1) Héritage - Visibilité

la table de visibilité selon les droits d'accès (niveaux de visibilité):

Visibilité						
Modificateur	Classe	Paquetage	Sous-classe	Partout		
private	Oui	Non	Non	Non		
Rien	Oui	Oui	Non	Non		
protected	Oui	Oui	Oui	Non		
Public	Oui	Oui	Oui	Oui		

Indiquez par Oui ou Non après les signes de commentaires (//) si les attributs de la classe Personne sont accessibles ou visibles dans les classes FaculteSciences, Employe, DepartementInformatique, Enseignant, et Universite.

```
package pl;
                                                 package p1;
public class Personne {
                                                 public class FaculteSciences {
       protected String nom ;
       public String prenom ;
                                                   public void estAccessible(){
       private int age ;
       double taille;
                                                      Personne p = new Personne();
                                                      p.nom = "Daniel";
                                                     p.prenom = "Diderot";
                                                            = 99;
                                                      p.age
                                                      p.taille = 1.83;
package p2;
                                                 package p2;
                                                 import p1.Personne;
import p1.Personne;
public class Employe extends Personne {
                                                 public class DepartementInformatique {
  public void estAccessible(){
                                                  public void estAccessible(){
           = "Daniel";
                                                      Personne p = new Personne();
     nom
     prenom = "Diderot";
                                                              = "Daniel";
                                                      p.nom
                                                      p.prenom = "Diderot";
           = 99;
     age
     taille = 1.83;
                                                      p.age = 99;
                                                      p.taille = 1.83;
  }
package p3;
                                                 package p3;
import p2.Employe;
                                                 import p2.Employe;
public class Enseignant extends Employe {
                                                 public class Universite {
  public void estAccessible(){
                                                  public void estAccessible(){
           = "Daniel";
                                                      Personne p = new Personne();
     nom
                                                      p.nom = "Daniel";
     prenom = "Diderot";
          = 99;
                                                      p.prenom = "Diderot";
     age
                                                             = 99;
     taille = 1.83;
                                                      p.age
                                                      p.taille = 1.83;
```

2) Héritage - Constructeurs des superclasses & sous-classes

Selon la définition des constructeurs de la superclasse Personne, indiquez par Oui ou Non si la classe Employe compile sans aucune erreur de compilation.

a)

```
package constrcuteur;
                                                  package constructeur;
                                                  public class Employe extends Personne {
public class Personne {
   private String prenom;
                                                    private String matricule;
   private String nom;
                                                    private float salaire;
   public Personne(){
                                                    public Employe(){
                                                      super();
   public Personne(String prenom,
                   String nom) {
                                                    public Employe(String prenom, String nom,
                                                                  String matricule, float salaire ){
       this.prenom = prenom;
       this.nom = nom;
                                                      super(prenom, nom);
                                                      this.matricule = matricule;
                                                      this.salaire = salaire;
                                                    }
                                                       OUI
                                                                         NON
```

b)

```
package constrcuteur;
                                                  package constructeur;
                                                  public class Employe extends Personne {
public class Personne {
   private String prenom;
                                                    private String matricule;
   private String nom;
                                                    private float salaire;
                                                    public Employe(){
   public Personne(String prenom,
                   String nom) {
       this.prenom = prenom;
       this.nom = nom;
                                                    public Employe(String prenom, String nom,
                                                                  String matricule, float salaire ){
                                                      super(prenom, nom);
                                                      this.matricule = matricule;
                                                      this.salaire = salaire;
                                                    }
                                                       OUI
                                                                         NON
```

c)

d)

```
package constrcuteur;
                                                 package constructeur;
public class Personne {
                                                 public class Employe extends Personne {
  private String prenom;
                                                   private String matricule;
   private String nom;
                                                   private float salaire;
   public Personne(String prenom,
                   String nom) {
       this.prenom = prenom;
                                                   public Employe(String prenom, String nom,
       this.nom = nom;
                                                                 String matricule, float salaire ){
                                                     this.matricule = matricule;
                                                     this.salaire = salaire;
                                                   }
                                                       OUI
                                                                         NON
```

e)

```
package constrcuteur;
                                                 package constructeur;
public class Personne {
                                                 public class Employe extends Personne {
  private String prenom;
                                                   private String matricule;
   private String nom;
                                                   private float salaire;
                                                   public Employe(){
                                                     super();
                                                   public Employe(String prenom, String nom,
                                                                 String matricule, float salaire ){
                                                     super();
                                                     this.matricule = matricule;
                                                     this.salaire = salaire;
                                                       OUI
                                                                        NON
```

