TD : Compilation d'exercices et d'extraits d'examens

1. Classes et visibilité

a. Parmi les appels suivant indiquez ceux qui sont corrects et ceux qui ne le sont pas

```
public class A {
      public int i ;
      private int a;
      protected int b;
      public void afficher() { System.out.println(i+a+b); }
}
public  class B extends A {
      public void afficher() { System.out.println(i+a+b); }
}
public class D {
      Public A a = new A();
      Public void afficher() { System.out.println(a.i + a.a + a.b); }
}
public class E {
      Public B b = new B();
      Public void afficher() { b.afficher();}
}
public class F {
      Public B b = new B();
      Public void afficher() { System.out.println(b.i + b.a + b.b) }
}
```

2. Relations entre classes

1. Définir les relations entre les classes suivantes
   a. Personne, homme, femme, enfant
   b. Animal, chien, chat, ravet, dalmatien, domestique, sauvage, racoon, ours, comestible
2. Généraliser
   a. Bateau, voiture, avion
   b. Souris, clavier, écran
3. Spécialiser
   a. Ordinateur, …

D'après le code suivant, donnez le diagramme de classe (classes et relations).

Public class A {public B b ;}

Public class B { public C c; public B(C c) {this.c = new C(c);}}

Public class C { public int I, j; public C(C c) { this.i = C.i; this.j = C.j ;} }

Public class D { void test() {B b =new B(); A a = new A(b); A a2 = new A(b); }

3. Analyse de code 2009 (4 points)

```
public class Parent  {
  int x;
  Parent(int k) {x=k;}
  int ajoute(int a) {     return x+a;  }
  public void moi() {  System.out.println(" x = "+ x); }
}
public class Enfant1 extends Parent {
   int y;
   Enfant1 (int k, int l) { super(k); y=l; }
   int ajoute(int a) { return x+2*a;}
}
public class Enfant2 extends Enfant1 {
   int z ;
   Enfant2 (int k, int l, int m) { super(k, l); z= m; }
   int ajoute(int a) { return x+3*a;}
   public void moi() {
      super.moi();
      System.out.println(" z = "+ z);
   }
}

public class Essai{
   public static void main (String args[]) {
     int a =2;
     Parent p = new Parent(3);
     p.moi();
     System.out.println(" ajoute("+ a +") = "+ p.ajoute(a) );
     Enfant1 e1 = new Enfant1(3, 4);
     e1.moi();
     System.out.println(" ajoute("+ a +") = "+ e1.ajoute(a) );
     e1 = new Enfant2(3, 4, 5);
     e1.moi();
     System.out.println(" ajoute("+ a +") = "+ e1.ajoute(a) );
   }
}
```

1. Quels sont les attributs dont disposent les classes `Enfant1` et `Enfant2` ?
2. Ecrivez le résultat de l'exécution de la classe `Essai`.

4. Cours 2010 (5 points)

- Quelle est la différence entre une classe et un objet ?

- En quoi l'héritage est-il un outil puissant pour le développement d'applications ?
- Quelle est la différence entre
  - Point p[][] = new Point[5][4] ;
  - Point p = new Point(5,4) ;

5. Généricité et figures 2010 (4 points)

On souhaite donner la possibilité d'uniformiser les sommets d'un polygone en forçant ceux-ci à être des pointNommé ou des pointPondéré, … Pour cela on utilise une version générique de la classe polygone. Les sommets sont stockés dans une liste chaînée et le paramètre générique doit garantir que les sommets soient des points.

1. Ecrivez l'entête de la classe polygone générique
2. Ecrivez la déclaration de la liste chaînée
3. Ecrivez le constructeur de la classe polygone
4. Ecrivez la méthode getPoints() qui renvoie la liste de points

6. Clonage et liste chaînée 2010(5 points)

Soit A une classe implémentant l'interface Cloneable, et B une classe n'implémentant pas l'interface Cloneable, le code suivant est extrait de la méthode main de la classe test

```
LinkedList Liste = new LinkedList ();

Liste.add(new A());

Liste.add(new B());

LinkedList Liste2 = (LinkedList) Liste.clone();
```

1. Sachant que ce code ne renvoie pas d'erreur ni à la compilation ni à l'exécution, expliquez sont fonctionnement et l'état de la mémoire.
2. Comment s'appelle le clonage réalisé par la classe LinkedList ?
3. Si on voulait réaliser l'autre type de clonage sur les LinkedList, quelle condition serait nécessaire sur les objets passés à la liste ?
4. Comment s'appelle le `(LinkedList)` devant `Liste.clone()` et pourquoi est-ce nécessaire dans le cas de la method clone().

