Examen écrit - Programmation par objets (Java)

Tous documents autorisés

Dans ce sujet, nous étudions diverses opérations liées à la gestion d'un restaurant administratif (restaurant destiné aux membres du personnel d'une administration et à leurs invités). Le schéma de la figure 1 présente les différentes classes auxquelles nous allons nous intéresser de manière progressive et partielle dans ce sujet. Le restaurant administratif (`RestaurantAdministratif`) sert des repas (`RepasServi`), qui peuvent être soit des repas pris par des invités (`RepasInvite`), soit des repas pris par des membres du personnel (`RepasPersonnel`). Un repas pris par un membre du personnel est associé à ce membre du personnel (`MembrePersonnel`).
Dans toutes les questions, les accesseurs qui ne sont pas spécifiquement demandés sont supposés exister, n'écrivez pas leur code. N'écrivez pas non plus les constructeurs sans paramètres, qui seront supposés exister également.

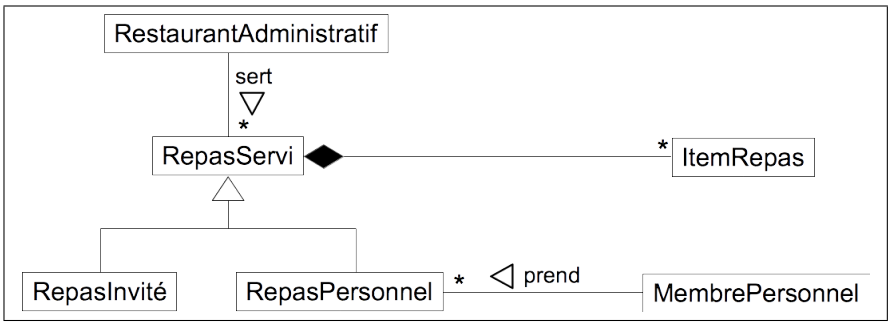

FIGURE 1 – Restaurant administratif

Question 1 *Prenez connaissance de l'énumération et de la classe présentées à la figure 2 et qui représentent respectivement les catégories d'items que l'on peut choisir dans un menu (comme une entrée ou un dessert) et un item de menu. Pour chacune des 4 instructions du programme de la figure 3, indiquez si elle compile ou non (une simple réponse par oui ou non qui ne serait pas justifiée n'apporte pas de point).*

Réponse 1

(0,25pt) `ItemRepas i1 = new ItemRepas(); // I1`
Cette instruction crée un `ItemRepas` *avec le constructeur sans paramètre (qui existe bien). Elle compile et s'exécute sans erreur.*

(0,25pt) `ItemRepas i2 = new ItemRepas("crumble",Categorie.dessert,0.77); // I2`
Cette instruction crée un `ItemRepas` *avec le constructeur de signature* `(String, Categorie, double)` *qui existe bien). Elle compile et s'exécute sans erreur.*

(0,25pt) `ItemRepas i3 = new ItemRepas(0.74); // I3`
Il n'existe aucun constructeur de signature `(double)`*. L'analyse de cette instruction provoque une erreur de compilation.*

(0,25pt) `ItemRepas i4 = new ItemRepas(Categorie.dessert); // I4`
Cette instruction crée un `ItemRepas` *avec le constructeur de signature* `(Categorie)` *qui existe bien). Elle compile et s'exécute sans erreur.*

```
public enum Categorie {entree, plat, accompagnement, dessert;}

public class ItemRepas {
    private String nom = "inconnu";
    private Categorie categorie = Categorie.accompagnement;
    private double prix; // en euros
    public ItemRepas() {}
    public ItemRepas(Categorie categorie) {
        this.categorie = categorie;
    }
    public ItemRepas(String nom, Categorie categorie, double prix) {
        this.nom = nom;
        this.categorie = categorie;
        this.prix = prix;
    }
    public String getNom() {return nom;}
    public void setNom(String nom) {this.nom = nom;}
    public Categorie getCategorie() {return categorie;}
    public void setCategorie(Categorie categorie) {this.categorie = categorie;}
    public double getPrix() {return prix;}
    public void setPrix(double prix) {this.prix = prix;}
    public String toString(){return this.nom+" "+this.categorie+" "+this.prix;}
}
```

