Durée : 1h30 Le 02/02/2015

CONTRÔLE FINAL
MASTER II - SRI

Questions de cours (5 pts)

1. Est-ce que les SGBDOO supportent l'héritage multiple ?
2. Est-ce qu'un objet composant peut être partagé entre plusieurs objets de plusieurs classes composites ?
3. Est-ce que tous les SGBOO acceptent la contrainte inverse ?
4. Quel est le but principal d'avoir l'identifiant d'un objet dans un SGBDOO permanent, fixe et unique dans la base et dans le temps ?
5. Expliquer l'instruction suivante : **CS Is-a CG**.

Modélisation (15 pts)

Une entreprise aérienne enregistre dans sa base de données pour chaque client son numéro de client, son nom et prénom et son adresse. A un vol on associe un numéro de vol, une ville origine, une ville destination, une date, un type d'appareil, nombre de places libres en classe économique, nombre de places libres en classe affaire, nombre de places libres en première classe et nombre total de places du vol.

Quand un client effectue une réservation, il choisit un vol et une classe (économique, affaire, première).

1. Ecrivez en java les classes nécessaires, avec leurs attributs, pour modéliser le système de réservation de cette compagnie aérienne. La classe principale sera nommée « **BOOKINGTOOL** ».
2. Ecrivez la méthode **isFlightFull()** qui permet de vérifier si un vol est plein.
3. Ecrivez la méthode **getNbFreeSeats()** qui retourne le nombre de places libres dans un vol.
4. Ecrivez la méthode **lastBookingId()** qui retourne le dernier numéro de réservation dans la base de données.
5. Ecrivez la méthode **toBook()** qui permet d'établir une réservation pour un client.
6. Ecrivez les méthodes **cancelBooking()** qui permet de d'annuler une réservation d'un client.

Remarque : Pour les méthodes précédentes, vous devez choisir les paramètres nécessaires et les classes adéquates.

7. Ecrivez la méthode **main()** de la classe **BOOKINGTOOL** qui fait les travaux suivants :
   a. Créer un vol avec 200 places vides (10 dans la classe affaire, 10 dans la 1ère classe et 180 dans la classe économique) avec "**Batna"** comme ville d'origine, **"Alger"** comme ville d'arrivée, **"Boeing 747"** comme type d'appareil et une date **"01/01/2015"**.
   b. Faites 3 réservations pour 3 clients différents en classe économique.
   c. Faites annuler la réservation du dernier client.
   d. Afficher la liste des clients de ce vol.

Bonne chance…

NB: Le corrigé type vous le trouverez sur le site :
http://www.larbiguezouli.com

CORRECTION DU CONTRÔLE FINAL
MASTER II - SRI

Questions de cours (5 pts)

1. Est-ce que les SGBDOO supportent l'héritage multiple ?
   Pas tous les SGBDOO supportent l'héritage multiple.
2. Est-ce qu'un objet composant peut être partagé entre plusieurs objets de plusieurs classes composites ?
   Non, un objet composant peut être partagé entre plusieurs objets de la même classe composite.
3. Est-ce que tous les SGBOO acceptent la contrainte inverse ?
   Quelques SGBDO acceptent la contrainte inverse: l'objet composite dépend de ses objets composants.
4. Quel est le but principal d'avoir l'identifiant d'un objet dans un SGBDOO permanent, fixe et unique dans la base et dans le temps ?
   Ces caractéristiques de l'identifiant permettent à plusieurs objets de partager le même objet composant.
5. Expliquer l'instruction suivante : **CS Is-a CG**.
   CS représente un sous-ensemble de CG. Du point de vue conceptuel, l'ensemble des objets de CS est inclut dans celui de CG.

Modélisation (15 pts)

Une entreprise aérienne enregistre dans sa base de données pour chaque client son numéro de client, son nom et prénom et son adresse. A un vol on associe un numéro de vol, une ville origine, une ville destination, une date, un type d'appareil, nombre de places libres en classe économique, nombre de places libres en classe affaire, nombre de places libres en première classe et nombre total de places du vol.

Quand un client effectue une réservation, il choisit un vol et une classe (économique, affaire, première).

1. Ecrivez en java les classes nécessaires, avec leurs attributs, pour modéliser le système de réservation de cette compagnie aérienne. La classe principale sera nommée « **BOOKINGTOOL** ». (1.5 pts)
2. Ecrivez la méthode **isFlightFull()** qui permet de vérifier si un vol est plein. (1.5 pts)
3. Ecrivez la méthode **getNbFreeSeats()** qui retourne le nombre de places libres dans un vol. (1.5 pts)
4. Ecrivez la méthode **lastBookingId()** qui retourne le dernier numéro de réservation dans la base de données. (1.5 pts)
5. Ecrivez la méthode **toBook()** qui permet d'établir une réservation pour un client. (1.5 pts)
6. Ecrivez les méthodes **cancelBooking()** qui permet de d'annuler une réservation d'un client. (1.5 pts)

Remarque : Pour les méthodes précédentes, vous devez choisir les paramètres nécessaires et les classes adéquates.

