Révision

POO-Java Niveau 1

```
class Petite {
protected String n ;
String getN() {return n;}
public Petite(String x) { n = x ; }
public String toString() { return "Petite ! "; }
public void meth1() {
System.out.println("Petite.meth1 : " + this+" "+ n );
}

}
public class Grande extends Petite{
private int n ;
private int cpt=1;
int getNG() {return n;}
public Grande(String a, int b) { super(a) ; this.n = b ; cpt++; }
public String toString() { return "Grande ! "+"cpt= "+cpt; }
public void meth1() {
super.meth1() ;
System.out.println(
"Grande.meth1 : " + this +" "+ super.toString()+" "+n+" "+ super.n);
}
public void test() {  System.out.println("Grande.test : " + this);  }
```

```
public static void main(String args[]) {
Grande m = new Grande("POO", 2) ;
System.out.println(m) ;
m.meth1();
Petite n = new Petite("Java") ;
System.out.println(n) ;
n.meth1() ; }
```

Indiquez si les instructions suivantes sont correctes ou pas (ce qui se passe à la compilation et à l'exécution) et, pour les instructions qui vous semblent correctes, indiquez ce qui serait affiché, si il y a affichage.

```
class Petite {
protected String n ;
String getN() {return n;}
public Petite(String x) { n = x ; }
public String toString() { return "Petite ! "; }
public void meth1() {
System.out.println("Petite.meth1 : " + this+" "+ n );}}

public class Grande extends Petite{
private int n ;
private int cpt=1;
int getNG() {return n;}
public Grande(String a, int b) { super(a) ; this.n = b ; cpt++; }
public String toString() { return "Grande ! "+"cpt= "+cpt; }
public void meth1() {
super.meth1() ;
System.out.println(
"Grande.meth1 : " + this +" "+ super.toString()+" "+n+" "+ super.n);}
public void test() {  System.out.println("Grande.test : " + this);  }
```

	Compilation	Exécution
1) Petite v = new Grande("POO", 2) ;		
2) System.out.println("n vaut: "+v.n); System.out.println("cpt vaut : "+v.cpt) ;		

Indiquez si les instructions suivantes sont correctes ou pas (ce qui se passe à la compilation et à l'exécution) et, pour les instructions qui vous semblent correctes, indiquez ce qui serait affiché, si il y a affichage.

```
class Petite {
protected String n ;
String getN() {return n;}
public Petite(String x) { n = x ; }
public String toString() { return "Petite ! "; }
public void meth1() {
System.out.println("Petite.meth1 : " + this+" "+ n );}}

public class Grande extends Petite{
private int n ;
private int cpt=1;
int getNG() {return n;}
public Grande(String a, int b) { super(a) ; this.n = b ; cpt++; }
public String toString() { return "Grande ! "+"cpt= "+cpt; }
public void meth1() {
super.meth1() ;
System.out.println(
"Grande.meth1 : " + this +" "+ super.toString()+" "+n+" "+ super.n);}
public void test() {  System.out.println("Grande.test : " + this);  }
```

	Compilation	Exécution
1) Petite v = new Grande("POO", 2) ;	ok upcasting	ok
2) System.out.println("n vaut: "+v.n); System.out.println("cpt vaut : "+v.cpt) ;	ok not ok	n vaut: POO

```
class Petite {
protected String n ;
String getN() {return n;}
public Petite(String x) { n = x ; }
public String toString() { return "Petite ! "; }
public void meth1() {
System.out.println("Petite.meth1 : " + this+" "+ n );}}

public class Grande extends Petite{
private int n ;
private int cpt=1;
int getNG() {return n;}
public Grande(String a, int b) { super(a) ; this.n = b ; cpt++; }
public String toString() { return "Grande ! "+"cpt= "+cpt; }
public void meth1() {
super.meth1() ;
System.out.println(
"Grande.meth1 : " + this +" "+ super.toString()+" "+n+" "+ super.n);}
public void test() {  System.out.println("Grande.test : " + this);  }
```

