BTS SERVICES INFORMATIQUES AUX ORGANISATIONS

E5 : Production et fourniture de services informatiques

SESSION 2020

Cas Coopain

Eléments de correction

Barème :

Mission 1	Gestion des demandes d'insémination	40 points
Mission 2	Nouvelles fonctionnalités de l'application Android *MobiSemin*	15 points
Mission 3	Gestion des tournées des inséminateurs	35 points
Mission 4	Calcul du retour sur investissement	10 points
	Total	100 points

Mission 1 – Gestion des demandes d'insémination (40 points)

D1.1 Analyse de la demande
D4.1 Conception et réalisation d'une solution applicative
• Conception ou adaptation d'une base de données

Question 1.1 : Déduire de la requête *SQL* fournie dans le dossier documentaire, la règle de gestion utilisée par Coopain pour choisir une paillette lors d'une demande d'insémination.

La requête SQL présentée permet d'exprimer la règle de gestion suivante :

Pour le taureau et le type de paillette demandés, la paillette sera choisie dans un des lots de la collecte la plus ancienne pour lesquels il reste des paillettes en stock.

Question 1.2 : Indiquer pour chaque information à imprimer sur une paillette, la ou les données nécessaires en précisant pour chacune d'où elle provient dans le modèle existant et si elle nécessite une transformation.

- Date de collecte : issue de l'attribut **date** de la table **Lot** ou **Collecte**
- N° d'agrément du CCS : **non disponible** dans la base de données (il s'agira d'une constante fixée avec le numéro d'agrément de Coopain)
- N° national d'identification du taureau : issu de l'attribut **idNationalTaureau** de la table **Lot** ou **Collecte** ou **Taureau**
- N° IE du taureau : issue de l'attribut **numeroIE** de la table **Taureau**
- Quantième date de collecte : calcul du nombre de jours entre le **01/01/2002** et l'attribut **date** de la table **Lot**
- Numéro de lot : issu de l'attribut **numero** de la table **Lot** ou **Collecte**
- Le code de la race : issue de la clé étrangère **codeRace** de la table **Taureau**
- Le nom du taureau : issu de l'attribut **nom** de la table **Taureau**
- Le code à barres : généré à partir des attributs **numeroIE** (table **Taureau**), calcul du nombre de jours entre le **01/01/2002** et l'attribut **date** (table **Lot ou Collecte**), **numero** (table **Lot**).

Question 1.3 : Écrire la requête permettant d'obtenir pour le taureau FR0103015562 les tarifs des types de paillette (libellé du type de paillette, prix de la paillette).

```
SELECT libelle, prixPaillette
FROM Tarif
JOIN TypePaillette ON idTypePaillette = id
WHERE idNationalTaureau = 'FR0103015562'
```