FIGURE 2 – Classe ItemRepas

```
public class ProgrammeRestaurant {
      public static void main(String[] argv)
      {
            ItemRepas i1 = new ItemRepas(); // I1
            ItemRepas i2 = new ItemRepas("crumble",Categorie.dessert,0.77); // I2
            ItemRepas i3 = new ItemRepas(0.74); // I3
            ItemRepas i4 = new ItemRepas(Categorie.dessert); // I4
      }
}
```

FIGURE 3 – Classe Programme

Question 2 *Ecrivez une classe* `RepasServi` *contenant les attributs et le constructeur avec paramètre(s) déduits des informations suivantes :*

— *la classe est sans instance propre,*
— *un repas servi se compose d'items de repas. La liste d'items de repas est toujours vide à la création du repas,*

— *on mémorise la date à laquelle le repas est servi. Cette date est une chaîne de caractères sous le format AAAAMMJJ.*

Réponse 2
(0,5pt) pour abstract
(0,5pt) pour déclaration de l'attribut `listeItem`
(0,5pt) pour l'initialisation de l'attribut
(0,25pt) pour déclaration de l'attribut `date`
(0,5pt) pour le constructeur

```
public abstract class RepasServi {
      private ArrayList<ItemRepas> listeItem = new ArrayList<ItemRepas>();
      private String date; // AAAAMMJJ
      public RepasServi(String date){this.date = date;}
      .........
}
```

Question 3 *Ecrivez une classe* `RepasInvite` *contenant les attributs et le constructeur avec paramètre(s) déduits des informations suivantes :*

— *l'invité est un adulte ou un enfant (mais on ne mémorise que cette information et pas l'identité de l'invité).*

Réponse 3
(1pt) pour extends ...
(0.25pt) pour l'attribut
On peut coder le fait que l'invité est un adulte ou un enfant par un booléen, une chaîne de caractères, une énumération ... Ici on a choisi un booléen.
(1pt) pour constructeur incluant 0,5 pt pour signature, affectation et 0,5pt pour appel à super ...

```
public class RepasInvite extends RepasServi {
     private boolean adulte = true; // versus enfant
     public RepasInvite(String date, boolean adulte) {
          super(date);
          this.adulte = adulte;
     }
     .........
}
```

Question 4 *Etudiez la classe* `MembrePersonnel` *partielle présentée à la figure 4. Ecrivez une classe* `RepasPersonnel` *contenant les attributs déduits des informations suivantes (on ne demande pas de constructeur ici) :*

— *le repas est associé à un membre du personnel,*

— *à chaque repas est associé un droit d'admission, le même pour tous les membres du personnel, fixé à 2,59 euros (mais qui peut changer).*

Réponse 4
(1pt) pour extends ...

(0.5pt) pour l'attribut `membre`
(0.5pt) pour l'attribut `droitAdmission`

```
public class RepasPersonnel extends RepasServi {
private MembrePersonnel membre;
private static double droitAdmission = 2.59;
```

Question 5 *Ecrivez pour la classe* `RepasPersonnel` *une méthode* `subvention` *qui retourne :*

- *2,49 € pour les personnels dont l'indice INM est <= à 320,*
- *2.24 € pour les personnels dont l'indice INM est compris entre 321 et 400,*
- *1.84 € pour les personnels dont l'indice INM est compris entre 401 et 465,*
- *0.45 € pour les personnels dont l'indice INM est compris entre 466 et 550,*
- *0 dans les autres cas.*

Réponse 5
(2pt) incluant 0,5 pour liste des paramètres et type de retour, 0,5 pour récupérer l'INM, et 1 pour les tests et les return.

```
public double subvention()
     {
          // on peut introduire des variables statiques pour les tranches
          // mais l'enonce ne le specifie pas
          int inmMembre = this.getMembre().getINM();
          if (inmMembre <= 320) return 2.49;
          else if (inmMembre <= 400) return 2.24;
          else if  (inmMembre <= 465) return 1.84;
          else if  (inmMembre <= 550) return 0.45;
          else return 0;
     }
```

Question 6
(a) Complétez la classe `MembrePersonnel` *par un attribut représentant la liste des repas pris par ce membre du personnel.*
(b) Ajoutez à cette même classe une méthode permettant d'ajouter un repas à un membre du personnel. Pensez à mettre à jour les attributs correspondant des deux classes. Si le repas a été préalablement associé à une autre personne, un message d'information est affiché, le repas est retiré à cette autre personne et affecté au membre du personnel courant.