3) Héritage - Redéfinition (masquage) / Surcharge (surdéfinition)

```
package redefinition;
                                                  package redefinition;
public class Personne {
                                                  public class Employe extends Personne {
                                                   public boolean equals(Object obj)
public boolean equals(Object obj) {
                                                     System.out.println("Employe-equals(Object)");
  System.out.println("Personne-equals(Object)");
                                                     return true;
  return true;
                                                   public String toString()
                                                     System.out.println("Employe-toString()");
public String toString() {
                                                     return "";
   System.out.println("Personne-toString()");
   return "";
package redefinition;
                                                  package redefinition;
public class Enseignant extends Employe {
                                                  public class TestRedefinition {
                                                    public static void main(String[] args) {
public boolean equals(Object obj) {
  System.out.println("Enseignant-equals(Object)");
                                                      Object o = null;
  return true;
                                                      Personne pers = new Personne();
                                                      pers.equals(o);
public String toString() {
                                                      pers.toString();
  System.out.println("Enseignant - toString()");
  return "";
                                                      Employe emp = new Employe();
                                                      emp.equals(o);
                                                      emp.toString();
                                                      Enseignant ens = new Enseignant();
                                                      ens.equals(o);
                                                      ens.toString();
                                                      pers = emp;
                                                      pers.equals(o);
                                                      pers.toString();
                                                      pers = ens;
                                                      pers.equals(o);
                                                      pers.toString();
a) Les méthodes equals et tostring sont redéfinies dans les classes personne, Employe, et
   Enseignant. Donnez ce qui sera affiché après l'exécution des instructions de la méthode main
   de la classe TestRedefinition.
```

```
package surcharge;
                                                    package surcharge;
public class Personne {
                                                    public class Employe extends Personne {
                                                     public void valider(Employe emp) {
public void valider(Personne p) {
                                                       System.out.println("Employe-valider(Employe)");
 System.out.println("Personne-valider(Personne)");
package surcharge;
                                                    package surcharge;
public class Enseignant extends Employe {
                                                    public class TestSurcharge {
public void valider(Enseignant ens) {
                                                      public static void main(String[] args) {
 System.out.println("Enseignant-valider(Enseignant)");
                                                        Personne pers = new Personne();
                                                        Employe emp = new Employe();
                                                        Enseignant ens = new Enseignant();
                                                        ens.valider(ens);
                                                        ens.valider(emp);
                                                        ens.valider(pers);
b) La méthode valider est surchargée dans les classes Employe, et Enseignant. Donnez ce qui
   sera affiché après l'exécution des instructions de la méthode main de la classe TestSurcharge.
```

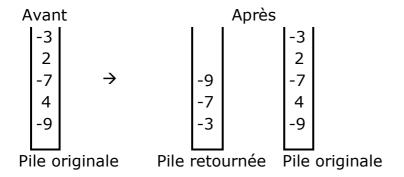
```
package surcharge;
                                                     package surcharge;
public class Personne {
                                                     public class Employe extends Personne {
                                                      public void valider(Object o) {
public void valider(Personne pers) {
                                                        System.out.println("Employe-valider(Object)");
   System.out.println("Personne-valider(Personne)");
                                                      public void comparer(Employe emp) {
public void comparer(Personne pers) {
                                                       System.out.println("Employe-comparer(Employe)");
   System.out.println("Personne-comparer(Personne)");
                                                      public void demenager(Object o) {
public void demenager(Personne pers) {
                                                       System.out.println("Employe - demenager(Object)");
 System.out.println("Personne-demenager(Personne)");
package surcharge;
                                                        package surcharge;
public class Enseignant extends Employe {
                                                        public class TestSurcharge {
public void valider(Employe emp) {
                                                          public static void main(String[] args) {
  System.out.println("Enseignant - valider(Employe)");
                                                            Employe emp = new Employe();
                                                            Enseignant ens = new Enseignant();
public void comparer(Object o) {
   System.out.println("Enseignant - comparer(Object)");
                                                            ens.valider(ens);
                                                            ens.comparer(ens);
public void demenager(Enseignant ens)
                                                            ens.demenager(emp);
   System.out.println("Enseignant-demenager(Enseignant)");
c) Les méthodes valider, comparer, et demenager sont surchargées dans les classes Employe, et
   Enseignant. Donnez ce qui sera affiché après l'exécution des instructions de la méthode main
   de la classe TestSurcharge.
```

```
package surcharge;
                                                      package surcharge;
public class Personne {
                                                     public class Employe extends Personne {
                                                      public void valider(Object o) {
public void valider() {
                                                        System.out.println("Employe-valider(Object)");
   System.out.println("Personne-valider()");
                                                      public void comparer(Personne pers) {
public void comparer(Object o) {
                                                       System.out.println("Employe-comparer(Personne)");
   System.out.println("Personne-comparer(Object)");
                                                      public void demenager() {
public void demenager(Personne pers) {
                                                       System.out.println("Employe - demenager()");
 System.out.println("Personne-demenager(Personne)");
package surcharge;
                                                        package surcharge;
public class Enseignant extends Employe {
                                                        public class TestSurcharge {
public void valider(Personne pers) {
                                                          public static void main(String[] args) {
  System.out.println("Enseignant - valider(Personne)");
                                                            Enseignant ens = new Enseignant();
                                                            ens.valider(ens);
public void comparer(Object o) {
                                                            ens.comparer(ens);
   System.out.println("Enseignant - comparer(Object)");
                                                            ens.demenager(ens);
public void demenager(Object o) {
   System.out.println("Enseignant-demenager(Object)");
                                                        }
d) Les méthodes valider, comparer, et demenager sont surchargées dans les classes Employe, et
   Enseignant. Donnez ce qui sera affiché après l'exécution des instructions de la méthode main
   de la classe TestSurcharge.
```