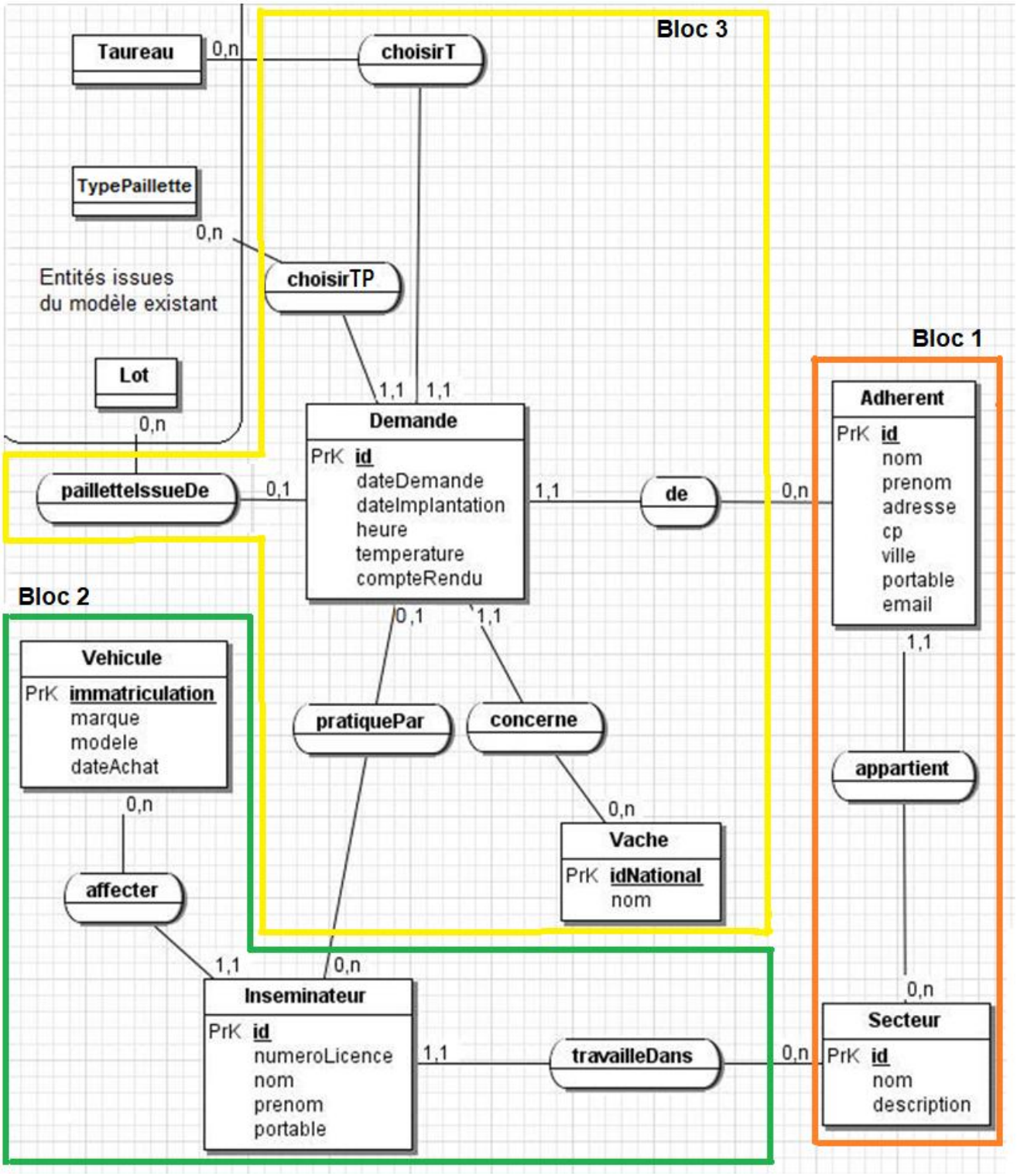
Question 1.4 : Écrire la requête permettant d'obtenir la liste des taureaux (identifiant national, nom, stock total restant de paillettes), le tout trié de façon croissante par stock total restant.

```
SELECT T.idNational, nom, SUM (stockRestant) as StockTotal
FROM Taureau T
JOIN Lot L ON T.idNational = L.idNationalTaureau
GROUP BY T.idNational, nom
ORDER BY StockTotal
```

On acceptera la solution de passer par la table Collecte.