	Compilation	Exécution
3) v.test() ;		
4) Grande maV=(Grande)v ; maV.test() ;		
5) System.out.println(maV) ;		

```
class Petite {
protected String n ;
String getN() {return n;}
public Petite(String x) { n = x ; }
public String toString() { return "Petite ! "; }
public void meth1() {
System.out.println("Petite.meth1 : " + this+" "+ n );}}

public class Grande extends Petite{
private int n ;
private int cpt=1;
int getNG() {return n;}
public Grande(String a, int b) { super(a) ; this.n = b ; cpt++; }
public String toString() { return "Grande ! "+"cpt= "+cpt; }
public void meth1() {
super.meth1() ;
System.out.println(
"Grande.meth1 : " + this +" "+ super.toString()+" "+n+" "+ super.n);}
public void test() {  System.out.println("Grande.test : " + this);  }
```

	Compilation	Exécution
3) v.test() ;	Not ok	
4) Grande maV=(Grande)v ; maV.test() ;	ok ok	ok Grande.test : Grande ! cpt= 2
5) System.out.println(maV) ;	ok	Grande ! cpt= 2

```
class Petite {
protected String n ;
String getN() {return n;}
public Petite(String x) { n = x ; }
public String toString() { return "Petite ! "; }
public void meth1() {
System.out.println("Petite.meth1 : " + this+" "+ n );}}

public class Grande extends Petite{
private int n ;
private int cpt=1;
int getNG() {return n;}
public Grande(String a, int b) { super(a) ; this.n = b ; cpt++; }
public String toString() { return "Grande ! "+"cpt= "+cpt; }
public void meth1() {
super.meth1() ;
System.out.println(
"Grande.meth1 : " + this +" "+ super.toString()+" "+n+" "+ super.n);}
public void test() {  System.out.println("Grande.test : " + this);  }
```

	Compilation	Exécution
6) Petite v1= new Petite("essai") ; Grande varm=(Grande) v1 ; varm.test() ;		
7) Object obj=(Object)v ;		
8) System.out.println(obj) ;		

