bille, 0.56euros, 10 en stock
ardoise, 2.78euros, 5 en stock
```

Nous allons maintenant écrire une classe `Aliment` représentant un type de produit particulier ayant une date de péremption.

Question 20 Écrire une classe `Aliment` qui hérite de `Produit`. Elle y ajoute un champ textuel (de format non précisé) `dateDePeremption`. Y inclure un constructeur, ainsi que la méthode `public String toString()` qui ajoute la date de péremption à la suite des informations sur le produit.

Question 21 Avez-vous besoin d'ajouter autre chose à la classe Aliment ou d'adapter le reste de vote code ? Comment se comporte la méthode `toString` d'`Inventaire` lorsque le stock comprend à la fois des produits alimentaires et non-alimentaires ?

Nous nous intéressons maintenant à la création d'une classe `Achat`, qui permet de faire diminuer le stock en fonction de l'achat de plusieurs produits en une fois (lors de passages en caisse). Un squelette vous est donné en annexe 2.

Question 22 Écrire la méthode `public boolean ajouteProduit(Produit p, int quantite)` permettant d'ajouter un produit à l'achat en une certaine quantité. On ne vérifiera pas si le même produit est déjà présent dans le contenu de l'achat. En revanche, l'appel à cette méthode doit réduire de la quantité indiquée le stock du produit correspondant dans l'inventaire. La méthode ne fait rien et renvoie faux dans le cas où le stock serait insuffisant.

Question 23 Écrire la méthode `public double getPrix()` qui calcule le prix total de l'achat.

Question 24 Écrire la méthode `public void annule()` qui remet en stock tous les produits du contenu puis vide le contenu de l'achat.

Question 25 Avez-vous des remarques sur vos choix d'implémentation ? Quels points supplémentaires faudrait-il développer ?

3 Annexe 1 : squelettes pour la gestion de stock

Produit.java

```
1  package vente;
2
3  public class Produit {
4      private String designation;
5      private String description;
6      private final String codeProduit;
7      private double prix;
8
9      public Produit(String designation, String description,
10                    String codeProduit, double prix) {
11         this.designation = designation;
12         this.description = description;
```

```
13          this.codeProduit = codeProduit;
14          this.prix = prix;
15      }
16
17      public Produit(String designation, String description,
18                          String codeProduit, String prixStr) {
19          //TODO Question 12
20      }
21
22      public double getPrix() {
23          return prix;
24      }
25
26      public void setPrix(double prix) {
27          this.prix = prix;
28      }
29
30      public String getCodeProduit() {
31          return codeProduit;
32      }
33
34      public String getDesignation() {
35          return designation;
36      }
37
38      @Override
39      public String toString() {
40          //TODO Question
41          return designation + ", " +
42                  ((description == null)?"":(description + ", ")) +
43                  prix + "euros";
44      }
45  }
```

Achat.java

```
1   package vente;
2
3   import java.util.ArrayList;
4   import java.util.List;
5
6   public class Achat {
7       private List<ProduitEnQuantite> contenu;
8       private Inventaire inventaire;
9
10      public Achat(Inventaire inventaire){
11          contenu = new ArrayList<>();
12          inventaire = inventaire;
13      }
14
15      public boolean ajouteProduit(Produit p, int quantite){
16          //TODO Question 22
17      }
18
19      public double getPrix(){
```

```
20          //TODO Question 23
21      }
22
23      public void annule(){
24          //TODO Question 24
25      }
26  }
```

Un main, ici dans Inventaire.java :

```
1   package vente;
2
3   public class Inventaire {
4
5           //TODO Questions 17-18-19
6
7       public static void main(String[] args) {
8           Produit trombone = new Produit("trombone", null, "T438", 0.38);
9           Produit stylo = new Produit("stylo", "un stylo noir à bille", "S003", 0.56);
10          Produit ardoise = new Produit("ardoise", null, "A768", 2.78);
11
12          Inventaire stockDuMagasin = new Inventaire();
13
14          stockDuMagasin.ajouter(trombone, 4);
15          stockDuMagasin.ajouter(stylo, 10);
16          stockDuMagasin.ajouter(ardoise, 5);
17
18          Produit yaourt = new Aliment("yaourt", "4 pots de yaourt à la fraise",
19                  "2Y45", 0.73, "17-01");
20
21          stockDuMagasin.ajouter(yaourt, 30);
22          Achat unClient = new Achat(stockDuMagasin);
23          unClient.ajouteProduit(yaourt, 1);
24          unClient.ajouteProduit(stylo, 2);
25          unClient.ajouteProduit(trombone, 26);
26
27          System.out.println(stockDuMagasin);
28      }
29  }
```

La sortie de ce programme est :

```
yaourt, 4 pots de yaourt à la fraise, 0.7euros, date de péremption : 17-01, 29 en stock
trombone, 0.38euros, 4 en stock
stylo, un stylo noir à bille, 0.56euros, 8 en stock
ardoise, 2.78euros, 5 en stock
```