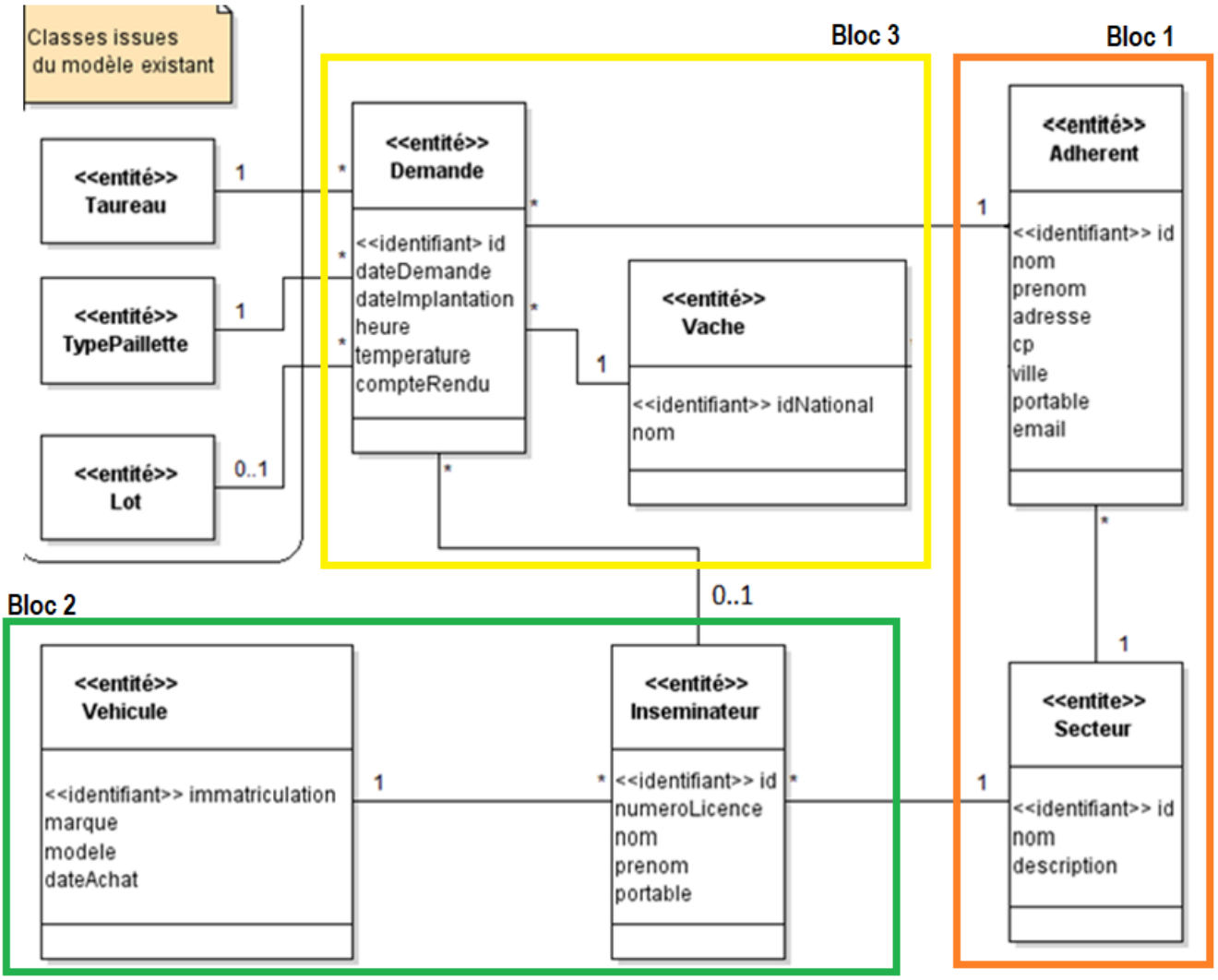
Question 1.5 : Présenter le modèle de données nécessaire à la gestion des demandes d'insémination en reprenant les éléments nécessaires du modèle existant.

Modèle entité-association



On pourra trouver une association "appartient" entre les entités Vache et Adherent.
On ne pénalisera pas la cardinalité 1,1 (au lieu de 0,1) entre Demande et Inseminateur
Toute solution pertinente d'identification d'une demande sera acceptée

Diagramme de classes



On pourra trouver un lien entre les classes Vache et Adherent.

On ne pénalisera pas la multiplicité 1 (au lieu de 0..1) entre Demande et Inseminateur

Toute solution pertinente d'identification d'une demande sera acceptée

Mission 2 – Nouvelles fonctionnalités de l'application Android *MobiSemin* (15 points)

D2.3 Gestion des problèmes et des changements
D4.1 Conception et réalisation d'une solution applicative
• Développement, utilisation ou adaptation de composants logiciels

Question 2.1 Compléter les scénarios d'erreur du cas d'utilisation « Saisir une tournée » concernant la saisie des kilomètres d'une tournée.

