Exercices dirigés
séance n°5 - corrigé

Exercice 1 : contrôle de parité

On suppose qu'une matrice 8*8 (tableau à 2 dimensions) contient une séquence de 8 octets (chaque octet représentant un caractère ASCII 7 bits).
Cette séquence d'octets est le résultat d'une transmission de données. On souhaite détecter les éventuelles erreurs de transmission. Pour cela, on met en place un "contrôle de parité".
Le premier bit de chaque octet représente le bit de parité. Celui-ci vaut 0 si le nombre de 1 est impair, 1 s'il est pair. La première colonne contient donc les valeurs de parité de chaque ligne

Notes

- Les données utiles sont donc représentées par une matrice 8*7. La première colonne n'apporte pas d'information (sinon la parité).
- Les caractères codés en ASCII 7 bits sont les caractères usuels à l'exclusion des caractères accentués.

Question 1

On veut écrire un programme qui évalue l'intégrité de données sur chaque ligne sur la base de la parité et affiche le résultat.

Construire l'algorithme par raffinements successifs.

Question 2

Coder l'algorithme en Java. Chaque niveau de raffinement est codé par une fonction.

--- Exercice 1 ------ Solutions ---------- --

Algorithme

On construit la solution par raffinements successifs :

```
raffinement niveau 1 : Calcul et affichage de la parité d'une d'une
séquence de 8 caractères ASCII 7 bits
 début
  Initialisation de la matrice représentant la séquence des 8 caractères
   ASCII;
  Calcul de la parité des lignes;
  Affichage de la parité des lignes;
 fin

raffinement niveau 2.1: Calcul de la parité des lignes
 début
   faire 8 fois
     Calculer la parité d'une ligne;
   fin faire
 fin

raffinement niveau 3.2.1: Calcul de la parité d'une ligne
 début
   Compter le nombre de 1 dans la ligne (les 7 bits significatifs);
   si le nombre de 1 est pair
   alors
      si  le bit de parité vaut 1
      alors les données sont intègres
      sinon erreur
      fin si;
   sinon
      si  le bit de parité vaut 0
      alors les données sont intègres
      sinon erreur
      fin si;
  fin si
 fin

raffinement niveau 2.2 : Affichage de la parité des lignes
 début
   faire 8 fois
      Afficher la parité d'une ligne;
   fin faire
 fin

raffinement niveau 4.3.2.1 : Compter le nombre de 1 dans la ligne (les 7
bits significatifs)
 début
   faire 7 fois
      cumuler l'élément courant;
   fin faire
 fin
```

Programme

```
import static java.lang.System.*;
public class Parite{
  public static void main(String[] args){
      new Parite();
  }

  Parite(){
      // initialisation de la matrice représentant
      // la séquence des 8 caractères ASCII
     int[][] matrice= {{0,1,1,0,1,1,0,1},
                       {0,1,1,0,1,1,0,1},
                       {1,1,0,0,1,1,1,1},
                       {0,0,1,0,0,1,0,1},
                       {1,1,0,0,1,1,0,1},
                       {1,1,1,0,1,1,1,1},
                       {0,1,0,0,0,1,0,1},
                       {0,0,1,0,1,1,0,1}};
      // calcul de la parité des lignes
      boolean[] tableauParites = paritesDesLignes( matrice );
      // affichage de la parité des lignes
      println( tableauParites );
    }

    // raffinement niveau 2 : calcul de la parité des lignes
    boolean[] paritesDesLignes( int[][] matrice ){
        boolean[] tableauParites = new boolean[matrice.length];
        for( int i=0;i<matrice.length;i++ )
            tableauParites[i] = pariteDUneLigne( matrice[i] );
        return tableauParites;
    }

    // raffinement niveau 3 : calcul de la parité d'une ligne
    boolean pariteDUneLigne( int[] ligne ){
        if( nombreDeUns( ligne )%2 == 1 )
            return (ligne[0]==0)? true:false;
        else
            return (ligne[0]==0)? false:true;
    }

    // raffinement niveau 4 : comptage du nombre de "uns" dans une ligne
    int nombreDeUns( int[] ligne ){
        int uns = 0;
        for( int i= 1;i<ligne.length;i++ ){
            if( ligne[i]==1 )
                  uns++;
        return uns;
    }

    // raffinement niveau 2 : affichage de la parité des lignes
    void println( boolean[] parites ){
        out.println("\f");
        out.println("------ Parité des lignes ------");
        for( int i=0;i< parites.length;i++ )
            out.println("intégrité de la ligne "+i+" = "+ parites[i]);
    }
}
```

