D.A./S.M. **Avril 2009**

Programmation Orientée Objets et Evénementielle
DUT SRC IUT Meaux, année 2009-Module M3.23

TD 3 & 4

Java : initiation à la notion de classe, héritage, polymorphisme

Enoncés

Exercice 1 : Analyse d'un programme Java. Objectifs : hiérarchie de classe, héritage

Considérons le programme Java suivant :

```
package exo1;

class Livre {
  //
  //  les attributs
  //
  protected string titre, auteur, proprietaire ;
  protected int nb_page ;
  double prix ;
  //
  //  les méthodes
  //
  public Livre(String t, String a, double p, int nb){
    titre = t ; auteur = a ;
    prix = p ; proprietaire = "" ;
    nb_page = nb ;
  }
  public void Afficher() {
    System.out.println("Titre : " + titre) ;
    System.out.println("Auteur : " + auteur) ;
    System.out.println("Prix : " + prix) ;
    System.out.println("Nombre de pages : " + nb_page);
    if ( this.Est_neuf() ) {
      System.out.println("Aucune proprietaire" ) ;
    } else {
      System.out.println("Proprietaire: "+proprietaire);
    }
    System.out.println() ;
  }
  public boolean Est_neuf() {
       if ( proprietaire =="" ) return true ;
       else return false ;
  }
  public void Acheter(String nom) {
       proprietaire = nom ;
  }
}

class BD extend Livre {
  private boolean encouleur ;
  public BD(String t,String a,double p,int nb, boolean c){
    super(t,a,p,nb) ;
    encouleur = c ;
  }
}
```

```
class Album extends Livre {
  boolean page_coloriee[];
  public Album(String t, String a, double p, int n){
    super(t,a,p,n) ;
    page_coloriee = new boolean[n];
    int i ;
    for (i=0 ; i<100 ; i++)
      page_coloriee[i] = false ;
  }
  public void Colorie(int num_page){
    if((page_coloriee[num_page] == false) && !Est_neuf()){
      page_coloriee[num_page] = true ;
    } else {
      System.out.println("page deja coloriee" ) ;
    }
  }
}

// la classe principale d'appel

public class Test {
  public static void main(String[] args) {

    Livre l1 = new Livre("Le petit prince","St
                                    Exupéry",10.40, 50) ;
    Livre l2 = new Livre("Contes","Grimm",14.40,254) ;
    l1.Afficher() ;
    l1.acheter("moi") ;
    l1.Afficher() ;
    l1.prix = 0.0 ;
    l2.Acheter("lui") ;
    l2.Afficher();

    BD b1 = new BD("Lucky Luke","Morris",10.40, 45, true);
    BD b2 = new BD("Tintin","Herge",200.40, 45, false) ;
    b1.Acheter("moi");
    b1.Afficher() ;
    b2.Afficher() ;

    Album a1 = new Album("Dora","Dora", 300, 3.5) ;
    a1.Afficher() ;
    a1.Colorie(23) ;
    a1.Acheter("moi");
    a1.Colorie(23) ;
  }
}
```

Questions :

1) Expliquer les informations que le programme va afficher lors de son exécution.
2) Dans la classe Livre, l'attribut prix a été défini sans règle d'encapsulation (public, private…) Comment Java va-t-il l'interpréter ? Comment pouvez-vous le tester ou comment est-il testé dans le programme ?
3) Expliquer comment on teste dans ce programme si un livre est neuf ou non.
4) Décrivez la hiérarchie de classe décrite dans ce programme et expliquer le processus d'appel entre les constructeurs.
5) Expliquer comme ce programme gère le fait de colorier une page d'un album à colorier.
6) Découvrez les erreurs de frappe présentes dans le code. Corriger-les en codant et en commentant ces quelques lignes sous Eclipse.

Exercice 2 : Héritage de classe et constructeurs

1) Un site internet est spécialisé dan la vente de livres pour enfant. Ces livres sont soit des bandes dessinées, soit des albums à colorier. Un livre est défini par son titre, son auteur, son prix et son nombre de pages. Les bandes dessinées sont soit en couleur soit en noir&blanc alors l'utilisateur a la possibilité de colorier une page d'un album présenté. Proposer et implémenter et tester sous Eclipse une solution à ce problème.
2) Le site web veut donner la possibilité aux utilisateurs de revendre un livre et de s'échanger deux bandes dessinées si elles ont un prix équivalent. Modifier le programme précédent pour prendre en compte ces fonctions supplémentaires.
3) Enfin, le site web veut étendre son offre d'œuvre culturelles à des films (DVD) qui sont définis eux-aussi par un titre, un auteur et un prix mais avec en plus une information sur la durée du film. Comment modifier la hiérarchie de classe pour intégrer ces modifications ? Programmer-le.

Exercice 3 : Héritage de classe et polymorphisme

Vous devez développer pour une chaîne de télévision un logiciel permettant de gérer la programmation journalière d'émissions télévisuelles. Pour simplifier le problème, toutes les heures ou les durées sont représentées par des valeurs entières (13h, 15h, 22h…). Une émission peut être de trois types : divertissement, fiction et reportage. Une émission de type divertissement dure obligatoirement 2 heures, possède un nom et est systématiquement présentée par un animateur. Une fiction se définit par le nom du film, l'année de sa réalisation, le nom du réalisateur et s'il s'agit ou non d'une rediffusion. Enfin un reportage est défini par un nom, un thème (fixé parmi les trois thèmes : information, animalier, culturel) et une durée.

1) Proposer une solution fondée sur les héritages entre classe pour représenter toutes les émissions possibles. Donner pour chaque classe, la liste de ses attributs et les paramètres de ses constructeurs. Comment coderiez-vous le fait que le thème d'un reportage soit prédéfini ?
2) Implanter cette solution sous Eclipse en Java et tester tout d'abord les classes que vous avez imaginées en instantiant différents objets de votre choix (avec les constructeurs) pour chacune de celles-ci.
3) La programmation d'une émission dans la journée dépend du type d'émission mais se traduit par le fait de lui fixer une heure de début de diffusion et de calculer l'heure de fin.
    a. Les divertissements durent systématiquement 2 heures, mais on ne peut les programmer qu'entre 18h et 23h.
    b. Les fictions qui ne sont pas des rediffusion ne se programment qu'en début de soirée, c'est-à-dire qu'à 21h, alors qu'une rediffusion peut se programmer n'importe quand dans la journée.
    c. Enfin, les reportages ne se programment qu'à des heures creuses (14h –18h et 0h-6h) et s'ils ont une durée inférieure, égale à 1heure.