Fiche descriptive du cas d'utilisation (use case) « Saisir une tournée »

Pré-condition

L'utilisateur devra être authentifié

Scénario nominal

1. Le système présente un formulaire avec un calendrier initialisé avec la date du jour, la liste des véhicules disponibles et son véhicule affecté par défaut.
2. L'utilisateur met éventuellement à jour les données (choix d'un autre véhicule et/ou d'une autre date) et clique sur le bouton valider du formulaire.
3. Le système vérifie les données saisies et présente un formulaire de saisie des kilomètres.
4. L'utilisateur renseigne le kilométrage au compteur de fin de tournée et clique sur le bouton valider du formulaire.
5. Le système vérifie les données saisies, enregistre toutes les données, informe l'utilisateur que les données ont été enregistrées et renvoie sur la page d'accueil.

Scénarios alternatifs

2.a L'utilisateur abandonne la saisie. Fin.
4.a L'utilisateur abandonne la saisie. Fin.

Scénarios d'erreur

3.a La date saisie est supérieure à la date du jour, le système en informe l'utilisateur et retourne à l'étape 1.

5.a le nombre de kilomètres saisi n'est pas numérique. Le système retourne à l'étape 3.

5.b l'utilisateur n'a pas saisi de nombre de km de fin. Le système en informe l'utilisateur et retourne à l'étape 3.

5.c l'utilisateur a saisi un nombre de km de fin inférieur au nombre de km de début. Le système retourne à l'étape 3.

Post-condition

Les données d'une tournée sont enregistrées en local sur le mobile.

On acceptera :
- D'autres tests s'ils sont pertinents
- 4.a en lieu et place de 5.a
- 4.b en lieu et place de 5.b
- 4.c en lieu et place de 5.c

On vérifie la cohérence des réponses

Question 2.2
Compléter la méthode *onCreate()* de la classe MobiDb.

```
// Méthode pour créer la table histoKm : onCreate()
@Override
public void onCreate(SQLiteDatabase db){

  // la table histoKm contient 6 champs qui sont l'id, la date et les
  // kilomètres effectués à cette date, ainsi que les clés étrangères faisant
  // références aux inséminateurs et aux véhicules :

  String strReq;
  strReq = "CREATE TABLE histoKm(";
  strReq += " id INTEGER PRIMARY KEY AUTOINCREMENT,";
  strReq += " date TEXT,";
  strReq += " kmDebut INTEGER,";
  strReq += " kmFin INTEGER,";
  strReq += " idInsemin INTEGER,";
  strReq += " idVehicule INTEGER ,";
  strReq += " FOREIGN KEY(idInsemin) REFERENCES inseminateur (id),";
  strReq += " FOREIGN KEY(idVehicule) REFERENCES vehicule (id))";

  db.execSQL(strReq);
}
```

On pourra trouver la présence de la clause FOREIGN KEY après la description de chaque attribut concerné.

Mission 3 – Gestion des tournées des inséminateurs (35 points)

D4.1 Conception et réalisation d'une solution applicative
• Développement, utilisation ou adaptation de composants logiciels

Question 3.1

3.1.1 Expliquer le mécanisme qui permet de différencier les deux méthodes de même nom (*CATournee*) de la classe GestionTournée.

On attend ici la notion de **surcharge**, les méthodes sont différenciées par leur signature (nombre des paramètres et type des paramètres). On acceptera une réponse qui décrit le mécanisme sans mentionner la surcharge.

3.1.2 Écrire la méthode *getCoordGPS()* de la classe Adherent qui renvoie les coordonnées GPS (latitude et longitude) sous la forme d'une chaîne de caractères.

```
public String getCoordGPS() {

      return this.latitude + "  " + this.longitude ;

}
```

3.1.3 Écrire le code de la méthode *getAdherents()* de la classe GestionTournee

```
public ArrayList<Adherent> getAdherents() {
      ArrayList<Adherent> adherents = new ArrayList<Adherent> ();
      for (Visite v : this.tournee.getLesVisites()) {
            adherents.add(v.getLeAdherent());
      }
      return adherents;
}
```

3.1.4 Écrire le code de la méthode *montantAFacturer()* de la classe Visite.

```
public float montantAfacturer() {
      float montant = 0;
      for (PrestationVisite pv : this.lesPrestationsVisite) {
            montant += pv.getNombreActes() *
pv.getLeTypePrestation().getPrixForfaitaire();
      }
      return montant;
}
```

Question 3.2

Compléter les tests unitaires correspondants à la méthode *CATournee()* de la classe GestionTournee en respectant le scénario fourni par le chef de projet et en respectant la syntaxe utilisée.