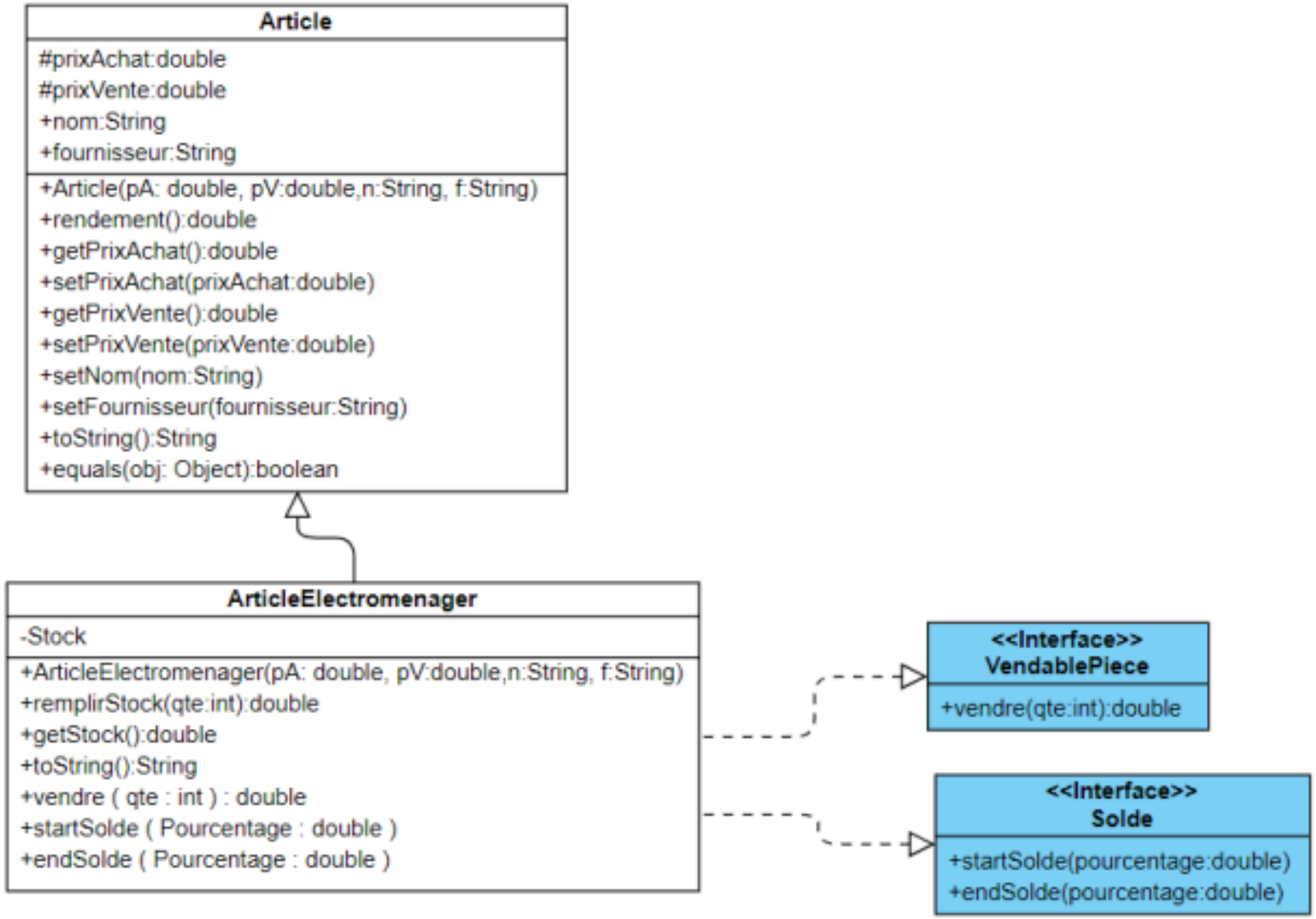
```
class Petite {
protected String n ;
String getN() {return n;}
public Petite(String x) { n = x ; }
public String toString() { return "Petite ! "; }
public void meth1() {
System.out.println("Petite.meth1 : " + this+" "+ n );}}

public class Grande extends Petite{
private int n ;
private int cpt=1;
int getNG() {return n;}
public Grande(String a, int b) { super(a) ; this.n = b ; cpt++; }
public String toString() { return "Grande ! "+"cpt= "+cpt; }
public void meth1() {
super.meth1() ;
System.out.println(
"Grande.meth1 : " + this +" "+ super.toString()+" "+n+" "+ super.n);}
public void test() {  System.out.println("Grande.test : " + this);  }
```

	Compilation	Exécution
6) Petite v1= new Petite("essai") ; Grande varm=(Grande) v1 ; varm.test() ;	Ok Ok Ok	 Not ok : Petite cannot be cast to Grande not ok
7) Object obj=(Object)v ;	Ok	OK
8) System.out.println(obj) ;	OK	Grande ! cpt= 2

Exercice 2

Écrire le code java des classes suivantes décrivant différents types d'articles.



Exercice 2

1) (7points) Écrivez une classe ***Article*** avec quatre variables **prixAchat**: le prix pour lequel le supermarché achète le produit, **prixVente**: le prix pour lequel le supermarché vend le produit, **nom**: le nom du produit, **fournisseur**: le nom du fournisseur du produit. Seuls le noms et le nom du fournisseur sont publiques.

a. Écrire son constructeur complet Article(double pa, double pv, String nom, String frx),

b. Implémentez la méthode rendement() qui permet de calculer le taux de rendement d'un article à savoir son prix de vente - son prix d'achat sur son prix d'achat,

c. Implémentez la méthode String toString() qui représente un article ainsi:

Nom du produit: ..., fournisseur: ..., prix d'achat: ..., prix de vente: ...