    Proposer une solution fondée sur la notion de classe abstraite et de polymorphisme permettant de décrire la programmation ou non n'importe quelle émission à une heure donnée. Comment initialiser efficacement ces heures de début et de fin de diffusion.
4) Définissez enfin un programme télé comme un ensemble fini (tableau) d'émissions hétérogènes que vous remplirez d'émission de votre choix, programmer à une heure de votre choix. Décrire puis implémenter les algorithmes vous permettant :
    a. Afficher la liste des émissions programmées dans la journée
    b. Tester s'il y a une superposition de programmation.
    c. Afficher heure par heure les émissions programmées pour vérifier que tous les créneaux horaires ont bien été remplis.

TD 3 & 4

Java : initiation à la notion de classe, héritage, polymorphisme, collections

Corrigés

Exercice 1 : Analyse d'un programme Java. Objectifs : hiérarchie de classe, héritage

```java
package exo1;

class Livre {
  //
  //  les attributs
  //
  protected string titre, auteur, proprietaire ;
  protected int nb_page ;
  double prix ;
  //
  //  les méthodes
  //
  public Livre(String t, String a, double p, int nb){
    titre = t ; auteur = a ;
    prix = p ; proprietaire = "" ;
    nb_page = nb ;
  }
  public void Afficher() {
    System.out.println("Titre : " + titre) ;
    System.out.println("Auteur : " + auteur) ;
    System.out.println("Prix : " + prix) ;
    System.out.println("Nombre de pages : " + nb_page) ;
    if ( this.Est_neuf() ) {
      System.out.println("Aucune proprietaire" ) ;
    } else {
      System.out.println("Proprietaire : "+proprietaire);
    }
    System.out.println() ;
  }
  public boolean Est_neuf() {
    if ( proprietaire =="" ) return true ;
    else return false ;
  }
  public void Acheter(String nom) {
    proprietaire = nom ;
  }
}

class BD extend Livre {
  private boolean encouleur ;
  public BD(String t, String a,double p,int nb,boolean c){
    super(t,a,p,nb) ;
    encouleur = c ;
  }
}
```

```java
class Album extends Livre {
  boolean page_coloriee[];
  public Album(String t, String a, double p, int n){
    super(t,a,p,n) ;
    page_coloriee = new boolean[n];
    int i ;
    for (i=0 ; i<100 ; i++)
      page_coloriee[i] = false ;
  }
  public void Colorie(int num_page){
    if((page_coloriee[num_page] == false) && !Est_neuf()){
      page_coloriee[num_page] = true ;
    } else { System.out.println("page deja coloriee" ) ;
    }
  }
}

// la classe principale d'appel

public class Test {
  public static void main(String[] args) {
    Livre l1 = new Livre("Le petit prince","St
                                    Exupéry",10.40, 50) ;
    Livre l2 = new Livre("Contes","Grimm",14.40,254) ;
    l1.Afficher() ;
    l1.acheter("moi") ;
    l1.Afficher() ;
    l1.prix = 0.0 ;
    l2.Acheter("lui") ;
    l2.Afficher();

    BD b1 = new BD("Lucky Luke","Morris",10.40, 45, true);
    BD b2 = new BD("Tintin","Herge",200.40, 45, false) ;
    b1.Acheter("moi");
    b1.Afficher() ;
    b2.Afficher() ;

    Album a1 = new Album("Dora","Dora", 300, 3.5) ;
    a1.Afficher() ;
    a1.Colorie(23) ;
    a1.Acheter("moi");
    a1.Colorie(23) ;
  }
}
```

Questions :

1) *Expliquer les informations que le programme va afficher lors de son exécution.*

```
Titre : Le petit prince
Titre : Le petit prince
Auteur : St Exupéry
Prix : 10.4
Nombre de pages : 50
Aucun proprietaire

Titre : Le petit prince
Auteur : St Exupéry
Prix : 10.4
Nombre de pages : 50
Proprietaire : moi

Titre : Contes
Auteur : Grimm
Prix : 14.4
Nombre de pages : 254
Proprietaire : lui

Titre : Lucky Luke
Auteur : Morris
Prix : 10.4
Nombre de pages : 45
Proprietaire : moi

Titre : Tintin
Auteur : Herge
Prix : 200.4
Nombre de pages : 45
Aucun proprietaire

Titre : Dora
Auteur : Dora
Prix : 3.5
Nombre de pages : 300
Aucun proprietaire

page 23 deja coloriee
```

2) *Dans la classe Livre, l'attribut prix a été défini sans règle d'encapsulation (public, private…) Comment Java va-t-il l'interpréter ? Comment pouvez-vous le tester ou comment est-il testé dans le programme ?*
Java l'interprète comme un attribut public : on y a d'ailleurs accès dans la classe Test directement dans l'instruction l1.prix = 0.0.

3) *Expliquer comment on teste dans ce programme si un livre est neuf ou non.*
En fait, un livre est neuf s'il n'a pas de propriétaire. D'où l'intérêt du constructeur qui initialise l'attribut propriétaire à

une chaîne de caractères vide. Il faut initialiser toutes les variables ! Sinon le test risque de ne pas fonctionner systématiquement.

4) *Décrivez la hiérarchie de classe décrite dans ce programme et expliquer le processus d'appel entre les constructeurs.*

Trois classes : Livre, Bd et Album, les deux dernières héritant de propriétés de la première. La classe Livre ayant un constructeur (paramètres titre, auteur, prix et nombre de page), les sous-classes doivent avoir leur propre constructeur qui appelle (méthode super) le constructeur de la classe Livre. La classe BD possède un attribut de type booléen (en couleur) qui spécifie si la bande dessinée est ou non en couleur.