Tout usage cohérent des méthodes d'assertion sera valorisé.

```
public class GestionTourneeTest {
      private Inseminateur ins1;
      private Adherent adh1, adh2;
      private Visite v1, v2;
      private TypePrestation tp1, tp2;
      private Tournée t1;
      private GestionTournee gt;

            // Méthode setUp() automatiquement exécutée
            // avant chaque appel de la méthode de test
      @Before
      public void setUp() throws Exception {
          tp1 = new TypePrestation("Insémination", 100);
          tp2 = new TypePrestation("Echographie", 10);
          ins1 = new Inseminateur("Petit", "Ferdinand", "0600000100");
          adh1 = new Adherent("Duboeuf", "Georges" , "0600000020", "48.5331",
"7.72");
          adh2 = new Adherent("Cow", "Marguerite", "0600000003", "48.5833",
"7.75");
          t1 = new Tournee(new SimpleDateFormat("dd-MM-yyyy").parse("05-06-2020"),
ins1);
          gt = new GestionTournée(t1) ;
         // Préparation des visites

            v1 = new Visite(adh1, "09 :00") ;
            v1.ajouterPrestationVisite(tp1, 3) ;
            v1.ajouterPrestationVisite(tp2, 1) ;
            v2 = new Visite(adh2, "10:30") ;
            v2.ajouterPrestationVisite(tp1, 1) ;
      }

      @Test
      public void testCATournee() {
            assertEquals("Montant nul attendu", 0, gt.CATournee()) ;
            t1.ajouterVisite(v1) ;
            assertEquals("somme à facturer après 1ere visite",310 ,
gt.CATournee()) ;
            t1.ajouterVisite(v2) ;
            assertEquals("somme à facturer finale",410 , gt.CATournee()) ;

      }
}
```

Autre solution : une visite peut être rajoutée au niveau de l'objet gt

Mission 4 – Calcul du retour sur investissement (10 points)

D5.1 – Gestion des configurations
A5.1.6 Évaluation d'un investissement informatique

Question 4.1

Calculer le retour sur investissement (*ROI*) de la mise en place des nouvelles fonctionnalités ; donner et expliquer le détail du calcul.

Tout d'abord il faut calculer l'économie faite sur la différence de km parcourus entre l'année n-1 et l'année n de mise en service des nouvelles fonctionnalités permettant la remontée et le contrôle des km parcourus :

Il s'agit de 5% des 2 000 000 km en moins, soit 100 000 km. L'économie est de 100 000 fois le prix du km (0.4), soit 40 000 €.

Ensuite on évalue, selon les critères indiqués dans le devis, le coût de développement des nouvelles options du logiciel :

- 6 jours de CAE pour l'analyse et les modifications de la base de données, soit 6 x 600 = 3600 €
- 13 jours de AP pour le développement des différentes options, soit 13 x 400=5200 €
- 3 jours de AP pour les tests, soit 3 x 400 = 1200€

Au total les nouvelles fonctionnalités seront facturées 3600 + 5200 + 1200 = 10000 €

Le ROI pour cet investissement est donc de : (40000 - 10000) / 10000 soit 3 %.

Question 4.2

Justifier dans une courte note la pertinence de cet investissement, connaissant la durée de rentabilité.

On constate que le Roi est important et la durée de rentabilité est très courte (3 mois), ce qui permet de dire sans ambiguïté qu'il fallait faire cet investissement.

Ceci valorise aussi l'image de marque de l'entreprise.

Pour information le calcul des 92 jours a été déterminé de la façon suivante :

Point mort = Investissement / gain

soit 10000 / 40000 = 0.25,

365 jours x 0.25 = 92 jours.