Programmation
M1 IPS, BCD, SSV, M2 Géomatique
HMIN111M - 05/01/17 - 16h-17h30
Tous documents sur support papier autorisés. Durée : 1h30

CORRECTION

1 Présentation

Dans ce sujet, nous présentons quelques éléments simplifiés d'un logiciel pour la gestion d'un parc d'attraction animalier. Le parc propose un ensemble d'attractions autour de la thématique animalière (spectacles de dressage, projection de films, activités de nourrissage, promenades avec des animaux, etc.). La partie du logiciel que nous étudierons est destinée à effectuer un suivi des prévisions de la rentabilité des attractions prises indépendamment et prises dans leur globalité ; toute attraction n'étant pas forcément destinée à être rentable individuellement.

2 Représentation des attractions

Les attractions (`Attraction`) sont toutes décrites par :
— une description (chaîne de caractères) (`description`).
— un nombre de spectateurs attendus annuels (`nbSpect`).
— le tarif d'entrée pour un spectateur (`tarif`). On ne gèrera pas ici de tarifs réduits pour simplifier.
Deux types d'attractions seront également étudiées plus en détails :
— les spectacles de dressage (`Dressage`).
— les projections de films (`Projection`).
Les spectacles de dressage sont des attractions qui sont de plus décrites par :
— le type d'animaux (chaîne de caractères) impliqués dans ce spectacle (`typeAnimaux`).
— le coût annuel prévisionnel de l'entretien de ces animaux (`coutEntretien`).
— une somme correspondant au salaire annuel du ou des dresseurs de ces animaux (`salaires`).
Les projections de films sont des attractions qui sont de plus décrites par :
— le titre du film (`titre`).
— le coût annuel prévisionnel du matériel de projection (`coutMateriel`).
— le coût de location du film par spectateur, comme le prévoit le contrat entre le parc animalier et la médiathèque qui loue les films (`coutLocationSpect`).

Question 1. (4,5 points) Ecrivez en Java les trois classes représentant les attractions, munies uniquement de leurs attributs et des constructeurs avec paramètres. On ne devra pas pouvoir créer d'instances directes de la classe `Attraction`, seulement des instances des classes `Dressage` et `Projection`. Pour la suite, vous supposerez que les constructeurs sans paramètres et les accesseurs existent pour tous les attributs de ces trois classes.

```
// entête 0,5
public abstract class Attraction {
// trois attributs 0,5
    private String description;
    private int nbSpectateursAnnuel;
    private double tarifEntree;
// le constructeur 0,5
    public Attraction(String description, int nbSpectateursAnnuel, double tarifEntree) {
        this.description = description;
        this.nbSpectateursAnnuel = nbSpectateursAnnuel;
        this.tarifEntree = tarifEntree;
    }
...

// entête 0,5
public class SpectacleDressage extends Attraction {
// trois attributs 0,5
    private String typeAnimaux;
    private double coutAnimauxAnnuel;
    private double salaireDresseurAnnuel;
// le constructeur 0,5
```

```
    public SpectacleDressage(String description, int nbSpectateursAnnuel,
            double tarifEntree, String typeAnimaux, double coutAnimauxAnnuel,
            double salaireDresseurAnnuel) {
        super(description, nbSpectateursAnnuel, tarifEntree);
        this.typeAnimaux = typeAnimaux;
        this.coutAnimauxAnnuel = coutAnimauxAnnuel;
        this.salaireDresseurAnnuel = salaireDresseurAnnuel;
    }
...

// entête 0,5
public class ProjectionFilm extends Attraction {
// trois attributs 0,5
private String titreFilm;
private double coutAnnuelMateriel;
private double coutLocationFilmParSpectateur;
// le constructeur 0,5
public ProjectionFilm(String description, int nbSpectateursAnnuel,
double tarifEntree, String titreFilm, double coutAnnuelMateriel,
double coutAnnuelLocationFilm) {
super(description, nbSpectateursAnnuel, tarifEntree);
this.titreFilm = titreFilm;
this.coutAnnuelMateriel = coutAnnuelMateriel;
this.coutLocationFilmParSpectateur = coutAnnuelLocationFilm;
}
...
```

Question 2. (2 points) Ecrivez en Java les méthodes nécessaires au calcul du coût annuel (`coutAnnuel`) d'une attraction avec les informations suivantes :

- Elles retournent le coût annuel (elles ne l'affichent pas).
- Pour un spectacle de dressage, le coût annuel est donné par la somme du coût d'entretien annuel des animaux et des salaires annuels des dresseurs.
- Pour une projection de film, le coût annuel est donné par la somme : (1) du coût annuel du matériel de projection et (2) du coût annuel de location du film qui se calcule en fonction du nombre de spectateurs attendus par an.

```
// 1 Dans Attraction
public abstract double coutAnnuel();

// 0,5 Dans SpectacleDressage
public double coutAnnuel() {
  return coutAnimauxAnnuel+salaireDresseurAnnuel;
}

// 0,5 Dans Projection
public double coutAnnuel() {
  return coutAnnuelMateriel+coutLocationFilmParSpectateur*this.getNbSpectateursAnnuel();
}
```