5) *Expliquer comme ce programme gère le fait de colorier une page d'un album à colorier.*

Un tableau de booléen permet de conserver l'information qu'une page a été coloriée. Ce tableau est initialisé à faux dans le constructeur de la classe Album. Chaque appel à la méthode de coloriage teste si la page a ou non été coloriée.

6) *Découvrez les erreurs de frappe présentes dans le code. Corriger-les en codant et en commentant ces quelques lignes sous Eclipse.*

Cinq erreurs figurent dans le texte. Elles sont en rouge dans le code source ci-dessus.

Voici la correction :

```
package package exo1_corrige;

class Livre {
  //
  // les attributs
  //
  protected String titre, auteur, proprietaire ;
  protected int nb_page ;
  double prix ;
  //
  // les méthodes
  //
  public Livre(String t, String a, double p, int nb){
    titre = t ; auteur = a ;
    prix = p ; proprietaire = "" ;
    nb_page = nb ;
  }
  public void Afficher() {
    System.out.println("Titre : " + titre) ;
    System.out.println("Auteur : " +auteur);
    System.out.println("Prix : " + prix) ;
    System.out.println("Nombre de pages : " + nb_page);
    if ( this.Est_neuf() ) {
      System.out.println("Aucun proprietaire" ) ;
    } else {
    System.out.println("Proprietaire:"+proprietaire);
    }
    System.out.println() ;
  }
  public boolean Est_neuf() {
    if ( proprietaire =="" ) return true ;
    else return false ;
  }
  public void Acheter(String nom) {
    proprietaire = nom ;
  }
}

class BD extends Livre {
  private boolean encouleur ;
  public BD(String t, String a, double p, int nb,
boolean c){
    super(t,a,p,nb) ;
    encouleur = c ;
  }
}

class Album extends Livre {
  boolean page_coloriee[];
  public Album(String t, String a, double p, int n){
    super(t,a,p,n) ;
    page_coloriee = new boolean[n];
    int i ;
    for (i=0 ; i<n ; i++)
      page_coloriee[i] = false ;
  }

  public void Colorie(int num_page){
    if ( (page_coloriee[num_page] == false) &&
!Est_neuf() )  {
      page_coloriee[num_page] = true ;
    } else {
      System.out.println("page"+num_page+"deja
                                        coloriee");
    }
  }
}

//la classe principale d'appel

public class Test {
  public static void main(String[] args) {
    Livre l1 = new Livre("Le petit prince","St
                                Exupéry",10.40, 50) ;
    Livre l2 = new Livre("Contes","Grimm",14.40,254) ;
    l1.Afficher() ;
    l1.Acheter("moi") ;
    l1.Afficher() ;
    l1.prix = 0.0 ;
    l2.Acheter("lui") ;
    l2.Afficher();

    BD b1 = new BD("LuckyLuke","Morris",10.40,45,true);
    BD b2 = new BD("Tintin","Herge",200.40, 45, false);
    b1.Acheter("moi");
    b1.Afficher() ;
    b2.Afficher() ;

    Album a1 = new Album("Dora","Dora", 3.5, 300) ;
    a1.Afficher() ;
    a1.Colorie(23) ;
    a1.Acheter("moi");
    a1.Colorie(23) ;
  }
}
```

Exercice 2 : Exemple de classe, de constructeur et de polymorphisme

1) *Un site internet est spécialisé dan la vente de livres pour enfant. Ces livres sont soit des bandes dessinées, soit des albums à colorier. Un livre est défini par son titre, son auteur, son prix et son nombre de pages. Les bandes dessinées sont soit en couleur soit en noir&blanc alors l'utilisateur a la possibilité de colorier une page d'un album présenté. Proposer et implémenter et tester sous Eclipse une solution à ce problème.*

C'est volontairement l'énoncé de l'exercice 1.

2) *Le site web veut donner la possibilité aux utilisateurs de revendre un livre et de s'échanger deux bandes dessinées si elles ont un prix équivalent. Modifier le programme précédent pour prendre en compte ces fonctions supplémentaires.*

La méthode suivante est ajoutée dans la classe Livre

```
public void Revendre(String acheteur, double prix_occas) {
  proprietaire = acheteur ;
  prix = prix_occas ;
}
```

Nous avons fait le choix de revendre un livre à un nouveau propriétaire et à un nouveau prix d'occasion (les deux paramètres de la méthode) mais on aurait pu choisir d'autres paramètres comme rien n'est clairement spécifié.

La méthode suivante est à ajouter dans la classe BD

```
public void Echanger(BD b) {
  if (this.prix==b.prix){
    System.out.println("Prix égaux : échange possible" ) ;
    String ProTmp = this.proprietaire;
    this.proprietaire = b.proprietaire ;
    b.Acheter(ProTmp) ;
  } else // ne pas échanger les BD
    System.out.println("Prix différents : échange impossible" ) ;
}
```

C'est la première fois que l'on voit une des contraintes des langages objet. La variable b étant d'un autre type

3) *Enfin, le site web veut étendre son offre d'œuvre culturelles à des films (DVD) qui sont définis eux-aussi par un titre, un auteur et un prix mais avec en plus une information sur la durée du film. Comment modifier la hiérarchie de classe pour intégrer ces modifications ? Programmer-le.*