7. Ecrivez la méthode **main()** de la classe **BOOKINGTOOL** qui fait les travaux suivants :
   a. Créer un vol avec 200 places vides (10 dans la classe affaire, 10 dans la 1ère classe et 180 dans la classe économique) avec "**Batna**" comme ville

d'origine, **"Alger"** comme ville d'arrivée, **"Boeing 747"** comme type d'appareil et une date **"01/01/2015"**. (1.5 pts)

b. Faites 3 réservations pour 3 clients différents en classe économique. (1.5 pts)
c. Faites annuler la réservation du dernier client. (1.5 pts)
d. Afficher la liste des clients de ce vol. (1.5 pts)

Réponse

```
public class CLIENT {
	public int Id;
	public String Nom;
	public String Prenom;
	public String Adresse;

	CLIENT(int idClient) {
		Id = idClient;
	}

	CLIENT(int idClient, String n, String pn, String adr) {
		Id = idClient;
		Nom = n;
		Prenom = pn;
		Adresse = adr;
	}
}

import java.sql.Date;

public class FLIGHT {
	public int Id;
	public String Origine;
	public String Destination;
	public String DateVol;
	public String TypeAppareil;
	public int NbPlaces;
	public int NbClasseEconomiqueLibre;
	public int NbClasseAffaireLibre;
	public int NbPremiereClasseLibre;

	FLIGHT() {

	}

	FLIGHT(int IdFlight, String org, String dest, String dateV, String typeA,
int nbP, int nbCE, int nbCA, int nbPC) {
		Id = IdFlight;
		Origine = org;
		Destination = dest;
		DateVol = dateV;
		TypeAppareil = typeA;
		NbPlaces = nbP;
		NbClasseEconomiqueLibre = nbCE;
		NbClasseAffaireLibre = nbCA;
		NbPremiereClasseLibre = nbPC;
	}

	boolean isFlightFull() {
```

```
            if (NbClasseEconomiqueLibre == 0 && NbClasseAffaireLibre == 0 &&
NbPremiereClasseLibre == 0) return true;
            return false;
      }

      int getNbFreeSeats() {
            return NbClasseEconomiqueLibre + NbClasseAffaireLibre +
NbPremiereClasseLibre;
      }

}

public class BOOKING {
      public int Id;
      public int IdClient;
      public int IdFlight;
      public int Classe; // 0:économique, 1:affaire, 2:1ère classe

      BOOKING() {
      }

      BOOKING(int newId) {
            Id = newId;
      }

      void toBook(int IdC, int IdF, int C) {
            IdClient = IdC;
            IdFlight = IdF;
            Classe = C;
      }
}

import java.io.File;
import java.sql.Date;

import com.db4o.*;

public class BOOKINGTOOL {
      public static String DBOFILENAME = "c:/bookingtool.dbo"; // Nom du fichier
contenant la base de données orientée objet

      static int lastBookingId(ObjectContainer db, int IdFlight) {
            int ret = 0;
            BOOKING myBooking = new BOOKING();
            myBooking.IdFlight = IdFlight;
            ObjectSet<BOOKING> result = db.queryByExample(myBooking);
            for (int i = 1; i <= result.size(); i++) {
                  myBooking = result.next();
                  if (myBooking.IdFlight == IdFlight && ret < myBooking.Id)
                        ret = myBooking.Id;
            }
            return ret;
      }

      static void cancelBooking(ObjectContainer db, int IdBooking) {
            BOOKING myBooking = new BOOKING(IdBooking);
            ObjectSet<BOOKING> result = db.queryByExample(myBooking);
            if (result.size() > 0) {
                  myBooking = result.next();
```

```
                db.delete(myBooking);
            }
        }

        public static void main(String[] args) {
            // Préparation de la base de données objet
            new File(DBOFILENAME).delete();
            ObjectContainer db = Db4o.openFile(DBOFILENAME); // Ouverture de la
base de données objet, si elle n'existe pas elle sera créée
            try {
                //Créer le vol
                FLIGHT myFlight = new FLIGHT(1, "Batna", "Alger", "01/01/2015",
"Boeing 747", 200, 180, 10, 10);
                db.store(myFlight);

                //Créer les 3 clients
                CLIENT myClient1 = new CLIENT(1, "A", "B", "C");
                db.store(myClient1);
                CLIENT myClient2 = new CLIENT(2, "D", "E", "F");
                db.store(myClient2);
                CLIENT myClient3 = new CLIENT(3, "G", "H", "I");
                db.store(myClient3);

                //Chercher le dernier identifiant des reservations dans le vol
numéro 1
                int lastId = lastBookingId(db, 1);

                //Créer les réservations
                BOOKING myBooking1 = new BOOKING(lastId+1);
                myBooking1.toBook(myClient1.Id, myFlight.Id, 0);
                db.store(myBooking1);
                BOOKING myBooking2 = new BOOKING(lastId+2);
                myBooking2.toBook(myClient2.Id, myFlight.Id, 0);
                db.store(myBooking2);
                BOOKING myBooking3 = new BOOKING(lastId+3);
                myBooking3.toBook(myClient3.Id, myFlight.Id, 0);
                db.store(myBooking3);

                //Annuler la réservation du client numéro 3
                cancelBooking(db, myBooking3.Id);

                //Lister les client du vol myFlight
                BOOKING myBooking = new BOOKING();
                myBooking.IdFlight = myFlight.Id;
                ObjectSet<BOOKING> result = db.queryByExample(myBooking);
                for (int i = 1; i <= result.size(); i++) {
                    myBooking = result.next();
                    CLIENT myClient = new CLIENT(myBooking.IdClient);
                    ObjectSet<CLIENT> result1 = db.queryByExample(myClient);
                    if (result1.size() > 0) {
                        myClient = result1.next();
                        System.out.println("Client: Nom: " + myClient.Nom
+ " Prénom: " + myClient.Prenom + " Adresse: " + myClient.Adresse);
                    }
                }


            } finally {
                db.close();
```

```
                }

        }
}
```

Pour télécharger la solution de cette partie cliquez sur ce lien :
http://www.larbiguezouli.com/Fichiers/Enseignement/MasterSRI/bdoo/Controle_Final_Correction_2014_2015_Modelisation.zip