ING 1

TD les interfaces

Écriture d'une interface

Soit le programme suivant :

```
public interface IVehicule {
    // indique le coût d'une amélioration
    public double coutAmelioration();

    // indique si le véhicule peut transporter ou non numPassagers
    public boolean peutTransporter(int numPassagers);
  }

  public class Voiture implements Vehicule {
    private int kilométrage;
    private int année;
    private int nombrePortes;

    // constructeur

    // indique si le véhicule est construite avant l'année passée en paramètre
    public boolean construitAvant(int autreAnnée) {
      return this.annéer < autreAnnée;
```

```
  }
}


public class Vélo implements IVehicule {
  private int kilométrage;
  private int nombreVitesses;


  // constructeur
}
```

- Quelles sont les méthodes à ajouter à Voiture et Vélo pour que le code se compile (en dehors des constructeurs?)
- Est-ce que `construitAvant` pourrait être ajouter à l'interface `iVéhicule` ?
- Dans une classe Exemple vous voulez créer un objet de type Vélo de la façon suivante :
  `_____ nouveauVelo =  new Vélo(0,1) ;`
  Quel(s) type(s) peuvent remplacer les tirets ?
- En supposant qu'un objet Voiture est défini de la manière suivante :
  `Ivehicule vielleVoiture = new Voiture(300000, 1985, 2) ;`
  Quelles sont les méthodes appelables sur vielleVoiture ?
- En suppose que vous avez défini un objet de type Voiture de la manière suivante
  `Voture vielleVoiture = new Voiture(300000, 1985, 2) ;`
  Quelles méthodes pouvez-vous appeler sur vielleVoiture ?

En implémentant dans la classe Voiture l'interface Comparable qui permet de comparer les voitures suivant leur kilométrage, créez un tableau de 10 voitures de kilométrage au hasard et affichez-les et utilisez la méthode sort de la classe Arrays pour les trier.

un exercice complexe sur les interfaces (inspiré de l'université de Marseille)

Dans cet exercice, nous allons écrire une interface Formule qui nous permet de manipuler des formules mathématiques simples (nous nous limiterons pour le moment aux additions et aux multiplications) sous forme formelle mais également permet d'obtenir la valeur de ces formules mathématiques.

Nous définissons une formule de la manière suivante :

- un paramètre (classe Param) est une formule
- la somme de deux formules (classe Somme) est une formule
- le produit de deux formules (classe Produit) est une formule

L'exemple suivant vous donne une idée du fonctionnement de l'interface Formule et des classes Param, Somme et Produit.

```
public class Test {


    public static void main(String [] args){

    Param x1 = new Param("x1", 4.0);
    Param x2 = new Param("x2", -1.5);

    Formule f = new Somme(x1, new Produit(x2, x2));
    System.out.println(f.ecritureFormelle());
    System.out.println(f.valeurNumerique());

    Formule f1 = new Somme(x1,f);
    System.out.println(f1.ecritureFormelle());
    System.out.println(f1.valeurNumerique());
    }

}
```

Affichera :

```
(x1+(x2*x2))
6.25
(x1+(x1+(x2*x2)))
10.25
```

A partir de ces éléments :

1. Écrire l'interface Formule et le diagramme UML des autres classes ;
2. Écrire le code des classes Somme, Produit et Param

On souhaite rendre dynamique le calcul des formules de manière à ce que le changement d'une valeur des paramètres provoque un changement de la valeur finale (*cf* exemple ci-dessous)

```
x1.setValue(8.0);
    System.out.println(f1.valeurNumerique());
```

18.25

1. Modifier si cela est nécessaire vos classes
2. Comment faire pour écrire des formules avec des constantes telles que 5,0*x – 3,0 ?
3. Ajouter les classes Puissance et RacineCarrée