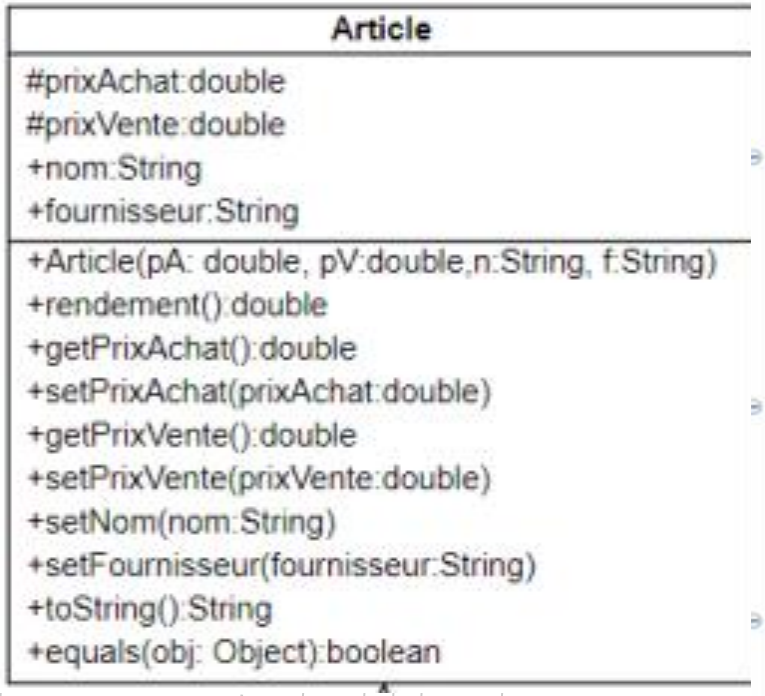
d. Implémentez les getters des prix de vente et d'achat.

Implémentez les setters de tous les attributs. En effet, les prix de vente et d'achat doivent être toujours *strictement positifs*,

le nom et le nom du fournisseur doivent être *non nuls*. Écrire le setter du nom en émettant *une exception adaptée*. Ecrire le setter du nom du fournisseur qui ne doit pas être null et dans ce cas il sera initialisé à une *chaîne vide*

e. Écrivez la redéfinition de la méthode equals() au niveau de la classe Article.

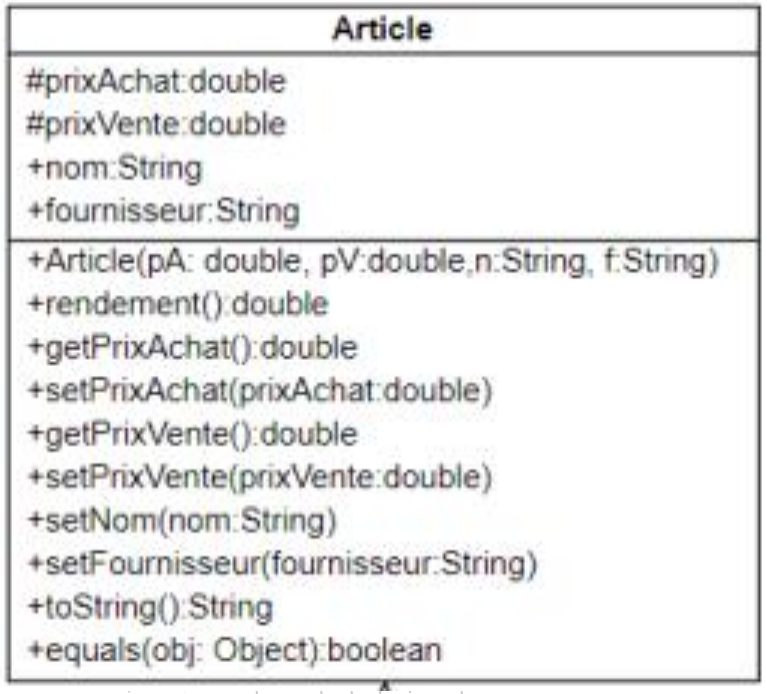
Exercice 2




```java
public class Article//0.5pt
{/**7Pt la classe**/
    /**
     *
     */
    protected double prixAchat;//0.25pt
    protected double prixVente;//0.25pt
    public String nom;//0.25pt
    public String fournisseur;//0.25pt
/**
 * le constructeur complet de la classe Article
 * @param pA le prix d'achat
 * @param pV le prix de vente
 * @param n le nom de l'article
 * @param f le nom du fournisseur
 */
    public Article(double pA, double pV, String n, String f) {//0.5pt
        setPrixAchat(pA);
        setPrixVente(pV);
        setNom(n);
        setFournisseur(f);
    }
    /**
     * la méthode rendement() qui permet de calculer
     * le taux de rendement d'un article à savoir son prix de vente - son prix d'achat sur son prix d'achat,
     * @return le rendement
     */
    public double rendement()    {//0.5pt
        return (prixVente - prixAchat) / prixAchat;
    }
```

```java
@Override
    public String toString() {//0.5pt
        return( "Nom du produit: " + nom+ " fournisseur: " + fournisseur+
                " Prix d'achat: " + prixAchat+ " Prix de vente: " + prixVente);
    }
```