4) Utilisation du TDA Pile<T> et son implantation PileArrayListImpl<T>

En utilisant les services de manipulation de pile vus en classe et son implémentation en ArrayList (Pile<T>, PileArrayListImpl<T>), écrire une méthode obtenirLesNegatifs qui reçoit en paramètre une pile d'entiers (Integer), et retourne une pile de tous les entiers négatifs qui se trouvent dans la pile passée en paramètre. Si la pile passée en paramètre est vide ou nulle, la méthode retourne une référence nulle. La pile passée en paramètre doit rester inchangée à la fin de l'opération.

Ci-dessous un exemple d'appel de la méthode obtenirLesNegatifs.



```
public static Pile<Integer> obtenirLesNegatifs(Pile<Integer> pl) {
```

5) Utilisation du TDA File<T> et son implantation FileArrayListImpl<T>

En utilisant les services de manipulation de file vus en classe et son implémentation en ArrayList (File<T>, FileArrayListImpl<T>), écrire une méthode existeElement qui reçoit en paramètres une file de String uneFile, un élément de type String s et vérifie si cet élément existe dans la file. La méthode retourne vrai (true) si l'élément existe, sinon elle retourne faux. La file doit rester inchangée à la fin de l'opération de vérification. Les paramètres uneFile et s peuvent être nuls ou vides.

Par exemple, en appelant la méthode existeElement(uneFile, "Q") où uneFile contient les éléments o, P, Q, R, s (d'où o est le premier de la file), la méthode retourne vrai. Par contre, si on appelle la méthode existeElement(uneFile, "Z") avec la même file, la méthode retourne faux.

public	static	boolean	existeElement(File <string< th=""><th>uneFile,</th><th>String s)</th><th>{</th></string<>	uneFile,	String s)	{
}						

6) Récursivité - Méthodes et traces d'exécution

a) Ci-dessous la méthode itérative afficher qui prend en paramètre une valeur entière n supérieure ou égale à 0 et affiche les valeurs de 1 à n.

```
public static void afficher (int n) {
    for (int i = 1; i <= n; i++) {
        System.out.println(i);
    }
}</pre>
```

Écrire la version récursive de la méthode afficher.

```
public static void afficherRecursive(int n) {
```

b) Écrire une méthode récursive public static String inverserChaine (String chaine) qui reçoit en paramètre une chaîne de caractères et retourne une nouvelle chaîne qui est l'inverse de la chaîne reçue en paramètre. Le paramètre chaine ne peut pas être nul ou vide.

```
public static String inverserChaine(String chaine) {
```

-,	vrai si la chaîne reçue en paramètre est un palindrome, sinon faux. Une chaîne est un palindrome si elle est égale à son inverse Ex: laval = laval (inversé). Le paramètre
	chaine ne peut pas être nul ou vide.
	<pre>public static boolean estUnPalindrome(String chaine) {</pre>
	}

c) Écrire la méthode public static boolean estUnPalindrome (String chaine) qui retourne