Exercice 2 : réservation d'un siège d'avion

Une petite compagnie d'aviation ne possédant qu'un seul avion de 10 sièges souhaite disposer d'un système de réservation.
Le programme permet à un utilisateur, à partir d'un état de réservation de l'avion quelconque, de réserver un ou plusieurs sièges pour le prochain vol.
L'avion comporte deux classes, première et économique. 5 sièges sont affectés à la classe économique et les 5 autres à la première classe.
Selon le choix de l'utilisateur, un siège est affecté en première ou en classe économique. Toutefois, si la classe économique est complète, le système demande à l'utilisateur s'il souhaite un siège en première classe. Inversement si la première classe est complète. Dans la négative, le système affiche le message suivant : "prochain vol dans 3 heures".
A l'arrêt du système l'état de la réservation sera affiché.

Question 1

Ecrire l'algorithme en utilisant la méthode de décomposition par raffinements successifs.

Question 2

Ecrire le programme correspondant.

--- Exercice 2 ------ Solutions ----------

Algorithme

```
début
      faire
            si l'avion est plein
            alors message : avion plein, réservation impossible!;
            sinon
                  saisir le choix utilisateur;
                  assigner un siège à l'utilisateur;
      tant que "nouvelle demande";
      afficher l'état de réservation de l'avion;
fin.

raffinement niveau 2 : assigner un siège à l'utilisateur
début
      si choix première classe
      alors
            si siège libre alors choisir un siège (en première classe);
            sinon
                  affecter un siège alternatif;
            fin si;
      sinon // choix classe eco
            si siège libre alors choisir un siège (en classe eco);
            sinon
                  affecter un siège alternatif;
            fin si;
fin.
```

```
raffinement niveau 3 : affecter un siège alternatif
début
      proposer à l'utilisateur des sièges dans l'autre classe;
      si l'utilisateur est d'accord
      alors
            choisir un siège (dans l'autre classe);
      sinon
            message : "prochain vol dans 3 heures";
      fin si;
fin.

raffinement niveau 3 : siège libre?
début
      pour chaque siège faire
            si le siège courant est libre
            alors retourner siège libre;
            fin si;
      fin pour;
      retourner pas de siège libre;
fin.


raffinement niveau 4 : choisir un siège
début
      pour chaque siège à proposer faire
            si le siège courant est libre
            alors retourner son numéro;
            fin si;
      fin pour;
fin.

raffinement niveau 2 : avion plein
début
      pour chaque siège faire
            s'il est libre
            alors avion non plein;
            fin si;
      fin pour;
      avion plein;
fin.

raffinement niveau 2 : afficher l'état de réservation de l'avion
début
      afficher en-tête;
      pour chaque siège faire
            afficher n° du siège et état d'affectation;
      fin pour;
fin.
```