d. Implémentez les getters des prix de vente et d'achat.
Implémentez les setters de tous les attributs. En effet, les prix de vente et d'achat doivent être toujours *strictement positifs*,




```java
/**
 *
 * @return le prix d'achat
 */
    public double getPrixAchat() {//0.25pt
        return prixAchat;
    }
/**
 * le setter du prix d'achat
 * @param prixAchat doit être positif
 */
    public void setPrixAchat(double prixAchat) {//0.5pt
        if(prixAchat>0) this.prixAchat = prixAchat;
    }
/**
 *
 * @return le prix de vente
 */
    public double getPrixVente() {//0.25pt
        return prixVente;
    }
/**
 * le setter du prix de vente
 * @param prixVente doit être positif
 */
    public void setPrixVente(double prixVente) {//0.5pt
        if(prixVente>0) this.prixVente = prixVente;
    }
```

le nom et le nom du fournisseur doivent être *non nuls*. Écrire le setter du nom en émettant *une exception adaptée*. Ecrire le setter du nom du fournisseur qui ne doit pas être null et dans ce cas il sera initialisé à une *chaîne vide.*

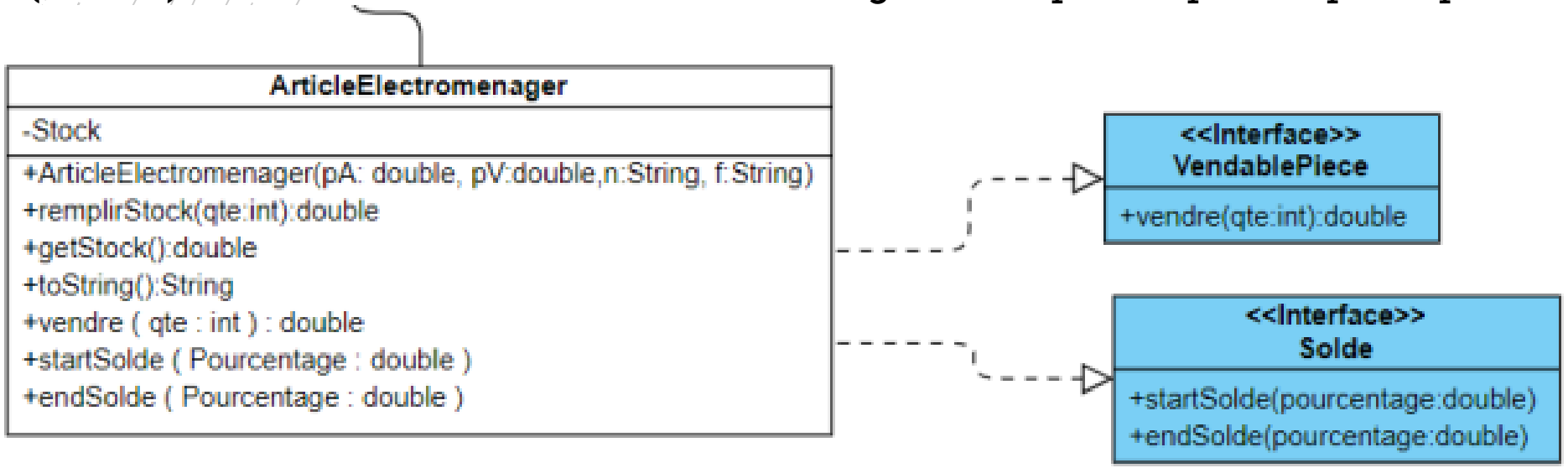
Écrivez la redéfinition de la méthode equals() au niveau de la classe Article.

```java
/**
 * le setter du nom
 * @param nom doit être non null, dans le cas contraire la méthode emet une exception
 * @throws IllegalArgumentException
 */
    public void setNom(String nom) throws IllegalArgumentException {//1pt
        if(nom==null)
            throw new IllegalArgumentException("La variable Objet ne doit pas etre null");
        else this.nom = nom;
    }
/**
 * le setter du nom du forunisseur de l'article
 * @param fournisseur doit être non null, dans le cas contraire il sera initialisé à une chaîne vide
 */
    public void setFournisseur(String fournisseur) {//0.5pt
        if(fournisseur!=null) this.fournisseur = fournisseur;
        else this.fournisseur="";
    }
/**
 * redéfinition de la méthode equals
 */
    @Override
    public boolean equals(Object obj) {//1pt
        if(this==obj) return true;
        if(!(obj instanceof Article)) return false;
        Article a= (Article) obj;
        return(a.fournisseur.equals(this.fournisseur)&&a.nom.equals(this.nom)&&
                        a.prixAchat==this.prixAchat&&a.prixVente==this.prixVente);
    }
```