La hiérarchie des classes doit être profondément modifiée. Livre et film doivent héritée d'une classe « supérieure » Œuvre. Dans ce cas, les attributs auteur, titre, proprietaire et prix « remontent dans la classe Œuvre. La classe Livre ne conserve plus que l'attribut nb_pages au même titre que la classe film une durée. Les constructeurs de ces deux classes doivent être modifiés. Les deux sous-classes BE et Album de la classe Livre ne changent pas.

```
package tp3et4_Ex2;
//la classe Oeuvre
class Oeuvre {
  // les attributs
  protected String titre, auteur, proprietaire ;
  protected double prix ;
  //le constructeur avec comme paramètre le titre,
l'auteur et le prix
  //l'attribut proprietaire est initialisé
  public Oeuvre(String t, String a, double p){
    titre = t ; auteur = a ; prix = p ; proprietaire = "";
  }
  //une oeuvre est neuve si elle n'a pas encore de
propriétaire
  //d'où l'intérêt de l'initialisation dans le
constructeur
  public boolean Est_neuf() {
    if ( proprietaire =="" ) return true ;
    else return false ;
  }
  //affiche les informations sur l'oeuvre
  public void Afficher() {
    System.out.println("Titre : " + titre) ;
    System.out.println("Auteur : " + auteur) ;
    System.out.println("Prix : " + prix) ;
    if (this.Est_neuf())
       System.out.println("Aucun proprietaire" );
    else
       System.out.println("Proprietaire : "+proprietaire);
    System.out.println() ;
  }
  //la méthode acheter donne un nom de propriét. à l'objet
  public void Acheter(String nom) {proprietaire = nom ;}
}

//La classe Film hérite de la classe Oeuvre
class Film extends Oeuvre {
  //un seul attribut supplémentaire:la durée du film en mn
  protected int durée ;
  //le constructeur de cette classe appelle celui de la
  //classe supérieure c'est-à-dire de la classe Oeuvre
  public Film(String t, String a, double p, int d){
    super(t,a,p) ;
    durée = d ;
  }
  //la méthode revente modifie simplement le propriétaire
et le prix du livre
  public void Revendre(String acheteur,double prix_occas){
    proprietaire = acheteur ;
    prix = prix_occas ;
  }
  public String Proprietaire() {
    return (proprietaire) ;
  }
}

// La classe Livre hérite de la classe Oeuvre
class Livre extends Oeuvre {
  //un seul attribut supplémentaire: nombre pages du livre
  protected int nb_pages ;
  //le constructeur de cette classe appelle celui de la
  // classe supérieure c'est-à-dire de la classe Oeuvre
```

```
  public Livre(String t, String a, double p, int nb){
    super(t,a,p) ;
    nb_pages = nb ;
  }
    //la méthode revente modifie simplement le
    // propriétaire et le prix du livre
  public void Revendre(String acheteur,double prix_occas){
    proprietaire = acheteur ;
    prix = prix_occas ;
  }
  public String Proprietaire() {
    return (proprietaire) ;
  }
}
//La classe BD hérite de la classe Livre
class BD extends Livre {
  //un seul attribut supplémentaire : un booléen qui
  //définit si la BD est en couleur ou non
  private boolean encouleur ;
  //Le constructeur qui appelle celui de la classe Livre
  //et donc celui de la classe Oeuvre
  public BD(String t,String a,double p,int n,boolean c){
    super(t,a,p,n) ;
    encouleur = c ;
  }
  //La méthode qui échange les noms des propriétaires si
  //le prix des BD est identique
  public void Echanger(BD b) {
    if (this.prix==b.prix){
      System.out.println("Prix égaux:échange possible" );
      String ProTmp = this.proprietaire;
      this.proprietaire = b.proprietaire ;
      b.Acheter(ProTmp) ;
    } else // ne pas échanger les BD
      System.out.println("Prix différents:échange
                                        impossible");
  }
}

//La classe Album qui hérite de Livre
class Album extends Livre {
  //un seul attribut supplémentaire : un tableau de
  //booléen de la taille du nombre de pages
  //définit si la page a été lue ou non
  boolean page_coloriee[] ;
  //le constructeur qui appelle celui de Livre
  //Notons que la taille du tableau de booléen est
  //initialisée dans ce constructeur en fonction des
  //paramètres passés au constructeur
  public Album(String t, String a, double p, int nb){
    super(t,a,p, nb) ;
    page_coloriee = new boolean[nb] ;
    int i ;
    for (i=0 ; i<nb ; i++)
      page_coloriee[i] = false ;
  }
  //la méthode qui colorie une page : mettre à vrai la
  //valeur du tabelau page_coloriée si ce n'a pas été fait
  public void Colorie(int num_page){
    if ((page_coloriee[num_page] == false) && !Est_neuf())
      page_coloriee[num_page] = true ;
    else {
      System.out.println("page"+num_page+"deja coloriee");
      System.out.println();
    }
  }
}

//la classe principale d'appel
public class Test {
  public static void main(String[] args) {
    //création de deux variables de type Livre
    Livre l1 = new Livre("Le petit prince","St
Exupéry",10.40, 30) ;
    Livre l2 = new Livre("Contes","Grimm",14.40, 250) ;
    //on affiche les caractéristiques de chacun
    l1.Afficher() ;
    l1.Acheter("moi") ;
    l1.Afficher() ;
    l1.prix = 0.0 ;
    l2.Acheter("lui") ;
    l2.Afficher();
    //création de deux variables de type BD
    BD b1 = new BD("Lucky Luke","Morris",10.40, 45, true);
    BD b2 = new BD("Tintin","Herge",200.40, 50, false) ;
    // "moi" achete deux BD b1 et b2
    b1.Acheter("moi");
    b1.Afficher() ;
    b2.Acheter("moi");
    b2.Afficher() ;
    // "moi" revend à "lui" b2
    b2.Revendre("lui",10.4);
    b2.Afficher() ;
    //on teste la fonction d'échange de b1 et b2
    b1.Echanger(b2) ;
    System.out.println("APRES ECHANGE DE "
                          +b1.titre+" et "+b2.titre) ;
    b1.Afficher();
    b2.Afficher() ;
    //on teste si la méthode de coloriage d'une page
    //fonctionne
    Album a1 = new Album("Dora","Dora",3.5, 200) ;
    a1.Afficher() ;
    a1.Colorie(23) ;
    a1.Acheter("moi");
    a1.Colorie(23) ;
    // On crée un objet film
    Film f = new Film("Taxi","Besson",20,90);
    f.Afficher();
  }
}
```

Editions :

```
Titre : Le petit prince
Auteur : St Exupéry
Prix : 10.4
Aucun proprietaire

Titre : Le petit prince
Auteur : St Exupéry
Prix : 10.4
Proprietaire : moi

Titre : Contes
Auteur : Grimm
Prix : 14.4
Proprietaire : lui

Titre : Lucky Luke
Auteur : Morris
Prix : 10.4
Proprietaire : moi

Titre : Tintin
Auteur : Herge
Prix : 200.4
Proprietaire : moi

Titre : Tintin
Auteur : Herge
Prix : 10.4
Proprietaire : lui

Prix égaux : échange possible
APRES ECHANGE DE Lucky Luke et Tintin
Titre : Lucky Luke
Auteur : Morris
Prix : 10.4
Proprietaire : lui

Titre : Tintin
Auteur : Herge
Prix : 10.4
Proprietaire : moi

Titre : Dora
Auteur : Dora
Prix : 3.5
Aucun proprietaire

page 23 deja coloriee

Titre : Taxi
Auteur : Besson
Prix : 20.0
Aucun proprietaire
```