7. Les poupées russes 2010(7 points)

On veut modéliser le fonctionnement d'un ensemble de poupées russes. Une poupée russe est une sorte de boîte. Il en existe de différentes tailles et on peut placer une poupée dans une autre plus grande dès lors que la plus petite est fermée et la plus grande ouverte.

1. Quels sont les caractéristiques qui définissent une poupée russe ou son état ?
2. Ecrire la classe PoupeeRusse contenant les méthodes suivantes
   a. Constructeur

b. Public void ouvrir() : ouvre la poupée si elle n'est pas déjà ouverte et si elle ne se trouve pas à l'intérieur d'une autre poupée
c. Public void fermer() : ferme la poupée si elle n'est pas déjà fermée et si elle ne se trouve pas à l'intérieur d'une autre poupée
d. Public void placerDans(PoupeeRusse p) : place la poupée courante dans la poupée p si elle n'est pas déjà dans une autre et si la poupée p ne contient aucune autre poupée et si la poupée courante est fermée et la poupée p est ouverte et plus grande
e. Public void sortirDe(PoupeeRusse p) : sort la poupée courante de la poupée p si elle est dans p et si p est ouverte

Vous devez ajouter les attributs dont vous avez besoin.

8. Les poupées russes 2 2010(6 points)

On souhaite modéliser à nouveau le fonctionnement des poupées russes mais cette fois ci à l'aide d'une structure de données.

Pour cela on définit la classe ListeDePoupees qui hérite de la version générique de la classe LinkedList. Le paramètre générique de la classe ListeDePoupees doit être défini de manière à n'autoriser que des poupées Russes à être ajoutées à la liste.

L'objectif est de modifier la méthode add() de la classe LinkedList de manière à réaliser les tests nécessaires avant l'ajout.

1. Donner l'entête de la classe ListeDePoupees
2. Donner le code et l'entête de la méthode add() de la classe ListeDePoupees qui permet d'ajouter en fin de liste la poupée passée en paramètre si les conditions suivantes sont respectées : la dernière poupée de la liste est fermée et a une taille inférieure à la poupée passée en paramètre, la poupée passée en paramètre est ouverte et vide. (on considère la classe PoupeeRusse déjà codée et possédant les méthodes nécessaires à son utilisation).
3. Si on souhaite garantir qu'une poupée n'est pas présente dans plusieurs listes de poupées (relation d'agrégation), comment peut-on procéder ?

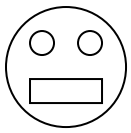
9. Cours 2009 (4 points)

1. Comment créer une constante partagée par toutes les instances d'une même classe ?
2. Expliquez chacun des termes de la déclaration de la méthode main
   public static void main(String[] args)

10. Les figures composées 2009 (6 points)

On souhaite ajouter la notion de figure composée. Comme son nom l'indique, une figure composée est une figure composée d'autres figures.

Par exemple, la figure suivante est composée de 3 cercles et d'un rectangle (même si le rectangle n'a pas été défini en TD, il peut être considéré comme un polygone).



- Quelle est la place de la classe FigureComposée dans la hiérarchie des Figures ?
- Indiquez ses attributs en utilisant les structures de données vues en cours.
- Détaillez la fonction d'affichage de ce nouveau Type de figure.

11.Les exceptions 2009 (5 points)

1. Définir une exception ExceptionNegatif. Cette classe aura simplement un attribut entier valeur et un constructeur avec un paramètre entier (correspondant à valeur).
2. Ecrire le code permettant de lever l'exception dans une méthode appelée *factoriel* si le paramètre passé à la méthode est négatif.
3. Ecrire le code permettant de récupérer l'exception et d'afficher la valeur (négative) ayant conduit à sa levée. On considère que la méthode *factoriel* se trouve dans la classe *Calcul*.

12. Cours 2008 (5 points)

- Citez et expliquez les différentes relations existant entre deux classes.
- Expliquez la notion de polymorphisme
- Qu'est-ce que la sérialisation. Donnez les différentes étapes pour la mettre en œuvre.

13. Analyse de code 2008 (4 points)

Indiquez si le code suivant est correct ou non. S'il ne l'est pas proposez une correction.