2) (0,5point) Écrivez une interface ***VendablePiece***, Vendable par pièce: l'interface pour les produits qui se vendent par pièces avec une méthode vendre: cette méthode reçoit la quantité vendue du produit, retourne le revenu du magasin et modifie le stock si le stock est suffisant.

3) (1 point) Écrivez une interface ***Solde***, pour les articles susceptibles d'être vendu en solde avec deux méthodes:

- startSolde(double) ou lancer le solde: cette méthode baisse le prix du produit par le pourcentage donné
- endSolde(double) ou terminer le solde: cette méthode augmente le prix du produit par le pourcentage donné




```java
public interface Solde//1pt
{
    public abstract void StartSolde(double Pourcentage);
    public abstract void EndSolde(double Pourcentage);
}
```

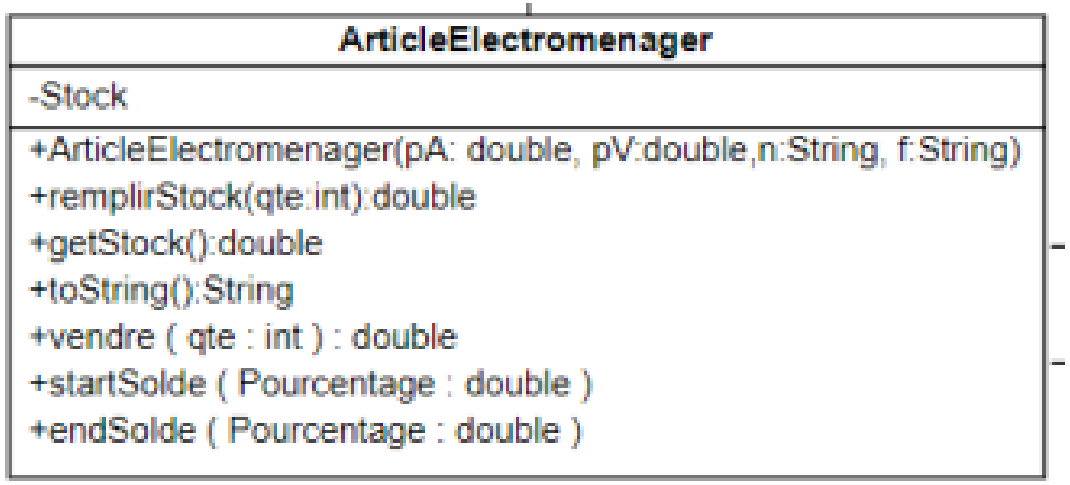
```java
public interface VendablePiece //0.5pt
{    //cette methode retourne le revenu de la vente
    public abstract double vendre(int qte);
}
```

4) (4,5points) Écrivez une classe ***ArticleElectromenager*** héritant de la classe Article avec un attribut privé stock (entier) et implémentant les deux interfaces VendablePiece et Solde.