Réponse 6
(1pt) pour (a)
(2pt) pour (b) à répartir sur les différentes opérations

```
public class MembrePersonnel {
 .....
      private ArrayList<RepasPersonnel> listeRepas = new ArrayList<RepasPersonnel>();
......
     public void ajouteRepas(RepasPersonnel repas)
     {
```

```
        if (! this.listeRepas.contains(repas))
             {
                  if (repas.getMembre()!=null)
                       if (repas.getMembre()!=this)
                            {System.out.println("repas deja servi a "
                                    +repas.getMembre().getNom());
                            repas.getMembre().listeRepas.remove(repas);}
                       else
                            System.out.println("repas deja associe a ce membre");
                  this.listeRepas.add(repas);
                  repas.setMembre(this);
             }
    }     ............
}
```

```
public class MembrePersonnel {
      private String nom;
      private int INM; // indice de remuneration
      private String numCarte;
      public MembrePersonnel() {}
      public MembrePersonnel(String nom, int iNM, String numCarte) {
            this.nom = nom;
            INM = iNM;
            this.numCarte = numCarte;
      }
      public String getNom() {return nom;}
      public void setNom(String nom) {this.nom = nom;}
      public int getINM() {return INM;}
      public void setINM(int iNM) {INM = iNM;}
      public String getNumCarte() {return numCarte;}
      public void setNumCarte(String numCarte) {this.numCarte = numCarte;}
}
```

FIGURE 4 – Classe `MembrePersonnel`

Question 7 *Ecrivez les méthodes de calcul du prix des repas (pour les trois classes représentant les repas). Si vous pensez que des attributs supplémentaires seraient utiles, ajoutez-les.*

- *Pour les invités, le prix est 8,5 euros si la personne est adulte et 6 euros s'il s'agit d'un enfant.*
- *Pour les membres du personnel, le prix retourné se calcule par la somme des prix des items le composant, à laquelle on ajoute le prix de l'admission et on retranche la subvention.*

Réponse 7
(1pt) pour RepasServi
(1,5pt) pour RepasInvite
(2pt) pour RepasPersonnel
On présente ci-dessous les ajouts réalisés dans chacune des trois classes.

```
public abstract class RepasServi {
     public abstract double prix();
}

public class RepasInvite extends RepasServi {
     private static double prixAdulte = 8.5;
     private static double prixEnfant = 6;
     public double prix() {
          if (adulte) return RepasInvite.prixAdulte;
          return RepasInvite.prixEnfant;
     }
}

public class RepasPersonnel extends RepasServi {
     public double prix() {
          double sommeItem = 0;
          for (ItemRepas it : this.getListeItem())
               sommeItem += it.getPrix();
          return sommeItem+RepasPersonnel.getDroitAdmission()-this.subvention();
     }
}
```

Question 8 *Ecrivez la (les) méthode(s) permettant d'ajouter un item à un repas. Un repas invité peut contenir au plus une entrée, au plus un plat, au plus un accompagnement et au plus un dessert (donc au plus un item de chaque catégorie). Il n'y a pas de restriction pour les repas des membres du personnel.*

Réponse 8
(0,5pt) pour RepasServi
(0,5pt) pour ne rien mettre dans RepasPersonnel
(2pt) pour RepasInvite
On présente ci-dessous les ajouts réalisés dans les classes.
On ne met rien dans la classe `RepasPersonnel` *car l'ajout est réalisé comme dans la super-classe.*

```
public abstract class RepasServi {
     public void ajoute(ItemRepas item)
     {
          if (! this.listeItem.contains(item))
               this.listeItem.add(item);
     }
 }

 public class RepasInvite extends RepasServi {
     public boolean contientCategorie(Categorie cat)
     {
          for (ItemRepas it : this.getListeItem())
               if (it.getCategorie().equals(cat))
                    return true;
          return false;
     }

     public void ajoute(ItemRepas item)
     {
          if  (!this.contientCategorie(item.getCategorie()))
               super.ajoute(item);
          else System.out.println("Probleme d'ajout");
     }
}
```