Programme

```
import javax.swing.JOptionPane;
import java.lang.NumberFormatException;
public class Reservation{
  private static final int ECO=2;
  private static final int FIRST=1;
  private boolean[] sieges =
             {false,false,false,false,false,false,false,false,false,false};

  public static void main(String[] args){
      new Reservation(sieges);
  }

  private Reservation(boolean[] sieges){
      this.sieges = sieges;
      boolean continuer = true;
      int choix;
      do{
          choix = saisirChoixUtilisateur
                       ("1ère classe, taper 1\n2ème classe, taper 2");
          if( choix!=FIRST && choix!=ECO )
                       printMessage(" Choix incorrect, recommencez!");
          else{
              if( avionPlein(sieges) ){
                    printMessage(" Réservation impossible, avion plein!");
                    continuer = false;
              }
              else{
                    boolean[] tab = assignerSiege(
                        sieges,choix,sieges.length/2-1,sieges.length-1
                       );
                    if( tab==null )
                              printMessage("Prochain vol dans 3 heures");
                    else sieges = tab;
              }
              printReservation( sieges );
             continuer = demander(" Voulez vous continuer  ?");
            }
      }while( continuer);

  }

  /**
   * Assigne un siège au client<br>
   * @param sieges  le tableau représentant l'occupation des sièges
   * @param choix   représente le choix de la classe (1ère ou eco)
   * @param i       indice du dernier siège de la 1ère classe
   * @param j       indice du dernier siège de la classe ECO
   * @return        le tableau de la nouvelle occupation des sièges.
   *                S'il la valeur retournée est nulle, aucun siège
   *               n'a pu être attribué<br>
   * <b>Préconditions</b><br>
   *    On suppose que le paramètre choix prend les valeurs 1 ou 2
   */
  boolean[] assignerSiege( boolean[] sieges, int choix, int i, int j ){
      if( choix == 1 || choix == 2 ){
        boolean[] affectation = new boolean[sieges.length];
        for( int k=0;k<sieges.length;k++)
          affectation[k] = sieges[k];
        int numero;
```

```
      switch( choix ){
        case FIRST : if( siegeLibre( sieges,0,i ) )
                       numero = choisirSiege( sieges,0,i );
                     else
                       if
                       (demander("Voulez vous un siège en classe ECO?") )
                        numero = choisirSiege(sieges,i+1,j);
                       else
                        return null;
                     affectation[numero]=true;break;
        case ECO : if( siegeLibre( sieges,i+1,j ) )
                       numero = choisirSiege( sieges,i+1,j );
                   else
                     if
                  (demander("Voulez vous un siège en première classe ?"))
                          numero = choisirSiege(sieges,0,i);
                     else
                       return null;
                   affectation[numero]=true;break;
        default : printMessage
                  ("votre choix est erroné, veuillez recommencer!");break;
    }
    return affectation;
  }else // Exception
}

 /**
  * Détermine si un siège est libre ou non dans la partie
  * du tableau sieges[i..j]
  * @param    sieges  le tableau représentant l'occupation des sièges
  * @param    i       borne inférieure du tableau à visiter
  * @param    j       borne supérieure du tableau à visiter
  * @return   true si un siège est libre dans siege[i..j], false sinon
  * <b>Préconditions<b><br>
  *   i>= 0 et j<sieges.length
  */
 boolean siegeLibre( boolean[] sieges,int i,int j){
     if( i>=0 && j<sieges.length ){
          for( int k=i;k<=j;k++)
                if( !sieges[k] ) return true;
          return false;
     }else // Exception
  }


 /**
  * Détermine le numéro du siège à réserver dans la partie
  * du tableau sieges[i..j]
  * @param    sieges  le tableau représentant l'occupation des sièges
  * @param    i       borne inférieure de la zone de réservation
  *                   à visiter
  * @param    j       borne supérieure de la zone de réservation
  *                   à visiter
  *
  * <b>Préconditions<b><br>
  *   i>= 0 et j<sieges.length
  */
 int choisirSiege( boolean[] sieges,int i,int j ){
    if( i>+0 && j<sieges.length ){
     int s = i;
     for( int k=i;k<=j;k++)
```

```
        if( !sieges[k] ){ s=k;break; }
    return s;
  }else // Exception
 }

 /**
  * Affichage du plan de réservation des sièges de l'avion
  * @param   sieges  le tableau représentant les sièges de l'avion
  */
 void printReservation( boolean[] sieges ){
     String panneau = " n∞ des sièges     disponibilité\n";
     for(int i=0;i<sieges.length;i++)
         if(i==sieges.length-1)
             panneau=
   panneau+"    "+(i+1)+"                          "+ sieges[i]+"\n";
         else
             panneau =
   panneau+"    "+(i+1)+"                            "+ sieges[i]+"\n";
     printMessage(panneau);
}

/**
  * Détermine si l'avion est plein ou non
  * @param   sieges  le tableau représentant les sièges de l'avion
  * @return  true si l'avion est plein, false sinon
  *
  */
 boolean avionPlein( boolean[] sieges ){
    boolean reponse = true;
    for(int k=0;k<sieges.length;k++)
        if( !sieges[k] ) return false;
    return reponse;
}

/**
 * Remarque : l'exception NumberFormatException n'est pas traitée.
 * => on ne traite pas le cas d'erreur où l'utilisateur taperait
 *    autre chose qu'un entier
 * @param message    Le message invitant l'utilisateur à entrer son
 *   choix
 * @return un entier représentant le choix de l'utilisataeur
 */
 int saisirChoixUtilisateur( String message ){
     do{
         String demande = JOptionPane.showInputDialog(
             null,
             message,
            "Système de réservation de la compagnie des vols parisiens",
            JoptionPane.QUESTION_MESSAGE
          );
         try{
             int choix = Integer.parseInt( demande );
             return choix;
         }
         catch(NumberFormatException e){
             printMessage("Recommencez la saisie");
         }
     }while(true);
 }

 void printMessage(String message){
```

```
        JOptionPane.showMessageDialog(null,
              message,
              "Système de réservation de la compagnie des vols parisiens",
              JOptionPane.INFORMATION_MESSAGE);
    }

    boolean demander(String question){
        int reponse = JOptionPane.showConfirmDialog(null,
              question,
              "Système de réservation de la compagnie des vols parisiens",
              JOptionPane.YES_NO_OPTION);
        return (reponse==0)?true:false;
    }
}
```