- Écrivez le constructeur ArticleElectromenager(double pa, double pv,String nom, String frx) en utilisant un appel à super(...). Au moment de la construction de l'objet, le stock est vide.
- Implémentez la méthode remplirStock(qte:int) qui permet de remplir le stock avec la quantité positive qte et retournant le prix total d'achat,
- Écrivez aussi la méthode String toString() qui représente un article électroménager ainsi:

Nom du produit: ..., fournisseur: ..., prix d'achat: ..., prix de vente: ..., le stock du produit est...pièces.

- Implémentez le getter du stock.
- Implémentez les méthodes des deux interfaces.



```
class ArticleElectromenager extends Article implements VendablePiece, Solde //0.5pt
{//4.5pt la classe
    private int stock;//0.25pt
    // constructeur complet
    public ArticleElectromenager(double pA, double pV, String n, String f)     {//0.5pt
        super(pA,pV,n,f);
        stock = 0;  }
    public double RemplirStock(int qte)     { //0.5pt
        if(qte>0) {
            stock = stock + qte;
            return prixAchat * qte; }
        else return 0;
    }
    public int getStock()     {//0.25pt
      return stock;   }
    // implementation de l'interface vendablePiece
    public double vendre(int qte)    {//1pt
        if (qte<stock)          {
            stock = stock - qte;
            return qte * prixVente; }
        else {
            System.out.println("Stock insuffisant.");
            return 0;  }
    }
     // implementation de l'interface solde
    public void StartSolde(double Pourcentage)     {//0.5pt
        setPrixVente(prixVente*(1- Pourcentage/100));    }
    public void EndSolde(double Pourcentage)     {//0.5pt
        setPrixVente(prixVente*(1+ Pourcentage/100));    }
    public String toString() {//0.5pt
        return super.toString()+", le stock du produit est "+ stock+ " pieces.";}
 }
```

5. (3 points) Donnez l'affichage du code suivant :

```
Solde s=null;
        try {
            Article a= new ArticleElectromenager(2700,3500,"tv", "Samsung");
            Object o=new Article(500,1200,"Radio","Sony");
            System.out.println(o);
            s=(Solde)o;
        }
        catch(ClassCastException e) {
            System.out.println("dans le class cast");
        }
        catch(Exception e1) {
            System.out.println("dans exception");
        }
        finally {
            if(s==null)
                    System.out.println("s est null");
        }
```

Nom du produit : Radio fournisseur : Sony Prix d'achat : 500.0 Prix de vente : 1200.0 (1pt)
dans le class cast (1pt)
s est null (1pt)

```
class Factorielle{

 public static void main (String[] args)
   {   int i, nbEntiers=0, factorielle=1;
       try { nbEntiers= Integer.parseInt(args[0]);
       if(nbEntiers<0) throw new NumberFormatException(args[0]+
                " est négatif: la factorielle d'un nombre négatif n'est pas definie");
       for (i=2;i<= nbEntiers;i++)
            factorielle *= i;
        System.out.println("La factorielle de "+ nbEntiers +
                    " = "+ factorielle );
       }
         catch(ArrayIndexOutOfBoundsException e1) {
           System.out.println("Donnez en paramètre la base de la factorielle");
         }
         catch(NumberFormatException e) {
         if(nbEntiers<0)System.out.println(e.getMessage());
         else System.out.println("La base de la factorielle doit être ENTIERE");

         }
  catch(Exception e) {System.out.println(e.getMessage()); }
    }
}
```

exécutez le programme Factorielle.java ci-dessus en essayant successivement:

1. de ne pas passer en ligne de commande le paramètre attendu. Que se passe t'il ?
2. de mettre un paramètre non-entier. Que se passe t'il ?
3. de mettre un paramètre entier négatif. Que se passe t'il ?