Exercice 3 : Héritage de classe et polymorphisme

Vous devez développer pour une chaîne de télévision un logiciel permettant de gérer la programmation journalière d'émissions télévisuelles. Pour simplifier le problème, toutes les heures ou les durées des émissions sont représentées par des valeurs entières (13h, 15h, 22h…). Une émission peut être de trois types : divertissement, fiction et reportage. Une émission de type divertissement dure obligatoirement 2 heures, possède un nom et est systématiquement présentée par un animateur. Une fiction se définit par le nom du film, l'année de sa réalisation, le nom du réalisateur et s'il s'agit ou non d'une rediffusion. Enfin un reportage est défini par un nom, un thème (fixé parmi les trois thèmes : information, animalier, culturel) et une durée.

1) *Proposer une solution fondée sur les héritages entre classe pour représenter toutes les émissions possibles. Donner pour chaque classe, la liste de ses attributs et les paramètres de ses constructeurs. Comment coderiez-vous le fait que le thème d'un reportage soit prédéfini ?*

Une classe Emission et trois sous-classes Divertissement, Fiction, Reportage avec les attributs et les constructeurs suivants.

```
//////////////////////////
abstract class Emission {
  protected String nom ;
  protected int heure_debut, heure_fin ;
}
//////////////////////////
class Divertissement extends Emission {
  private String animateur ;
  private static final int duree = 2;
  public Divertissement(String n, String anim) {
    nom = n ; animateur = anim ;
  }
}
///////////////////////////////////
class Fiction extends Emission {
    private String realisateur ;
    private boolean rediffusion ;
    private int duree, annee ;
    public Fiction (String n,String real,
                    boolean redif, int d ){
      nom = n ; realisateur = real ;
      rediffusion = redif ; duree = d ;
    }
}
///////////////////////////////////
class Reportage extends Emission {
  private String theme ;
  private int duree ;
  public Reportage (String n, String t, int d){
      nom = n ; theme = t ; duree = d ;
  }
}
```

Le fait de choisir un thème parmi les trois donnés peut se tester de différentes manières. Une solution consiste à créer un tableau static final contenant les trois thèmes et de conserver pour une émission l'adresse (1 2 ou 3) dans le tableau. Le constructeur a alors uniquement ce numéro comme paramètre.

```
class Reportage extends Emission {
  private static final String theme[] ={"Information","Animalier","Culturel"};
  private int duree, type_theme ;
  public Reportage (String n, int t, int d){
      nom = n ;  type_theme = t ;  duree = d ;
  }
}
```

2) *Implanter cette solution sous Eclipse en Java et tester tout d'abord les classes que vous avez imaginées en instantiant différents objets de votre choix (avec les constructeurs) pour chacune de celles-ci.*

En utilisant les constructeurs que l'on a définit précédemment, on a alors des déclarations du type :

```
Divertissement em1 = new Divertissement("La roue de la fortune","Foucault") ;
Fiction em2 = new Fiction("Citizen Kane", "Wells",true,3) ;
Reportage em3 = new Reportage("Voiture de luxe","Information", 2) ;
```

Pour réaliser des éditions « automatiques », utiliser la génération de méthodes déléguées avec le clic droit de la souris.

Solution :

```
package tp3et4_Ex3_Question2;

abstract class Emission {
  protected String nom ;
  protected int heure_debut, heure_fin ;
  public String toString() {
    // générée automatiquement : click droit de la souris puis « Source » et « Generate Delegate methods »
    return nom.toString();
  }
}
class Divertissement extends Emission {
  private String animateur ;
  private static final int duree = 2;
  public Divertissement(String n, String anim) {
    nom = n ; animateur = anim ;
  }
  public String toString() { // générée automatiquement
    return super.toString()+" : "+animateur.toString();
```

```
  }
}
class Fiction extends Emission {
  private String realisateur ;
  private boolean rediffusion ;
  private int duree, annee ;
  public Fiction (String n,String real,
       boolean redif, int d ){
    nom = n ; realisateur = real ;
    rediffusion = redif ; duree = d ;
  }
  public String toString() {
    return super.toString()+" : "+realisateur.toString();
  }
}
class Reportage extends Emission {
  private String theme ;
  private int duree ;
  public Reportage (String n, String t, int d){
    nom = n ; theme = t ; duree = d ;
  }
  public String toString() {
    return super.toString()+" : "+theme.toString();
  }
}
public class Test_question2 {
  public static void main(String[] args) {
    Divertissement em1 = new Divertissement("La roue de la fortune","Foucault") ;
    System.out.println(em1.toString());
    Fiction em2 = new Fiction("Citizen Kane", "Wells",true,3) ;
    System.out.println(em2.toString());
    Reportage em3 = new Reportage("Voiture de luxe","Information", 2) ;
    System.out.println(em3.toString());
  }
}
```

Editions :

```
La roue de la fortune : Foucault
Citizen Kane : Wells
Voiture de luxe : Information
```

3) *La programmation d'une émission dans la journée dépend du type d'émission mais se traduit par le fait de lui fixer une heure de début de diffusion et de calculer l'heure de fin.*
   - a- *Les divertissements durent systématiquement 2 heures, mais on ne peut les programmer qu'entre 18h et 23h.*
   - b- *Les fictions qui ne sont pas des rediffusions ne se programment qu'en début de soirée, c'est-à-dire à 21h, alors qu'une rediffusion peut se programmer n'importe quand dans la journée.*
   - c- *Enfin, les reportages ne se programment qu'à des heures creuses (14h –18h et 0h-6h) et s'ils ont une durée inférieure, égale à 1heure.*

Proposer une solution fondée sur la notion de classe abstraite et de polymorphisme permettant de décrire la programmation ou non de n'importe quelle émission à une heure donnée. Comment initialiser efficacement ces heures de début et de fin de diffusion.

