Numéro d'anonymat :

Modélisation et programmation par objets 2

Tous documents sur support papier autorisés. Durée : 2h00
L'ensemble des réponses sera à donner sur les feuilles d'énoncé. Ne pas dégrapher les feuilles.

Dans la première partie du sujet (questions 1 à 5), nous présentons quelques éléments d'un logiciel pour une agence événementielle, dont le métier est d'organiser (et de facturer) des séminaires, des soirées, des salons, des animations pour des lancements de produits, etc. Dans ce sujet, nous étudions quelques classes simplifiées pour représenter des organisations d'événements de type soirée avec repas ou animation(s) musicale(s). Un événement sera composé de plusieurs activités. Nous considérons ici deux catégories d'activités : la restauration (les repas) et l'animation musicale.

Question 1. Ecrivez en Java une interface `Iactivite` représentant une activité.

Pour une activité, on doit disposer de méthodes pour connaître et changer la valeur (1) de son nom, (2) d'une information indiquant le fait qu'elle se déroule ou non en extérieur. On doit également pouvoir (3) saisir (dans une même méthode) les deux informations pour les mettre à jour, la saisie se faisant à partir d'un `Scanner` passé en paramètre et enfin (4) comparer deux activités passées en paramètre à une méthode pour savoir si elles portent le même nom : remarquez que ces deux méthodes (3) et (4) doivent avoir un corps écrit dans l'interface. Enfin, on doit disposer d'une méthode pour (5) calculer le prix TTC de l'activité. On rappelle que la classe `Scanner` dispose de méthodes sur le format `T nextT()` pour saisir des valeurs de la plupart des types primitifs `T` et de la méthode `String next()` pour saisir des `String`.

Question 2. On souhaite représenter le taux de la TVA pour la restauration, qui vaut 10% actuellement mais peut changer dans le futur. Peut-on le représenter dans l'interface `Iactivite` ? Si oui, indiquez de quelle(s) manière(s) en proposant le code correspondant, et sinon expliquez pourquoi.

Question 3. Transformez la classe `Evenement` dont le code vous est donné ci-dessous en classe générique `EvenementG` paramétrée par le type des activités. Le type des activités doit respecter le contrat de l'interface `Iactivite`. Indiquer les transformations sur le code ci-dessous (sans le récrire).

```
public class Evenement
{
  private String nom;

  private ArrayList<Activite> listeActivites = new ArrayList<>();

  public void ajout(Activite a) {

    if (this.listeActivites.contains(a)) System.out.println("deja presente");

    else this.listeActivites.add(a);

  }

  public double prixTTC() {

    double res = 0;

    for (Activite a : this.listeActivites)

      res+=a.prixTTC();

    return res;
  }

  public ArrayList<String> activExt() {

    ArrayList<String> res = new ArrayList<String>();

    for (Activite a : this.listeActivites)

      if (a.isEstExt())

        res.add(a.getNom());

    return res;
  }
}
```

Question 4. Ecrivez pour la classe générique `EvenementG` une méthode d'instance `deplaceEvts` permettant de déplacer dans un événement toutes les activités d'un autre événement passé en paramètre. Dans un programme hypothétique, un événement (receveur du message) doit pouvoir accueillir par cette méthode un autre événement (paramètre de la méthode) dont les membres sont plus spécialisés. Par exemple, en supposant que l'on dispose d'une classe `Activite` implémentant l'interface `Iactivite` et d'une classe `AnimationMusicale` qui spécialise la classe `Activite`, un événement composé d'activités (paramétré par `Activite`) peut accueillir des événements musicaux (paramétré par `AnimationMusicale`) (mais pas l'inverse). La classe `ArrayList` dispose d'une méthode `public boolean addAll(Collection<? extends E> c)` que vous pouvez utiliser.

Question 5. Réécrivez le code de la méthode `prixTTC` à l'aide de *streams* et de *lambdas expressions*.

Question 6. Indiquez quelles lignes du `main` suivant compilent et ce que va afficher le programme (sans détailler les passages à la ligne). Justifiez votre réponse en expliquant quel est le type statique et le type dynamique et comment la liaison dynamique s'effectue dans chacun des cas. Que pensez-vous de la méthode `description` de la classe `MortSubiteKriek` ?

```
public class Biere{
  public void description() {System.out.print("Biere ");this.couleur();}
  public void couleur() {System.out.println("variee");}
}

public class BiereAcidulee extends Biere{}

public class Lambic extends BiereAcidulee{
  public void description() {super.description(); System.out.println("Lambic ");}
  public void couleur() {System.out.println("blond cuivre");}
}

public class MortSubiteKriek extends Lambic{
  public void description()  {System.out.print("Biere Kriek "); super.couleur();
      System.out.println("Lambic ");}
  public void couleur() {System.out.println("rouge vif");}
}
// ...
public static void main(String[] args) {
 Biere biere1 = new Biere();    biere1.description();                          // main1
 Biere biere2 = new Lambic();   biere2.description();                          // main2
 Biere biereA1 = new MortSubiteKriek();  biereA1.description();                // main3
 BiereAcidulee biereA2 = new MortSubiteKriek();  biereA2.description();        // main4
 MortSubiteKriek biereA3 = new MortSubiteKriek();  biereA3.description();// main5
}
```

Question 7. Représentez par un diagramme de séquences UML les messages échangés entre un utilisateur annonceur et différents composants d'un logiciel pour la parution d'une annonce sur un site d'annonces. L'`annonceur` envoie au `contrôleur d'annonce` un message de demande de publication avec un texte. Le `contrôleur d'annonce` exécute sa méthode de vérification sur l'identité de l'annonceur et le contenu du texte pour vérifier qu'il n'est pas malveillant. Il retourne ensuite une réponse à l'`annonceur`. Si la réponse est "OK", le `contrôleur d'annonce` envoie un message au `gestionnaire de planning` pour lui demander une date possible pour la parution. Le `gestionnaire de planning` lui retourne une date. Le `contrôleur d'annonce` envoie alors un message de demande de parution au `gestionnaire de parution` avec les informations utiles (identité de l'annonceur, texte, date). Le `gestionnaire de parution` active sa méthode de programmation de parution. Puis il envoie à l'`annonceur` un message de confirmation. Nous vous proposons de compléter les lignes de vie des objets du schéma ci-dessous pour le mettre en place.