A.java

```
public class A {
       public int i ;
       private int j;
       void afficher(){
              System.out.println(i+ j) ; }
       public int getI() { return i;}
       public int getJ() { return j;}
       public void setI(int i) {this.i=i;}
       public void setJ(int j) {this.j=j;}
       }
```

B.java

```
public class B extends A{
       public int k ;
       void afficher() {
       System.out.println(i+ j+k);}
                     }
```

Test.java

```
  public class Test {

  public void main(String[] args) {
       A a = new A();

       a.i = 1;

       a.j = 2;

       a.afficher();

       a = new B();

       a.i = 3;

       a.j = 4;

       a.k = 5;

       a.afficher();

  }
}
```

14. Singleton 2008 (6 points)

Soit le code suivant

```
public class Singleton
{
      static final private Singleton INSTANCE=new Singleton();
      … // Autres attributs
      private Singleton() { … };
      public static Singleton getInstance() {return INSTANCE;}
      … // Autres méthodes }
```

Que se passe t il à la compilation si on fait l'appel suivant à l'extérieur de la classe Singleton :
`Singleton S = new Singleton();` ?

D'après vous pourquoi le constructeur de la classe est il privé ?

Combien d'instances différentes de Singleton peut-il y avoir en tout dans le programme ?

15. Segments et exceptions 2008 (5 points)

On veut protéger la classe Segment pour éviter que les deux points ne soient confondus (Segment réduit à un point). Pour cela on définit une exception : PointsConfondusException.

1. Ecrire la classe PointsConfondusException

2. Ecrire un constructeur de la classe segment qui lève une exception si les points passés en paramètre sont confondus.

3. Ecrire la méthode « setP1 » qui modifie le point P1

4. Ecrire l'appel au constructeur de Segment (déclaration et instanciation d'un objet de type Segment).

16. Cours 2007 (5 points)

- Expliquez les deux types de clonage que vous connaissez
- Expliquez la notion de polymorphisme

17. Complément sur les figures 2007 (10 points)

On souhaite ajouter la possibilité de transférer des figures sur le réseau

1. Indiquez les différentes étapes pour réaliser ce transfert (classes à modifier, classes à créer (éventuellement), objets à créer pour le transfert.
2. On souhaite ajouter la notion de figure complexe (composée de plusieurs figures de base). En utilisant les structures de données et la hiérarchie des figures vue en TD, indiquez comment et où vous voulez y insérer la classe FigureComplexe. Listez les attributs et les méthodes de cette classe. Donnez le code des principales méthodes.

18. Analyse de code 2007 (5 points)

Soit le code Java suivant:

```
Public PolygoneSegment(Point[] t)
        {
        segments = new Segment[t.length];
        for (int i=0; i<segments.length; i++)
        {
                segments[i] = new Segment(t[i], t[(i+1) % t.length]);
        }
}
```

Commentez le plus précisément possible le code (type de méthode, type des variables, fonctionnement, rôle des appels de fonction, …).

Indiquez comment doit être déclaré l'objet « segments ».

19. Cours 2011(4 points)

1. Expliquez la notion d'Exception en java (fonctionnement, déclaration, …).
2. Qu'est-ce qu'une classe anonyme ?

20. Interfaces graphiques 2011 (6 points)

On souhaite définir un nouveau composant qui se comporterait comme un bouton mais qui serait autonome du point de vue des traitements à effectuer en cas d'actionnement de celui-ci.

Ce bouton est associé à une figure qui sera définie lors de son instanciation.

On suppose définie dans la classe figure une méthode saisie() qui affiche une interface graphique permettant la saisie des attributs de la figure.

Lorsque l'on clique sur le bouton celui-ci demande la saisie des attributs de la figure, puis demande un réaffichage du composant.

1. Rappelez le type d'évènement généré lors du clic sur un bouton.
2. Ecrivez la déclaration de ce nouveau type de composant (entête de la classe et attributs).
3. Ecrivez le constructeur de la classe.
4. Ecrivez la méthode appelée lors du clic sur le composant.
5. Quelle méthode est appelée lors de l'affichage du composant ? Quel est le paramètre passé à cette méthode et à quoi sert-il ?
6. Illustrez l'utilisation de ce type de composant (instanciation et ajout dans un JFrame par exemple).

21. Extraction d'une liste de points 2011 (5 points)

On souhaite créer une liste de points à partir de toutes les figures créées à l'aide de l'éditeur de figure.