Nous allons déclarer une méthode abstraite :

Programmer(heure de début) ;

dans la classe Emission et chacune des sous-classes définira sa propre méthode Programmer qui prendra en compte les spécificités de chacune des sous-classes. Une bonne solution consiste en une méthode Programmer qui retourne un booléen si la programmation est possible ou non. De ce fait, il est nécessaire d'initialiser les variables de début et de fin de diffusion. On peut par exemple les fixer à -1. Cette initialisation peut se faire soit dans les constructeurs des sous-classes, soit et c'est encore plus propre, dans le constructeur de la classe Emission. Ce sera alors réalisé par appel récursif de tous les constructeurs en cascade, lors de la création d'une émission quelconque.

Remarque : les méthodes d'affichage sont à réécrire car on attend plus et il faut les détailler soi-même.

On affichera les émissions programmées.

Solution

```
package tp3et4_Ex3_Question3;

abstract class Emission {
   protected String nom ;
   protected int heure_debut, heure_fin ;
   public abstract boolean Programmer(int heure) ;
   public Emission(){
      heure_debut = heure_fin = -1 ;
   }
   public boolean Programmee() {
      if ( heure_debut != -1 )
         return true;
      else
         return false ;
   }
   public void Affiche() {
      System.out.println("===============") ;
      System.out.println("Emission :" + nom) ;
      if ( heure_debut != -1 )
         System.out.println("Emission programmée à " +
                                          heure_debut);
   }
}

class Divertissement extends Emission {

   private String animateur ;
   private static final int duree = 2  ;
   public Divertissement(String n, String anim) {
      nom = n ; animateur = anim ;
   }
   public void Affiche(){
      super.Affiche() ;
      System.out.println("Animateur :" + animateur) ;
      System.out.println("Durée :" + duree) ;
   }
   public boolean Programmer(int heure) {
      if ( heure >= 18 && heure <= 21) {
         heure_debut = heure ;
         heure_fin = heure + duree ;
         return true ;
      } else {
         return false ;
      }
   }
}

class Fiction extends Emission {
   private String realisateur ;
   private boolean rediffusion ;
   private int duree, annee ;
   public Fiction (String n,String real,boolean redif,int d ){
      nom = n ; realisateur = real ; rediffusion = redif ;
      duree = d ;
   }
   public void Affiche(){
      super.Affiche() ;
      System.out.println("Réalisateur :" + realisateur) ;
      System.out.println("Duree :" + duree) ;
      System.out.println("Année de réalisation :" + annee) ;
      if ( rediffusion )
         System.out.println("C'est une rediffusion") ;
      else
         System.out.println("Première Diffusion !") ;
   }
   public boolean Programmer(int heure) {
      if ( rediffusion || heure == 21 ) {
         heure_debut = heure ;
         heure_fin = heure + duree ;
         return true ;
      } else
         return false ;
   }
}

class Reportage extends Emission {
   private static final String theme[] =
{"Information","Animalier","Culturel"};
   private int duree, type_theme ;
   public Reportage (String n, int t, int d){
      nom = n ;  type_theme = t ;  duree = d ;
   }
   public void Affiche(){
      super.Affiche() ;
      switch ( type_theme ) {
      case 1 :
         System.out.println("Theme :" + theme[1]) ;
         break ;
      case 2 :
         System.out.println("Theme :" + theme[2]) ;
         break ;
      case 3 :
         System.out.println("Theme :" + theme[3]) ;
         break ;
      }
      System.out.println("Duree :" + duree) ;
   }
   public boolean Programmer(int heure) {
      if ( duree ==1 && (( heure >=14 && heure <=18)
            || (heure >=0 && heure <=6)) ) {
         heure_debut = heure ;
         heure_fin = heure + duree ;
         return true ;
      } else {
         return false ;
      }
   }

}

public class Test_question3 {
   public static void main(String[] args) {
      int i ;
      System.out.println("/////////////// ") ;
      System.out.println("Question 2 : création et test de
                                          variables ") ;
      System.out.println("/////////////// ") ;

      Divertissement em1 = new Divertissement("La roue de la
                                       fortune","Foucault") ;
      if ( em1.Programmer(20) )
         System.out.println("ok emission programmée");
      else System.out.println("emission non programmée") ;

      Fiction em2 = new Fiction("Citizen Kane",
                                       "Wells",true,3) ;
      if ( em2.Programmer(17) )
         System.out.println("ok emission programmée");
      else System.out.println("emission non programmée") ;

      Reportage em3 = new Reportage("Voiture de luxe", 1, 1) ;
      if ( em3.Programmer(17) )
         System.out.println("ok emission programmée");
      else System.out.println("emission non programmée") ;

      System.out.println("/////////////// ") ;
      System.out.println("   Question 3 ") ;
      System.out.println("/////////////// ") ;
      if ( em1.Programmee() ) em1.Affiche() ;
      if ( em2.Programmee() ) em1.Affiche() ;
      if ( em3.Programmee() ) em1.Affiche() ;
   }
}
```

```
Editions :
///////////////
   Question 2 : création et test de variables
///////////////
ok emission programmée
ok emission programmée
ok emission programmée
///////////////
   Question 3
///////////////
===============
Emission :La roue de la fortune
 Emission programmée à 20
Animateur :Foucault
Durée :2
===============
Emission :La roue de la fortune
 Emission programmée à 20
Animateur :Foucault
Durée :2
===============
Emission :La roue de la fortune
 Emission programmée à 20
Animateur :Foucault
Durée :2
```

4) *Définissez enfin un programme télé comme un ensemble fini (tableau) d'émissions hétérogènes que vous remplirez d'émissions de votre choix, programmées à une heure de votre choix. Décrire puis implémenter les algorithmes vous permettant :*

a- Afficher la liste des émissions programmées dans la journée

C'est un simple parcours du tableau des émissions et un test si elles sont programmées. On est obligé de passer par la méthode Programmee qui teste si l'attribut heure_debut est égal à -1 ou non. On a ajouté une méthode Affiche() dans chacune des classes pour afficher les attributs respectifs. Cette méthode appelle via la méthode super la méthode Affiche() de la classe Emission.

```
int nb_max = 24 ;
Emission programme[] = new Emission[nb_max] ; // Attention polymorphisme
programme[0] = em1 ;
programme[1] = em2 ;
programme[2] = em3 ;
int nbEmissions = 3; // nombre d émissions créées
for (i=0; i<nbEmissions; i++){
  if ( programme[i].Programmee() )
    programme[i].Affiche() ;
}
```

Attention polymorphisme : Chaque type d'émission (divertissement, fiction, reportage) est transformée en type Emission. Pour pouvoir appeler la méthode Affiche(), il faut que cette méthode existe (éventuellement abstraite dans la super classe Emission). Sinon on ne peut pas l'appeler.
On peut alors redéfinir et en fait étendre Affiche() par l'appel de super.Affiche() dans la méthode Affiche() de la sous-classe concernée.

