E8

On veut gérer des véhicules terrestres (classe Terrestre), des véhicules marins (classe Marin) et des véhicules amphibies (classe Amphibie) :

- Ils ont comme caractéristique commune la vitesse maximale qu'on définit dans une classe Vehicule. Les véhicules terrestres ont de plus un nombre de roues, et les véhicules marins un tirant d'eau, les véhicules amphibies possédant chacune de ces deux caractéristiques.
- Chaque classe dispose d'une fonction d'affichage qui affiche toutes les caractéristiques de l'objet.

Définir un graphe d'héritage de classes et respecter la contrainte suivante : aucun objet de type Vehicule ne doit pouvoir directement être créé.

Noter que les véhicules amphibies héritent "deux fois" de Vehicule et possèdent deux vitesses maximales, une en tant que véhicule terrestre et un autre en tant que véhicule marin.

E9

Récupérer les classes Point et Segment développées dans l'exercice E3.

I) Fichier de points

Avec l'éditeur créer un fichier texte de points :

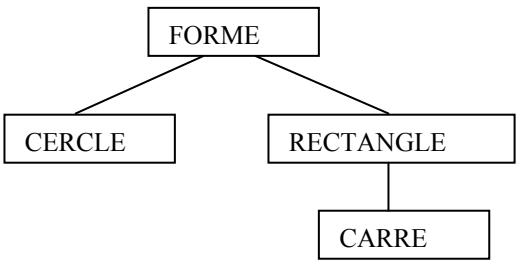
```
12.75  4.33
8  19.45
20.34  56.1
10.01  12.05
etc
```

Dans le module du "main" écrire une fonction qui lit un tel fichier et génère un autre fichier en gardant seulement les points dont la distance à l'origine est inférieure à une valeur donnée, reçue en paramètre. Le fichier résultat est formaté ainsi :

```
000: (12.75, 4.33) d=13.47
001: (8.00, 19.45) d=21.03
002: (10.01, 12.05) d=15.67
```

Chaque point est numéroté avec un numéro sur 3 chiffres, les réels sont affichés avec deux chiffres après la virgule.

II) Classes de formes géométriques



Il est nécessaire d'ajouter à la classe Segment une fonction qui retourne la longueur du segment.

1) Les contraintes à respecter sont les suivantes :

1a) la classe Forme doit comporter :

- une donnée privée *Forme* suiv* qui sera utile au 2) pour réaliser un chaînage, ainsi que deux fonctions publiques associées *void setSuiv(Forme*)* et *Forme* getSuiv() const*,
- une fonction virtuelle pure *double perimetre() const* qui calcule le périmètre,
- une fonction virtuelle pure *double surface() const* qui calcule la surface,
- une fonction virtuelle pure *void afficher() const* affichant le type de la forme, son périmètre, sa surface.

1b) Cercle, Rectangle et Carre doivent permettre la construction d'objets au moyen des constructeurs suivants :

```
Cercle (const Point&, double);          // centre, rayon
Rectangle (const Segment&, double);     // largeur, hauteur
Carre (const Segment&);                 // côté
```

Il conviendra de définir des données membres privées pour mémoriser les informations fournies par les constructeurs, en faisant en sorte qu'une classe ne duplique pas des données déjà mémorisées par sa classe mère.

1c) les classes Cercle, Rectangle et Carre doivent redéfinir si nécessaire le sens des fonctions héritées de Forme.

Tester ces classes de la manière suivante :

soient les points P1(12,6) et P2(15,6)
définir les formes suivantes et les afficher :

- cercle de centre P1 et de rayon 3.2
- rectangle de largeur P1P2 et de hauteur 8
- carré de côté P1P2

2) On définit maintenant la classe ListeFormes :

```
class ListeFormes
{
public :
        ListeFormes() ;
        void ajouter(Forme*);
        void afficher() const;
        void afficherMax() const;
private :
        Forme* tete;     // tête de liste
};
```

La fonction ajouter () ajoute une forme à la liste.
La fonction afficher() affiche les formes de la liste.
La fonction afficherMax() affiche la forme de plus grand périmètre et celle de plus grande surface.

Tester cette classe en définissant une liste contenant les trois formes créées au 1).