Question 3. (2 points) On ajoute dans la classe `Attraction` une méthode `toString` écrite de la manière suivante (dans un style qui n'est pas dans l'esprit de l'approche objet) :

```
public String toString(){
   String res = " description="+this.description;
   if (this instanceof Dressage)
      res += "\n type d'animaux="+((Dressage) this).getTypeAnimaux();
   else if (this instanceof Projection)
      res += "\n titre"+((Projection) this).getTitre();
   return res;
}
```

Expliquez en quoi le style n'est pas dans l'esprit de l'approche objet et proposez une écriture alternative en redistribuant le code sur les différentes classes.

```
// 0,5 pour l'explication
La méthode est écrite avec des tests de type, ce qui demande d'anticiper tous les cas et est non extensible.

// 0,5 Dans Attraction
public String toString(){return "description="+this.description;}

// 0,5 Dans SpectacleDressage
public String toString(){
  return super.toString()+" - types d'animaux presentes="+this.typeAnimaux;
}

// 0,5 Dans Projection
public String toString(){
  return super.toString()+"titre="+this.titreFilm;
}
```

Question 4. (1,5 points) Ecrivez en Java, dans la classe `Attraction`, une méthode retournant la balance annuelle de l'attraction (`balanceAnnuelle`). Cette balance annuelle est obtenue en soustrayant au coût annuel de l'attraction ses recettes (calculés grâce au nombre de spectateurs et au tarif d'une entrée).

```
public double balanceAnnuelle() //0,5
{return coutAnnuel()-nbSpectateursAnnuel*tarifEntree;} // 1
```

3 Représentation du parc animalier

Un parc animalier (`ParcAnimalier`) a un nom et propose une liste d'attractions.

Question 5. (1,5) Ecrivez en Java l'entête et les attributs de la classe `ParcAnimalier`, en les initialisant.

```
public class ParcAnimalier { // 0,25
private String nom="inconnu"; //0,25
private ArrayList<Attraction> listeAttractions = new ArrayList<>(); //0,5+0,5
```

Question 6. (1,5 point) Ecrivez en Java, dans la classe `ParcAnimalier`, une méthode `existeDescription` prenant en paramètre une chaîne de caractères et retournant vrai si la liste d'attractions contient une attraction qui a cette chaîne de caractères comme description (faux sinon).

```
public boolean existeDescription(String d){ //0,25
   for (Attraction a : listeAttractions) //0,25
      if (a.getDescription().equals(d)) //0,5
         return true; //0,25
   return false; // 0,25
}
```

Question 7. (1,5) Ecrivez en Java, dans la classe `ParcAnimalier`, une méthode `ajoute` prenant en paramètre une attraction et l'ajoutant dans la liste d'attractions si une attraction de même description ne s'y trouve pas déjà (utilisez la méthode réalisée à la question précédente). Si une attraction de même description est déjà présente dans la liste, un message d'erreur est affiché et l'ajout n'est pas effectué.

```
public void ajout(Attraction a){ //0,25
   if (! this.existeDescription(a.getDescription())) // 0,5
      listeAttractions.add(a); // 0,5
   else
      System.out.println("une attraction de meme description est deja presente"); // 0,25
}
```

Question 8. (2 points) Ecrivez en Java, dans la classe `ParcAnimalier`, une méthode `attractionsBeneficiaires` retournant la liste des attractions dont la balance annuelle est positive ou nulle.

```
// à cause de la formulation du sujet ce sont en fait les attractions déficitaires ...
// mais supposons que l'on respecte l'énoncé
public ArrayList<Attraction> attractionsBeneficiaires(){ //0,5
   ArrayList<Attraction> resultat = new ArrayList<Attraction>(); //0,5
   for (Attraction a : listeAttractions) //0,25
      if (a.balanceAnnuelle() >=0) //0,25
         resultat.add(a); //0,25
      return resultat; //0,25
  }
```

Question 9. (2 points) Ecrivez en Java, dans la classe `ParcAnimalier`, une méthode `balanceAnnuelleGlobale` retournant la somme des balances des attractions.

```
public double balanceAnnuelleGlobale(){ // 0,5
   double balance = 0; //0,5
   for (Attraction a : listeAttractions) //0,25
      balance += a.balanceAnnuelle(); //0,5
   return balance;//0,25
}
```

Question 10. (1,5 points) Ecrivez en Java, dans la classe `ParcAnimalier`, une méthode `beneficiaire` retournant vrai si le parc animalier est bénéficiaire (sa balance annuelle globale est positive ou nulle), faux sinon.

```
// à cause de la formulation du sujet, c'est en fait quand c'est déficitaire ...
// mais supposons que l'on respecte l'énoncé
public boolean beneficiaire(){ //0,5
   return balanceAnnuelleGlobale() >= 0; // 1
}
```