Exemple de la méthode affiche pour la classe Divertissement avec l'appel du super.Affiche() ;

```
public void Affiche(){
  super.Affiche() ;
  System.out.println("Animateur :" + animateur) ;
  System.out.println("Durée :" + duree) ;
}
```

b- Tester si deux émissions sont programmées à la même heure.

On évite de tester toutes les paires d'émissions. Pour cela on stocke dans un tableau si une plage horaire est utilisée ou non (attention, certaines émissions durent 2H ! ou 3H comme de longues fictions et peuvent se superposer durant 1H avec un reportage !). On définit un tableau de 24 booléens et on parcourt toutes les émissions pour mettre à jour ce tableau et afficher si on trouve une superposition quelque part.

```
boolean[] plage = new boolean [24] ;
/// initialisation du tableau à faux
for (i=0; i<24; i++) plage[i] = false ;
/// parcours de toutes les émissions
for (i=0; i<nbEmissions; i++)
  if ( programme[i].Programmee() ) {
    System.out.println("plage : " + programme[i].Debut()) ;
    for (int j = programme[i].Debut(); j<programme[i].Fin(); j++)
      if ( plage[j] == false ) plage[j] = true ;
      else System.out.println("attention : superposition sur la plage " + j);
}
```

c- Afficher heure par heure les émissions programmées pour vérifier que tous les créneaux horaires ont bien été remplis … sans pour autant corriger dans le détail les superpositions. On suppose qu il n y a pas de superposition

Il suffit en plus de la question précédente d'associer non seulement un booléen mais aussi le numéro de l'émission. On doit alors créer une nouvelle classe pour « regrouper » ces deux informations. Mais le

plus simple est de créer non pas un tableau de booléens mais un tableau de numéros d'émission (initialisé à -1 par exemple).

```
int[] plage2 = new int [24] ;
/// initialisation du tableau à -1
for (i=0; i<24; i++) plage2[i] = -1 ;
/// parcours de toutes les émissions
for (i=0; i<nbEmissions; i++) {
  if ( programme[i].Programmee() ) {
    System.out.println("plage : " + programme[i].Debut()) ;
    for (int j = programme[i].Debut(); j<programme[i].Fin(); j++)
      plage2[j] = i ;
  }
}
// l'affichage des 24 plages
System.out.println("PLANNING GENERAL") ;
for (i=0; i<24; i++) {
  System.out.println("Plage  :" + i) ;
  if ( plage2[i] != -1 )
    programme[plage2[i]].Affiche() ;
  else
    System.out.println("Aucune émission") ;
}
```