1. On ajoute une méthode getPoints() dans chaque figure qui renvoie la liste des points de cette figure. Codez cette méthode pour les classes Point, Segment, Polygone et Cercle
2. Codez la méthode createListe() prenant en paramètre une liste de Figure et permettant de construire la liste des Points

22. SauveQuiPeut 2011 (5 points)

On souhaite coder de deux manières la méthode SauveQuiPeut dont l'entête est : *public void SauveQuiPeut(Collection Col, ObjectOutputStream Out) { …}* et qui permet de sauver tous les objets serialisables de *Col* dans le flux Out.

1. Coder cette méthode en utilisant instanceOf
2. Coder cette méthode en utilisant les exceptions

23. Les exceptions 2010 (6 points)

On suppose définie une liste de figures : *LinkedList<Figure> figures ;*

On souhaite afficher la liste des noms des figures. Certaines figures possèdent un nom (cercleNommé, polygoneNommé, …) et d'autres pas (point, …). Les figures possédant un nom implémentent l'interface *Nomme* qui possède une méthode *public void String getNom()*.

1. En utilisant *instance of*, coder la méthode *public String Noms()* qui concatène les noms de toutes les figures qui en possède un.

2. En utilisant les exceptions (et sans utiliser *instance of*), coder la même méthode. On rappelle que lorsqu'un appel est fait sur une méthode qui n'est pas présente dans une classe, l'exception NoSuchMethodException est levée.

24. Les interfaces graphiques 2010 (5 points)

1. Expliquer brièvement le fonctionnement de l'interaction entre un utilisateur et une interface graphique en java.
2. On souhaite coder dans un unique fichier une interface graphique possédant un bouton appelé *inverse* qui affiche alternativement « Bonjour » et « Au revoir » dans un *Jlabel* (pour changer le texte du *Jlabel* on utilise la méthode *setText()* prenant en paramètre une chaîne de caractère).

25. Les structures de données 2010 (6 points)

1. Coder une méthode prenant en paramètre : une structure de donnée ne pouvant contenir que des *Figures* et une *Figure F*. La méthode renvoie le nombre de Figures de même type que *F* contenues dans la structure. La méthode getClass() est présente dans toutes les classes java et renvoie la classe de l'objet appelant la méthode (exemple : Integer I ; I.getClass() renvoie Integer).
2. On souhaite stocker les Figures dans une liste triée (*sortedSet*). Quelle interface doivent implémenter les figures pour que cela puisse fonctionner ?
3. On suppose codée la méthode *getPoids()* qui renvoie le poids de la figure (nombre de points de celle-ci par exemple). Coder la méthode présente dans l'interface précédente et permettant de faire la comparaison entre deux figure en utilisant le poids.

26. La sérialisation 2010 (3 points)

1. Expliquer le fonctionnement de la sérialisation.
2. Indiquer le contenu du fichier (en expliquant) après les opérations suivantes
    a. Point P= new Point(1,2) ;
    b. Ouverture d'un accès à un fichier *F*
    c. Sérialisation de *P* dans le fichier *F*
    d. P.translater(1,1)
    e. Sérialisation de *P* dans le fichier *F*
    f. Fermeture du fichier *F*

27. Réseau 2009 (7 points)

On cherche à produire un éditeur de figures coopératif. Chaque intervenant est donc sur sa propre machine et édite des figures sur un dessin global partagé.

On suppose que l'on a une architecture client/serveur (**une** application **serveur** et ***n*** applications **clientes)**.

1. On ne veut pas que les figures soient présentes en double dans notre collection. Quelle structure de données faut-il utiliser ?

2. Le serveur possède une liste des clients connectés, donner la déclaration et l'instanciation de cette structure de données.
3. Initialisation
    a. Le serveur créé et initialise une collection (à définir) de figures puis attend les demandes de connexion des clients.
    b. L'application cliente établit une connexion avec le serveur.
    c. Donner le code correspondant à chacune de ces étapes.
4. Echange des données
    a. Le serveur lit les figures ajoutées par chacun des clients puis constitue un nouvel ensemble qu'il envoie vers chacun des clients.
    b. Le client envoie sa propre liste de figures vers le serveur puis récupère la liste complète depuis le serveur et l'affiche.
    c. Donner le code correspondant à chacune de ces étapes

28. Thread 2009 (5 points)

On considère que plusieurs threads peuvent ajouter des figures dans une même liste appelée *listeFig*. Pour cela ils utilisent la méthode *public void add(Figure f)*.