La solution complète :

```
abstract class Emission {
   protected String nom ;
   protected int heure_debut, heure_fin ;
   public abstract boolean Programmer(int heure) ;
   public Emission(){
      heure_debut = heure_fin = -1 ;
   }
   public boolean Programmee() {
      if ( heure_debut != -1 )
         return true;
      else
         return false ;
   }
   public int Debut() { return heure_debut ; }
   public int Fin() { return heure_fin ; }
   public void Affiche() {
      System.out.println("===============") ;
      System.out.println("Emission :" + nom) ;
      if ( heure_debut != -1 )
         System.out.println(" Emission programmée à "+ heure_debut);
   }
}

class Divertissement extends Emission {
   private String animateur ;
   private static final int duree = 2  ;
   public Divertissement(String n, String anim) {
      nom = n ; animateur = anim ;
      // heure_debut = heure_fin = -1 ;
   }
   public void Affiche(){
      super.Affiche() ;
      System.out.println("Animateur :" + animateur) ;
      System.out.println("Durée :" + duree) ;
   }
   public boolean Programmer(int heure) {
      if ( heure >= 18 && heure <= 21) {
         heure_debut = heure ;
         heure_fin = heure + duree ;
         return true ;
      } else {
         return false ;
      }
   }
}

class Fiction extends Emission {
    private String realisateur ;
    private boolean rediffusion ;
    private int duree, annee ;
    public Fiction ( String n, String real, boolean redif, int d ){
      nom = n ; realisateur = real ; rediffusion = redif ;
      duree = d ;
      // heure_debut = heure_fin = -1 ;
    }
   public void Affiche(){
      super.Affiche() ;
      System.out.println("Réalisateur :" + realisateur) ;
      System.out.println("Duree :" + duree) ;
      System.out.println("Année de réalisation :" + annee) ;
      if ( rediffusion )
         System.out.println("C'est une rediffusion") ;
      else
         System.out.println("Première Diffusion !") ;
   }
    public boolean Programmer(int heure) {
      if ( rediffusion || heure == 21 ) {
        heure_debut = heure ;
        heure_fin = heure + duree ;
        return true ;
      } else
        return false ;
   }
}

class Reportage extends Emission {
private static final String theme[]
                              ={"Information","Animalier","Culturel"};
   private int duree, type_theme ;
   public Reportage (String n, int t, int d){
       nom = n ;  type_theme = t ;  duree = d ;
   }
   public void Affiche(){
      super.Affiche() ;
      switch ( type_theme ) {
      case 1 :
         System.out.println("Theme :" + theme[1]) ;
         break ;
      case 2 :
         System.out.println("Theme :" + theme[2]) ;
         break ;
      case 3 :
         System.out.println("Theme :" + theme[3]) ;
         break ;
      }
      System.out.println("Duree :" + duree) ;
   }
   public boolean Programmer(int heure) {
      if(duree ==1&&((heure>=14 && heure<=18)||(heure>=0 && heure<=6)) ) {
         heure_debut = heure ;
         heure_fin = heure + duree ;
         return true ;
      } else {
         return false ;
      }
   }
}

public class Test {
   public static void main(String[] args) {
      int i ;
      System.out.println("/////////////// ") ;
      System.out.println("   Question 2 : création et test de
                                                    variables ") ;
      System.out.println("/////////////// ") ;

      Divertissement em1 = new Divertissement("La roue de la
                                          fortune","Foucault") ;
      if ( em1.Programmer(20) )
         System.out.println("ok emission programmée");
      else
         System.out.println("emission non programmée") ;

      Fiction em2 = new Fiction("Citizen Kane", "Wells",true,3) ;
      if ( em2.Programmer(17) )
         System.out.println("ok emission programmée");
      else
         System.out.println("emission non programmée") ;
```

```
    Reportage em3 = new Reportage("Voiture de luxe", 1, 1) ;
    if ( em3.Programmer(5) )
       System.out.println("ok emission programmée");
    else
       System.out.println("emission non programmée") ;

    Reportage em4 = new Reportage("Bricolage", 1, 1) ;
    if ( em4.Programmer(18) )
       System.out.println("ok emission programmée");
    else
       System.out.println("emission non programmée") ;

    int nb_max = 24 ; // nbre maximum d émissions dans une journée
    Emission programme[] = new Emission[nb_max] ;

    programme[0] = em1 ;
    programme[1] = em2 ;
    programme[2] = em3 ;
    programme[3] = em4 ;
    int nbEmissions = 4;

    System.out.println("///////////////// ") ;
    System.out.println("   Question 4 a ") ;
    System.out.println("///////////////// ") ;
    for (i=0; i<nbEmissions; i++){
       if ( programme[i].Programmee() )
          programme[i].Affiche() ;
    }

    System.out.println("///////////////// ") ;
    System.out.println("   Question 4 b ") ;
    System.out.println("///////////////// ") ;

    //on évite de traiter toutes les paires d'émission en cochant
    //et donc stockant dans un tableau les plages utilisées

    boolean[] plage = new boolean [24] ;
    /// initialisation du tableau à faux
    for (i=0; i<24; i++) plage[i] = false ;
    /// parcours de toutes les émissions
    for (i=0; i<nbEmissions; i++)
       if ( programme[i].Programmee() ) {
          System.out.println("plage : " + programme[i].Debut()) ;
          for (int j=programme[i].Debut();j<programme[i].Fin();j++)
            if ( plage[j] == false ) plage[j] = true ;
            else System.out.println("attention :
                                superposition sur la plage " + j);
    }

    System.out.println("///////////////// ") ;
    System.out.println("   Question 4 c ") ;
    System.out.println("///////////////// ") ;
    // On suppose aucune superposition
    int[] plage2 = new int [24] ;
    /// initialisation du tableau à -1
    for (i=0; i<24; i++) plage2[i] = -1 ;
    /// parcours de toutes les émissions
    for (i=0; i<nbEmissions; i++) {
       if ( programme[i].Programmee() ) {
          System.out.println("plage : " + programme[i].Debut()) ;
          for (int j=programme[i].Debut();j<programme[i].Fin();j++)
            plage2[j] = i ;
       }
    }
    // l'affichage des 24 plages
    System.out.println("PLANNING GENERAL") ;

    for (i=0; i<24; i++) {
       System.out.println("Plage  :" + i) ;
       if ( plage2[i] != -1 )
          programme[plage2[i]].Affiche() ;
       else
          System.out.println("Aucune émission") ;
    }
  }
}
```

Editions :

```
///////////////
   Question 2 : création et test de variables
///////////////
ok emission programmée
ok emission programmée
ok emission programmée
ok emission programmée
///////////////
   Question 4 a
///////////////
==============
Emission :La roue de la fortune
 Emission programmée à 20
Animateur :Foucault
Durée :2
==============
Emission :Citizen Kane
 Emission programmée à 17
Réalisateur :Wells
Duree :3
Année de réalisation :0
C'est une rediffusion
==============
Emission :Voiture de luxe
 Emission programmée à 5
Theme :Animalier
Duree :1
==============
Emission :Bricolage
 Emission programmée à 18
Theme :Animalier
Duree :1
///////////////
   Question 4 b
///////////////
plage : 20
plage : 17
plage : 5
plage : 18
attention : superposition sur la plage 18
///////////////
   Question 4 c
///////////////
On suppose aucune superposition
plage : 20
plage : 17
plage : 5
plage : 18
PLANNING GENERAL
Plage  :0
Aucune émission
Plage  :1
Aucune émission
Plage  :2
Aucune émission
Plage  :3
Aucune émission
Plage  :4
Aucune émission
Plage  :5
==============
Emission :Voiture de luxe
 Emission programmée à 5
Theme :Animalier
Duree :1
Plage  :6
Aucune émission
Plage  :7
Aucune émission
Plage  :8
Aucune émission
Plage  :9
Aucune émission
Plage  :10
Aucune émission
Plage  :11
Aucune émission
Plage  :12
Aucune émission
Plage  :13
Aucune émission
Plage  :14
Aucune émission
Plage  :15
Aucune émission
Plage  :16
Aucune émission
Plage  :17
==============
Emission :Citizen Kane
 Emission programmée à 17
Réalisateur :Wells
Duree :3
Année de réalisation :0
C'est une rediffusion
Plage  :18
==============
Emission :Bricolage
 Emission programmée à 18
Theme :Animalier
Duree :1
Plage  :19
==============
Emission :Citizen Kane
 Emission programmée à 17
Réalisateur :Wells
Duree :3
Année de réalisation :0
C'est une rediffusion
Plage  :20
==============
Emission :La roue de la fortune
 Emission programmée à 20
Animateur :Foucault
Durée :2
Plage  :21
==============
Emission :La roue de la fortune
 Emission programmée à 20
Animateur :Foucault
Durée :2
Plage  :22
Aucune émission
Plage  :23
Aucune émission
```

Attention superposition