1. Donner la déclaration et l'instanciation de *listFig.*
2. Donner le code de la méthode *add()*
3. Comment peut-on éviter qu'il y ait des conflits entre les différents threads lors de l'ajout des figures dans la liste ?
4. On ajoute un thread donc la fonction est de parcourir la liste des figures et de supprimer les figures trop proches. Coder la méthode *filtrer()* qui filtre la liste et renvoie le nombre d'éléments supprimés.

29. Figure récursive 2009 (5 points)

On souhaite définir la notion de figure récursive (type fractale). Une figure récursive est définie par un ensemble de figures et un nombre de répétitions (*nb*). On affiche une figure récursive en affichant les figures qui la composent à différents niveau de résolution (*nb*). On considère pour cela que l'on dispose d'une méthode *réduction()* qui prend en paramètre une figure et renvoie sa réduction à un niveau de résolution juste inférieur.

1. Donner la place de la classe *figureRecursive* dans la hiérarchie des figures.
2. Donner la déclaration des attributs de cette classe.
3. Donner le code de la méthode *public void paint(Graphics g)* de cette classe qui affiche la figure récursive sur les *nb* niveaux.

30. Base de documents (20 points)

On s'intéresse au système d'information d'une médiathèque dans lequel un ensemble de *documents* est répertorié : des livres, films, ou disques. Ces documents peuvent être empruntés par les clients de la

Médiathèque.

Les documents sont tous caractérisés par un titre et un auteur. Du point de vue programmation, un *document* est caractérisé par l'interface ci-dessous :

```
import Auteur;
public interface Document {
/** @return l'auteur de ce document */
public Auteur getAuteur();
/** @return le titre de ce document */
public String getTitre();
/** @return une chaîne de caractères représentant ce document */
public String toString();
/** @param l'objet à tester
* @return true si le paramètre est un document identique à this */
public boolean equals(Object o);
/** indique si le document est emprunt´e ou non
* @return true ssi le document est emprunté
*/
public boolean estEmprunte();
/** permet de marquer comme emprunté ou non un document
* @param b true si le document est emprunté, false si il est disponible
*/
public boolean setEmprunte(boolean b);
}
```

1. (3 points) La classe `Auteur` permet de représenter les auteurs des documents. Un auteur est défini par son nom, de type chaîne de caractères, ses dates de naissance et de décès de type `Date`.

La valeur de la date de décès est `null` si l'auteur est encore vivant. On veut pouvoir accéder aux différentes informations sur un auteur et tester si deux objets auteurs sont égaux (mêmes nom et dates de naissance/décès).

La classe `Date` est définie en annexe.

Donnez un code java complet pour la classe `Auteur`.

2. (2 points) Le type Mois est une énumération. Qu'est-ce que cela signifie ? Définir le type Mois.

3. (5 points) Définir une classe BaseDocumentaire comportant une liste de Documents (utiliser la version générique des listes). Cette classe permet de rechercher des documents dans la liste par leur titre ou leur auteur (méthode *recherche*), permettant de connaître les documents empruntés ou non empruntés (méthodes *rechercheEmprunte* et *rechercheNonEmprunte*).

Coder chacune des méthodes (penser au type de donnée renvoyé par les méthodes).

4. (5 points) On souhaite que les méthodes de recherche renvoient une exception (NoDocumentFoundException) à la place de null. Expliquer les étapes nécessaires pour procéder de cette manière. Modifier une des méthodes de recherche en conséquence.

Que doit faire l'utilisateur de ces méthodes ?

Ecrire le code correspondant.

5. (5 points) On veut définir plusieurs types de documents (livre, vidéo (avec ou sans sous titres), enregistrement audio, photo, …). Définir le diagramme de ces classes.

Ces documents peuvent être lus, regardés ou écoutés. Comment définir ces propriétés sans utiliser d'attributs ? Compléter le daigramme.

Annexe : définition UML de la classe Date

Date

- jour : int
- mois : Mois
- annee : int
+ Date(j : int, mois : Mois, annee : int)
+ toString() : String
+ equals(o : Object) : boolean
+ getJour() : int
+ getMois() : Mois
+ getAnnee() : int
+ compareTo(o : Object):int
+ differenceEnJours(d : Date):int
+ static aujourdhui():Date